

UNIVERSITY OF TARTU  
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE  
Institute of Computer Science  
Software Engineering Curriculum

**Jonas Kiiver**

**NFC Security Solution for Web Applications**  
**Master's Thesis (30 ECTS)**

Supervisor: Professor Eero Vainikko

Tartu 2015

# **NFC Security Solution for Web Applications**

## **Abstract:**

This thesis compares existing and possible security solutions for web applications, analyses NFC compatibility for security solutions and proposes a new NFC authentication and signing solution using Google Cloud Messaging service and NFC Java Card. This new proposed solution enables authentication and signing via NFC enabled mobile phone and NFC Java Card without any additional readers or efforts to be made. This smart card solution can be used within multiple applications and gives the possibility to use same authentication solution within different applications.

## **Keywords:**

NFC, Java Card, Two-Factor Authentication, IsoDep, APDU, EstEID, Signing, Encryption, Authentication, Security, GCM, BroadcastReceiver, Dual interface card, Android, RSA, Certificate,

# **NFC Turvalahendus Veebirakendustele**

## **Lühikokkuvõte:**

Töö eesmärgiks on võrrelda erinevaid eksisteerivaid veebirakenduste turvalahendusi, analüüsida NFC sobivust turvalahenduste loomiseks ning pakkuda välja uus NFC autentimise ja signeerimise lahendus läbi Google Cloud Messaging teenuse ja NFC Java Card'i. Autori pakutud lahendus võimaldab kasutajal ennast autentida ja signeerida läbi NFC mobiiliseadme ja NFC Java Card'i, nõudmata kasutajalt eraldi kaardilugejat. Antud lahendust on võimalik kasutada kui ühtset kasutajatuvaustamise viisi erinevatele rakendustele, ilma lisaarenduseta.

## **Võtmesõnad:**

NFC, Java Card, 2 Faktori Autentimine, IsoDep, APDU, EstEID, Signeerimine, Enkrüptimine, Autentimine, Turvalisus, GCM, BroadcastReceiver, Kahe liidesega kaart, Android, RSA, Sertifikaat,

# **Acknowledgement**

Author of the thesis would like to give his greatest thanks to his supervisor, Professor Eero Vainikko, without whom this thesis would not have seen the light of day. He also thanks him for his patience and extreme helpfulness in guiding him with the preparation and execution of this thesis.

## Table of Contents

1	Introduction .....	5
2	NFC security solution analysis.....	7
2.1	NFC .....	7
	NFC Devices .....	8
2.2	NFC connection security .....	9
	Eavesdropping .....	9
	Data Corruption.....	10
	Data Modification .....	10
	Data Insertion .....	11
	Man-In-The-Middle .....	11
2.3	NFC Secure Channel Connection .....	13
	Public-Key Based Cryptography.....	13
	Public-Key-Cryptography Practical Considerations .....	14
	Symmetric Cryptography With Shared Secret.....	16
2.4	Web Application Security .....	17
	Two-Factor Authentication .....	18
	Three-Factor Authentication .....	19
	Proposed NFC Solution Web Application Security.....	19
	Comparison .....	20
2.5	Existing Authentication Solutions.....	21
	Image-Based Authentication .....	21
	SMS One-Time Password .....	22
	Device Generated One Time Password.....	22
	Out-of-Band Authentication.....	23
	Biometrics .....	23
	Another Application for Authentication .....	24
	Authentication Using Mobile Device NFC Emulation .....	24
	RFID Tag As Additional Security Token .....	25
	Password Authentication Solution .....	25
	Comparison .....	26
3	Proposed NFC Security Solution for Web Applications.....	29
3.1	Platform Selection .....	29
3.2	Architecture .....	29

Server .....	30
Smartphone Application.....	31
Java Card .....	31
3.3 Data Model.....	32
3.4 Security .....	34
Mobile Device Verification.....	34
Verify Authentication.....	34
Verify Signature .....	35
Security Between NFC Card and Reader.....	36
Security Between Device and Web Server .....	36
3.5 Application Flow.....	37
Authorise Device Flow .....	37
Authentication flow.....	39
Signing flow .....	41
Unauthorize Device Flow .....	43
3.6 Future Opportunities .....	45
Mobile Two-Factor-Authentication .....	45
Corporate Security .....	46
No Passwords Solution .....	46
EstEID NFC Mobile Reader .....	46
4 Conclusions .....	48
5 References .....	49
Appendix .....	52
I. Java Card Application APDU's .....	52
II. Cryptographic Algorithm Benchmarking .....	55
III. NFC Security Solution for Web Application Prototype.....	57
IV. Rest API calls.....	68
VI. Source code .....	72
VII. License.....	73



## 1 Introduction

As online security is becoming part of our everyday life and people are more aware of the different threats, there is always a need for better and innovative security solutions resistant to hacking and identity theft. There are multiple different solutions using a mobile device as an authentication endpoint, but what happens, when mobile is the actual means of internet access? Mobile is becoming the main internet access point for a large amount of people and at the moment there is no good solution to secure your mobile connection and to double-authenticate yourself without mobile being the second layer of authentication endpoint. What should be done to improve the authentication via mobile devices and how can secure authentication be enabled within a mobile device, using it as the only entry point?

Commonly used authentication solutions involve a mobile device as an authentication endpoint by communicating with it via some third party channel like cellular network, image recognition software or internet. There are multiple different authentication solutions out there, many of them are using NFC [24] for additional authentication, but all of them are using NFC simply to read the unique card ID or to emulate the mobile device into being an NFC card - by doing that, they enable a second level of authentication with only one “something user has” token and do not provide an additional level of authentication and are lacking security.

Therefore there is a need for a security solution that meets the following needs:

- a) Enables two-factor authentication for mobile devices – Additional means of authentication while using only mobile device as access point.
- b) Add another physical layer to the authentication process – Another “what user has” token within authentication flow.
- c) Resistant to internal attacks – solution must not be internally attackable, if device is compromised.
- d) One solution for all – one solution usable within multiple different applications and implementable with ease, requiring the user to remember less.
- e) Secure storage – Securely store user certificates and private keys.

In order to provide a solution matching those needs, this thesis compares different existing solutions to find pros and cons within these solutions and examines the level of security existing in commonly used means of authentication. In order to improve the field of security solutions, author proposes his own secure authentication solution for web applications involving NFC, GCM [18], Android mobile phone and web application server. To prove the validity of NFC as security endpoint, an analysis of NFC security and possible attack vectors were made. Based on this analysis, a decision was made, if NFC and the communication between two NFC devices (NFC Java Card and NFC mobile device) is secure enough to host authentication solution for high security risk web applications.

The solution proposed by author (shown in Figure 1) consists of a workstation, a mobile device and a NFC enabled Java Card. The workstation is a regular user computer running any operating system with the capability to access a web browser. The mobile device must be a device with NFC communication capabilities and smart card must be a Java Card

with NFC interface. By using only those components author proposes a new authentication functionality, that provides a second level of authentication via NFC card using a mobile device as a card reader, also authenticating mobile device within the authentication process. Additionally, the author is proposing solution providing the functionality to sign different user tasks and actions inside the web application via the same solution used for authentication. Signing is done by additional passphrase requested from the user and is separated from the authentication flow.



Figure 1. Proposed NFC solution concept.

## 2 NFC security solution analysis

### 2.1 NFC

NFC (Near Field Communication) is radio frequency based communication between different NFC enabled devices or smart cards. Connection between endpoints can be established by touching two devices together or bringing them into proximity (distance of 20 cm or less)[28]. Connection is established via electromagnetic induction between two loop inductor antennas and it operates on ISM Band 13.56 MHz radio frequency, having transfer rates between 106 Kbit/s and 424 Kbit/s [15].

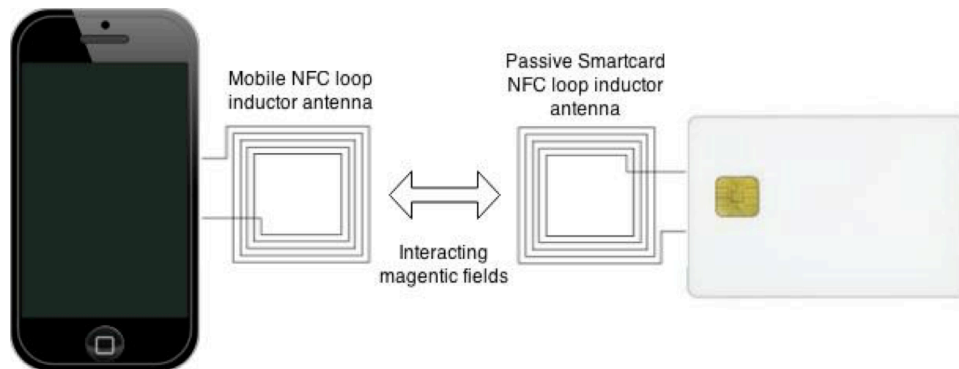


Figure 2. NFC communication core.

There are two different endpoint modes in NFC communication - Active or Passive. Active modes use Amplitude Shift Keying (ASK) to send the data [6]. Meaning RF signal is modulated with data according to coding scheme. If transfer rate is 106 KBaud, then Miller encoding scheme [42] is used and if transfer rate is bigger than 106 KBaud then Manchester encoding scheme [41] is used. Both of those coding schemes send one data bit in given time slot. The time slot is divided into two separate halves, as half bits.

- In Miller encoding [6] - 0 is encoded with delay into first half and 0 with no delay is encoded into second half bit. 1 is encoded with no delay into first half and 1 is encoded with delay into second half.
- In Manchester encoding [6] - the encoding is done similar to the Miller coding, but instead of having pause in either half bit, a whole half bit is either modulated or a pause.
- Modified Miller encoding [6] - additional rules are used for encoding zeros. If 1 is followed by 0, the two following half bits will have a pause, but in modified Miller encoding, the following two half bits are encoded without a pause.

Communication speed also determines the strength of the modulation. For 106 KBaud rate transfer, the modulation is 100% - meaning the pause in RF signal is actually 0, but for transfer rates greater than 106 KBaud 10% modulation ratio is used. Using 10% modulation ratio means that a pause in RF signal is not actually 0, but it is around 82% of the non-paused signal strength. This discrepancy in the modulation strength is key aspect of security flaws in NFC communication and this is further analyzed in section 2.3. In passive mode, the communication data is encoded using weak modulation, and Manchester encoding with 10% modulation strength is always used.

## NFC Devices

There are 3 types of NFC devices [6]:

- a. Active devices working as NFC readers - Devices use external power to power NFC electromagnetic field, sharing power with NFC tags, powering them up and communicating with them.
- b. Passive devices working as NFC tags - NFC tag solutions with no external power supply. They are started when entering into NFC electromagnetic field and their process lifecycle is equal to the duration of staying in the electromagnetic field.
- c. Active/Passive devices – These devices have the ability to be both an active NFC reader and a passive NFC.

Using previously described types of NFC devices, there are three different communication configurations possible between two NFC enabled devices described in Table 1.

Table 1. NFC communication configurations.

Device A	Device B	Description
Active	Active	Describing NFC peer-to-peer solution as the data sending device generates an RF field and acts as an active device and the data receiving device acts as a passive device. When device A sends data, device B is passive, and when device B sends data, device A is passive.
Passive	Active	Device B is acting as a passive device and device A is generating an RF field and is active.
Active	Passive	Device A is an active device and device B is passive

Additional to the active/passive modes of NFC devices, there are also two possible roles a device can play in NFC communication.

- a) Initiator - A device that initiates NFC communication between two devices
- b) Target - A device that receives a communication request and responds to it

As NFC communication is based on message and reply concept. Therefore, if we have an active device A and a passive device B, then A is able to start the communication and receive responses from device B, but device B is not able to start the communication itself and can only reply to requests made by device A. The possible combinations of an NFC device's roles and modes are described in Table 2.

Table 2. NFC communication configurations based on initiator and target.

	<b>Initiator</b>	<b>Target</b>
<b>Active device</b>	Possible	Possible
<b>Passive device</b>	Not Possible	Possible

In addition, it should be mentioned, that NFC communication is not limited to paired devices and one initiator can communicate with multiple devices at the same time. This means the NFC communication is started simultaneously, but the sending device must select the device the message is meant for and other devices must ignore the sent message, therefore the actual communication between devices is still one-to-one.

## 2.2 NFC connection security

As NFC communication is established without any physical restrictions, the communication between devices is open to the public, exposing it to multiple threats. NFC communication can be disrupted by corrupting the data, listening to it or replacing data while transferring.

### Eavesdropping

As NFC is a wireless technology, then it is certain that eavesdropping can be an easy security breach as devices communicating via NFC use RF waves to talk to each other. Attacker can use an antenna to also receive the communication sent between the devices and with sufficient knowledge of RF waves and how data is coded into them, the attacker can extract communication data from the received communication. The equipment needed to do such eavesdropping must be assumed to be available to an attacker, as equipment needed to create such an attack is publicly available to everyone and no special equipment is needed. As NFC communication is claimed to work only within 20 cm or less proximity between devices, a question arises: how close must the attacker be to the communicating devices to extract readable data from the received RF waves? This question cannot be answered with full accuracy, as there are multiple parameters that affect the distance of extracting readable data from RF waves. Parameters that affect the range of NFC radio frequency waves are [6]:

- a) Power sent out by the NFC device - The amount of power used to send RF waves affect the range of RF waves, as more power produces wider range, exposing the device more for attacks
- b) Location - RF waves travel differently in different environments.( e.g. metal walls, underground etc.)
- c) NFC characteristics of sender device - RF wave range depends greatly on the sending phone RF characteristics (e.g. antenna size and geometry, mobile case shielding effect etc.)
- d) Quality of the attackers devices - Distance between the attacker and the communication NFC devices depend vastly on the quality of the attackers devices (antenna size and geometry, RF signal decoder, receiver)

Due to those parameters, an exact distance of RF waves in NFC communication can not be given and if the distance is given, then it can only be correct for a given set of parameters and not for all possible parameter values [28].

Additionally, there is a huge difference in what mode the sender is operating. If the data sender is operating in passive mode, then the sender's RF field is powered by the other party of this communication and the field's distance is much smaller than the RF field distance on the active sender, making it much harder to eavesdrop information sent by the passive sender. Touch estimate on how big can the distance be to successfully eavesdrop NFC communication is about 10 m for active devices and 1 m for passive devices [6].

As NFC is a wireless technology, it can not protect itself against eavesdropping. Even if data transmitted in passive mode is harder to eavesdrop on, it is not sufficient to ignore the threat. Therefore, the only real solution against eavesdropping is to create a secure channel connection with software. This solution is further described in section 2.2.3.

### **Data Corruption**

Additional to the eavesdropping discussed in section 2.2.1.1, an attacker can also try to modify eavesdropped data that is transmitted via an NFC connection. One use case is, that an attacker simply disrupts the communication between devices by corrupting the data sent by other devices. This can be achieved by using eavesdropping during the communication and sending valid frequencies of data spectrum within a correct time period in communication. Correct time is calculated from the NFC communication modulation scheme. This attack simply does not allow the user to achieve correct communication results between devices [6]. NFC devices can detect the data corruption attacks by checking the RF field and the power of the transmitted RF waves. Power needed to create data corruption in NFC communication is much greater than the power used to normally communicate with NFC devices. Therefore all these attacks should be discoverable [6].

### **Data Modification**

A Data Modification attack is similar to a Data Corruption attack, but with the difference, that attacker-sent manipulated data is valid in NFC communication. Implementation-ability of the attack highly depends on the applied strength of signal amplitude modulation, as decoding of the signal differs for 100% and 10% modulation. When the communication is using 100% modulation, the decoder checks both half bits of RF signal on or RF signal off. To inject some valid data into the NFC communication, the attacker must fill the pause in the modulation with carrier frequency and generate a pause of RF signal, what is then received by correct party of the NFC communication. This means that the attacker must send RF signal in a so perfectly overlaps with the original RF signal, making the original signal a zero signal in receiving decoder, which is almost impossible to accomplish.

In modified Miller encoding, in case of subsequent bits, the attacker can change the second bit to zero by modifying the pause that encodes the second bit. Doing that, the decoder would see no pause in the second bit and would decode it as zero, because the first bit was 1. Therefore, in 100% modulation, the attacker can only change bit value from 1 to 0, in case bit is preceded by a bit of value 1, but never vice versa [6]. Using 10% modulation, the decoder is measuring signal levels of 82% modulation and 100% modulation, compares them, and if they respond to the correct range, the signal gets decoded. An attacker could try to increase the 82% signal to make it match the 100% signal, making it appear as full signal and making the actual full signal appear as 82% signal [29]. Doing that, the decoder would decode false bits into correct ones and correct bits into false ones.

Whether the attack is feasible, mostly depends on the input range of the receiver. It is likely that the higher signal levels would exceed the input range of the receivers' decoder, but the threat cannot be ruled out completely. We can say, that a data insertion attack is not achievable for 100% Miller encoding, but is possible for 10% Manchester encoding [6].

Possible solutions to avoid data modification attacks are:

- a) 106 KBaud transfer rates - By using only 106k Baud transfer rates, the attackers can not achieve data insertion attacks. However using this transfer rate makes the communication most vulnerable to eavesdropping described in section 2.2.
- b) Check RF field - Continuous check by sending the device to detect interference in RF field and stopping the data transaction when this kind of interference happens.
- c) Secure channel - Create an encrypted channel between two devices. Further described in section 2.3

### **Data Insertion**

Attacker inserts additional messages into NFC communication. This sort of attack can only happen, when the device answer takes a very long time. This means the attacker must send the data before the sending device sends it (replies to request) [29]. This attack is successful only if the attacker sends the whole data before the replying device starts to answer. When the two data streams are trying to transmit at the same time, then data collision happens and data gets corrupted [6].

There are multiple solutions to avoid this kind of attacks:

- a) No delay - Making the device communication without delays, so the answering device answers immediately, not giving the attacker enough time to interfere. (Attacker can not be faster than the correct device, they can be both the same speed, but sending the same data at once from two different sources results in data collision).
- b) Channel listening - Constantly listening to the communication channel so the devices can detect a 3rd party trying to interfere communication.
- c) Secure channel - Create an encrypted channel between two devices. Further described in section 2.3.

### **Man-In-The-Middle**

A typical man-in-the-middle attack is when two parties want to talk to each other and are tricked into a three party communication by an attacker. For example: Device A wants to communicate with Device B, but at the start of the communication, Device C as an attacker tricks both devices into communication with Device C instead of each other. Device A and Device B are convinced that they are communicating with each other - shown in Figure 3.

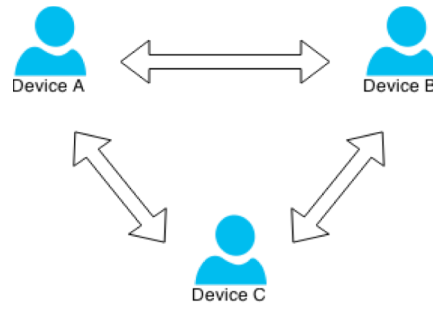


Figure 3. Man-in-the-middle communication directions.

This attack can happen regardless of a secure channel, as the secure channel endpoint is switched by Device C and Device A and B must have sufficient security knowledge to eliminate Device C as a valid endpoint for Device A or B. A Classical man-in-the-middle attack starts with key authentication, where Device A and Device B want to establish a secret key to set up a mutual secure channel. Device C creates secure channels with both Device A and Device B and the communication between Devices A and B seem completely normal to them.

How does the man-in-the-middle attack behave in case of NFC communication? Assuming Device A is in active- and device B in passive mode then the situation is as follows. Device A has generated an RF field to communicate with Device B. If Device C is close enough, it can read the data sent by Device A. Device C disrupts the transmission between A and B, so B does not receive the data. This can be achieved by Device C, but it is also visible to Device A and if A detects the attack it can stop the key agreement protocol communication. But to further analyze the man-in-the-middle attack we must assume that Device A does not detect the disturbance in the key exchange protocol. When Device C has successfully blocked A and B communication and received communication request by A, Device C must generate RF field to communicate with device B, meaning two RF fields must be active at the same time. It is impossible to align the two RF fields, making it impossible for Device B to understand the data sent by Device C. Because of this and the fact that Device A can easily detect the changes in the RF field it is evident that the man-in-the-middle attack is practically impossible for NFC communication.

The only other possible solution for the NFC man-in-the-middle attack is if Device A and Device B both use active modes. Then again assuming that Device A does not detect disturbance in the RF field the attack is possible. This is due to the Active-Active communication. When Device A has sent data to Device B (Intercepted by Device C), device A turns off its RF field. Now Device C turn on RF field and send data to Device B. Now the problem here is, that Device A is also listening and expecting an answer from Device B. Instead of a response from Device B, it will receive an answer from Device C and can yet again detect the problem and stop the communication. Only possible man-in-the-middle attack in this case is switching the communication endpoint for Device A from Device B to Device C, and Device C must know what kind of answers Device A is waiting for. Therefore it can be said, that man in the middle attack is practically infeasible for NFC communication [6] and it does not need additional solutions to counter fight it. It is recommended, however, to use Active-Passive communication modes for high security risk NFC communication. In which case the active party should detect all abnormalities in the NFC field.



## 2.3 NFC Secure Channel Connection

Having a secure channel between two NFC endpoints is clearly the best solution to avoid any kind of attack, as NFC has proven to be resistant to man-in-the-middle attack (described in section 2.2). Therefore we can assume that it is secure to establish a secure channel connection between two NFC devices. In our case, it is a connection between an active NFC mobile device and a passive NFC smartcard. To achieve that, it is possible to use either Asymmetric cryptography or Symmetric cryptography.

### Public-Key Based Cryptography

Public key based cryptography, also known as asymmetric cryptography, is a cryptographic algorithm that uses two separate keys for each communication participant. Each participant has a set of private and public keys, that are mathematically linked. Public key is used for encrypting the plaintext into cipher text and for verifying the digital signature. Private key on the other hand is used to decrypt cipher text and to create digital signatures. It is called asymmetric, due to the use of two keys performing opposite functions. The security lies within the private key of the key pair. Public-Key cryptography relies wholly on the fact that it is computationally infeasible for a correctly generated private key to be deduced from its public key pair [8]. Therefore, the public key can be securely published to other parties without compromising the security of the protocol. Some public key algorithms provide key distribution and secrecy, others provide digital signatures and some provide both. Therefore, a suitable public key cryptosystem must be chosen to handle key exchange and establish a secure channel. A standard RSA or Elliptic curve based cryptography is suited for this kind of problems.

Rivest Shamir Adelman (RSA) [26] is one of the oldest public key cryptography algorithms and also the most used one at the moment. The RSA cryptosystem is based on the high computation cost of factoring, which means that having sufficient computational resources and time, an adversary cannot obtain the private key from the key set via factoring. Factoring is not the only method to break RSA, but at the moment no other method has proven successful either [10]. The RSA public and private key are generated based on the algorithm [25]:

- a) Select two random prime numbers  $a, b$ , always in a way that the bit length of  $a$  is approximately the same as of  $b$
- b) Compute  $n = a * b$
- c) Compute  $\phi(n) = (a-1)*(b-1)$
- d) Select a random integer  $e$  so that  $e < \phi(n)$  and  $\text{gcd}(e, \phi(n)) = 1$ , after that compute integer  $d$ , with  $e*d \equiv 1 \pmod{\phi(n)}$
- e)  $(n, e)$  is public key and  $d$  is private key

To encrypt data  $m$  with public key then result  $s = \text{hash}(m)^d \pmod{n}$  and to verify the result then hash  $h = s^e \pmod{n}$

Elliptic Curve Cryptography (ECC) is described with an equation [27]

$$y^2 = x^3 + ax + b, \text{ where } 4a^3 + 27b^2 \neq 0$$

ECC public and private key are generated based on the following algorithm [10]:

- a) Find elliptic curve  $E(K)$ , where  $K$  is finite field such as  $F_p$  or  $F_2$ , and find point  $Q$  on  $E(K)$ .  $n$  is the order of  $Q$
- b) Select pseudo random number  $x$  in a way that  $1 \leq x \leq (n - 1)$
- c) Compute point  $P = xQ$

d) ECC key pair is  $(P, x)$  where  $P$  is public key and  $x$  is private key

ECC uses smaller keys than RSA encryption algorithm [10], which is vital in the proposed NFC solution, as Java Card memory sizes needed to store keys and certificates are minimal. In terms of key generation ECC outperforms RSA at all key lengths with massive differences, as RSA 1024 key length key pair generation takes 0.16 seconds, but ECC manages to generate responding key with 0.08 seconds. In terms of signature generation, RSA outperforms ECC with the 0.01 seconds and 0.15 seconds respectively. In terms of signature verification, RSA also outperforms of ECC, with the difference of 0.01 seconds and 0.23 seconds respectively.

### **Public-Key-Cryptography Practical Considerations**

Forward Public Key Encryption (FPKE), is an encryption that makes sure, that communication is kept secret during transmission, as all data moving along the communication channel is encrypted. It assumes that both receiver and sender are in possession of their own private key and other parties' public key. In order to send a message from one party to another using FPKE, the sender uses the receiver's public key as an encryption key and encrypts the contents of the message to be transmitted. After encryption, the message is sent to receiver. The receiver uses their own private key to decrypt the message contents. Using the receiver's public key to encrypt the data is also useful for preserving the confidentiality of the message, as only receiver with the appropriate private key can decrypt the message. Therefore once the sender has encrypted the message with receiver's public key, it is impossible for the sender to decrypt it. Although, FPKE do not protect against the non-repudiation problem, as the message could be sent by anyone who has access to receiver's public key. Therefore, the author argues, that FPKE alone is not secure enough for proposed NFC security solution and there is a need to further analyze possible usages of Public Key Encryption solutions.

Inverse Public Key Encryption (IPKE), also known as digital signature, is based on the sender encrypting the message with his private key and adding the result to the message as signature. The receiver receives the message along with the digital signature of the message, and then uses the sender's public key to verify the message signature, making sure that the sender is correct. IPKE is the complete opposite of the FPKE, as it ensures the non-repudiation of the message, but it does not secure the message itself. Therefore the author claims that IPKE is also unusable in proposed security solution, as it does not protect against eavesdropping.

Enveloped Public Key Encryption (EPKE) is applying public-key cryptography and both ensuring that the transaction communication is handled confidentially, and that the contents are protected against modifications and non-repudiation. This method is mainly used in open network environments similar to the environment in the proposed solution. EPKE makes use of the Transport Layer Security (TLS) or Secure Sockets Layer (SSL). EPKE consist of both Forward Public Key Encryption and Inverse Public Key Encryption, creating the foundation of Enveloped Public Key Encryption. In order for the EPKE to work, the following is required:

- a) Each communication party has their own unique set of public and private keys.
- b) Every participant's private and public key set must be mathematically related, as party's private key must be able to decrypt data encrypted by the party's public key and party's public key must be able to verify data signed with the party's private key.

- c) The private key is kept private and only the owner of the private key knows it, whereas the public key is published to other communication parties.
- d) Communication parties must be aware of the public keys used by other trusted parties, making sure the communication is only allowed between trusted participants.

In order to send the message using EPKE, the message must be encrypted by the sender's private key and the result must be added to the message, ensuring non-repudiation of the message. Then the sender encrypts the message together with a digital signature using the receiver's public key. Now the message is in a so called digital envelope and is sent to the receiver. After receiving, the receiver decrypts the message using its private key, revealing the message and the digital signature of this message. Now the receiver uses the sender's public key to verify that the message is received from the correct sender, and validating that the message is in fact correct. The author claims that EPKE is the most suitable form of Public Key encryption to be used within the proposed solution, as it is secure from eavesdropping and enables to verify the sender of messages.

The proposed solution uses Java Cards to handle one part of the communication protocol. This means the computational powers of these cards are limited, but due to the computationally complex nature of the RSA encryption algorithm, the time used to encrypt/decrypt amounts of data can increase vastly depending on the amount of data decrypted/encrypted. To overcome the issue of handling bigger amounts of data, the author suggests to use data hashing prior to the private key encryption, reducing the amount of data encrypted to fixed amount based on the hashing algorithm. Using SHA1 hashing algorithm the large amounts of data can be reduced to 160 bits of data that need to be encrypted. Both sender and receiver must be aware of the signature hashing and signing functionality, otherwise the digital signature comparisons will not match. The only downfall of using this sort of hashing prior to encryption is that smaller than 160 bit data gets hashed to 160 bit data, making the amount of data encrypted bigger than it originally was. Table 3 shows the difference between SHA1 hashing and RSA encryption signing in one second. There is a difference of over 100 times between SHA1 hashing and RSA encryption, as shown in Table 3. Further analysis is described in Appendix 2.

Table 3. SHA1 and RSA encryption benchmarking.

Block size	SHA1	RSA
512 bit	5807998.5 hash / s	7334.6 sign / s
1024 bit	416352.3 hash / s	2099.3 sign / s
2048 bit	57689.7 hash / s	397.3 sign / s

The author's proposed Public Key cryptography secure channel solution proposes the use of EPKE together with session handling and symmetric cryptography. As the sender and receiver have knowledge of each others public keys, they can obtain the knowledge during communication establishment or have it predefined for them. When communication is started, the initiator verifies the receiver by requesting a public key. On receiving receiver's public key, the sender generates a session key and encrypts it using the receiver's public key. The receiver then receives the session key by decrypting the message with its private key. Now sender and receiver have a shared temporary session

key they can use to encrypt the messages. By encrypting the messages with shared knowledge, the sender only needs to encrypt the message, without even digitally signing the message, as the temporary shared session key assures the communication is coming from the correct sender. This also allows the receiver and sender to use symmetric encryption which is faster than asymmetric encryption. This proposed solution is the same, as web TLS protocol is handling the key verification, secret sharing and channel creation.

### **Symmetric Cryptography With Shared Secret**

Symmetric cryptography is a single key cryptography, where the same key is used to encrypt and decrypt message data. This cryptography is mainly used to encrypt data with shared knowledge, so that whoever needs to use the encrypted data, they need the same encryption secret to decrypt it. Symmetric cryptography can be used for the proposed security solution, by simply sharing the secret between NFC mobile devices and NFC smart-cards. In this case the sender encrypts the message with the shared key and a suitable algorithm before sending the encrypted message to the receiver. Receiver decrypts the message with the same key obtained during initialization. As per Java Card 3.0.4 API documentation, the same Java Card version the Estonian ID card is based on, Java Card supports multiple different symmetric cryptography algorithms that can be used for this sort of shared secret security channel. Supported symmetric cryptography algorithms are AES, DES, 3DES and SEED [11].

Author rules out SEED algorithm, as it is mainly used in South Korea and not many solutions support it. Java Card does support it, but it works with a 64-bit key, which is too weak for our liking [12]. The author also rules out DES algorithm, as it is an older, less secure version of DDES algorithm, using a single 64-bit block cipher under a 56-bit key [13]. That leaves us with 3DES and AES algorithms. In the following section, the author analyses 3DES and AES symmetric encryption algorithms and tries to find the best suitable algorithm for the proposed NFC security solution.

Triple Data Encryption Standard (3DES) was developed on top of DES, correcting the flaws in the single DES encryption algorithm. DES key is 56-bit and 16 cycle Feistel system is used with 16 48-bit sub keys permuted from the one 56-bit key - One key for each Feistel cycle [1]. Algorithm basic for 3DES are the same, as they are in DES encryption, but the key size is triple the size of the key in DES algorithm. This is simply handled by doing 3 different encryptions with 3 given keys on the same block of data, therefore all operations of DES encryption must be done three times, each time using different part of the given 168-bit key. 3DES data encryption throughput is slightly smaller than DES throughput, coming from 3 times the encryption done with the message blocks, but it is not 3 times smaller as we would to assume [4].

Advanced Encryption Standard (AES) was developed to replace DES encryption algorithm, and is based on substitution and permutation. AES uses Rijndael algorithm, which has a fixed block size of 128-bit and key size of 128, 192 or 256 bit. Key size determines the number of repetitions of transformation rounds done to get the cipher text output.

- a) For 128-bit key, 10 repetition cycles are made
- b) For 192-bit key, 12 repetition cycles are made
- c) For 256-bit key, 14 repetition cycles are made

Each repetition consists of 4 processing rounds:

- a) SubBytes - Each byte in a state matrix is replaced with a sub byte, using an 8-bit substitution box, also known as Rijndael S-box
- b) ShiftRows - Shifts bytes in matrix rows by a certain offset
- c) MixColumns - Takes four bytes from column and outputs four bytes, where input bytes affect all four output bytes via linear transformation
- d) AddRoundKey - Adds derived sub key to round

Based on the research of [4], we can also assume that AES encryption is slightly better at encrypting and decrypting plain text, hex or byte data and that the AES key size affects the cipher text computation time very little.

There is no vital difference between AES and 3DES encryption when dealing with small amounts of data, as encryption time only becomes evident when encrypting or decrypting data bigger than 5 megabytes [4]. As the author's proposed system uses Java Cards to handle data encryption and decryption, the data sent between the Java Card and mobile device never exceeds 5 megabytes, as the maximum single message size is 32 kilobytes [11]. It would take  $5 \times 10^{21}$  years to break AES 128-bit encryption [1], but it only takes 800 Days to crack 3DES 112 bit key. Based on these findings, we can be certain, that AES encryption algorithm is secure and fast enough to use in the proposed NFC security solution. Based on these results, it is safe to claim, that the AES encryption would be the best choice for the proposed NFC security solution.

## 2.4 Web Application Security

Security is constantly the main concern when building a new web application from the ground up. There are multiple security solutions out there that have single-factor authentication and multiple-factor authentication schemes. Picking the correct security solution for your application depends largely on the security level needed for the application and how paranoid your system must be. The top four security breaches for web applications are to do with cross-site scripting, information leakage and broken access controls [16].

- a) Cross-site scripting is an attack where the attacker exploits the user logged in functionality and accesses other user data by brute-forcing himself/herself as another user, having the knowledge of how the site is scripted for every user.
- b) Information leakage is an attack, where the application handles errors badly, and leaking information through thrown errors.
- c) Broken access control is an attack that enables attacker to access someone else's data with their own account.
- d) Broken authentication is an attack where the attacker can authenticate himself/herself as someone else and use the system with someone else's account.

Broken authentication attacks are the main attack type the author's proposed solution is trying to fix. This is the only thing that can be generalized throughout every web application. Fixes for cross-site scripting, information leakage and broken access controls depend heavily on the web application they are used in and need to be prevented during the development of the web application. Therefore we are looking into different ways of web application authentication compared to the author's proposed one.

Single factor authentication uses only one type of authorization and is open to multiple different attacks - eavesdropping, dictionary attacks replay attacks etc. [7] Therefore, the single factor authentication can be completely ruled out when building a web application

with high security needs. After ruling out single-factor authentication, secure web application development is left with multiple factor authentications to pick the correct one for the respective solution.

## **Two-Factor Authentication**

Most commonly used authentication form is the two-factor authentication (TFA) where the authentication passes two different devices or tokens, both in the possession of the same user, making it twice as secure, as single-factor authentication. Two-factor authentication can give us three guarantees [3], as we can ask for three types of evidence for identity:

- a) Something material the user has (Smartcard, mobile device etc.)
- b) Something the user has remembered (Password, picture etc.)
- c) Something the user is (Biometrical - fingerprint, picture etc.)

In order to achieve TFA we need to choose two types of proof from the given list. Usually a user-remembered password phrase and user material object is used, as according to [3] biometrical authentication is the weaker form of security authentication. Biometrical factors can be easily copied - fingerprints can be retrieved from any surface you touch without gloves or protective gear. Even the mobile phone can be used to copy your biometrics - therefore biometrics is yet to be advanced to use it successfully in secure solutions. When creating an authentication scheme, where two device are in the loop of authentication, it is crucial to make sure that both of these devices are in control of the same user [7]. Therefore the registration phase of the devices must be secure and handled only once, at sign up or device switching. if the device registration phase is handled correctly, and authentication is done using two different endpoints, the system can be more sure that the user authenticating himself/herself is actually the user he/she claims to be. The device used for two-factor authentication can differ according to the selected authentication solution - mobile device, smart card, token generator, service, computer etc.

Most similar to the author's proposed NFC smartcard authentication from two-factor authentication, is the web application smart card authentication. A solution, where a smartcard is used together with a smartcard reader to authenticate a user to the system. The Estonian ID card solution can be taken as a perfect example - it allows users to log into different web applications by simply authenticating himself/herself via an Estonian government-issued smart card. Regular EstEID solutions work by simply asking the user for username and then prompting the user directly to the smart card authentication. This solution that can not be considered 100% regular TFA, as it requests the user to input his/hers remembered password on the smart card instead of the web application, but it still ticks all the needed boxes for two factor authentication [2].

The Estonian ID card solution can be altered to work similarly to a regular TFA, prompting the user for a password together with a username, and requesting the user to input password for the smart card also, but this kind of a solution is working against user experience and common practices of TFA. The author's proposed NFC solution smart card works similar to the regular EstEID smartcard, providing all the same functionality via NFC as the Estonian ID card provides via smart card reader, but it can't be considered as a rival solution for EstEID, as NFC security solution can act as an extra feature of EstEID solution.

### **Three-Factor Authentication**

Similar to the two-factor authentication, three-factor authentication uses the same three evidence of identity, but it is using biometrics in addition to the user-remembered passphrase and the physical device. Biometric authentication is an identification of user via human characteristics - fingerprint, voiceprint and iris scan [9]. These characteristics are believed to be reliable, as they hold vast amount of high-entropy information that can not be lost or forgotten. A third biometric authentication factor can be added to the simple two-factor smart card authentication by just adding another step to pass in biometric data to authentication flow. Together with adding biometric data to the authentication flow comes the responsibility to handle biometric data by allowing user to change it similar to a password change [9]. But having three-factor authentication, computational costs are multiplied, as biometric validation requires high computational power and the additional security received from it, is not that secure, as biometrics can be easily replicated. In conclusion, adding a third factor to TFA is costly and not very beneficial, as it does not eliminate any extra attack types that the two-factor authentication does not already address.

### **Proposed NFC Solution Web Application Security**

The author's proposed NFC web application security solution can be qualified as an advancement of the two-factor authentication and we can go to the extent of calling it 2.5-factor authentication. The proposed solution uses two user-remembered passphrases:

- a) Web application passphrase
- b) Smart card passphrase

In addition to that, it uses two user verified devices:

- a) NFC mobile device
- b) NFC smart card

The main authentication functionality is based on the regular smart card two-factor authentication. Similar to [14], this solution contains 4 + 1 different authentication phases:

- A. Registration phase
  - a. System distributes customer NFC smart cards with PIN envelopes to users.
  - b. User authenticates mobile device by entering their username and password within NFC mobile application.
- B. Login phase
  - a. User tries to login into web application. Web application prompts the user to select login device view
- C. Authentication phase
  - a. User receives verification request to selected mobile device. User authorizes himself/herself with the given NFC card or Estonian ID card by entering PIN into mobile device connected with appropriate NFC card
- D. Password change phase
  - a. User opens mobile application and wishes to change NFC card PIN. User verifies himself/herself with PIN2 and is able to enter new PIN1. This is completely offline phase and everything is handled by mobile device and NFC card
  - b. User requests to change web application passphrase, authentication is carried out similar to the Authentication phase and user gets to change the passphrase
- E. Device change phase (Additional phase)

- a. User can add, remove or switch accounts related to mobile device by being in possession of the username, NFC card and password.

What makes the author's proposed solution unique is the fact that it can verify two different customer devices, allowing the user be in possession of two authorized devices (Both smart card and mobile device). By doing that the solution adds an additional security layer on top of the regular smartcard authentication solution, allowing the user to use their mobile phone as smart card reader, verifying the user mobile device at the same time. This functionality is missing, when using regular smart card readers. Similarly to the Estonian ID card, the user NFC smart cards hold 3 different user PIN numbers, making it possible to request different level of authorization when performing different tasks. PIN numbers unlock the access to user certificates held on the NFC smart card. With these certificates, user can either authorize or sign requests.

### **Comparison**

In order to prove the legitimacy of the author's proposed solution, a comparison of different web application authentications against the author's proposed one is needed. To give a better comparison we take into account different web application authentication possibilities. Therefore we compare Single-Factor, Two-Factor, Three-Factor and the author's proposed authentications. Based on Table 4 we can completely rule out Single-Factor Authentication and Three-Factor Authentication as competitive solutions for author proposed NFC security authentication solution. Two-Factor authentication can be considered as a main competitive solution for the NFC security authentication solution, but they cannot be compared as equal, as the NFC security solution is an improvement of Two-Factor Authentication. Both solutions offer better security with multiple security layers at the cost of user experience. Both require an extra step from the user and take a bit longer to successfully authenticate the user than the Single-Factor Authentication does. Based on this, it is safe to say, that the author's proposed NFC security solution is a valid Two-Factor Authentication improvement and can be taken as a competitive authentication solution for other Two-Factor Authentication solutions.



Table 4. Single-Factor, Two-Factor, Three-Factor and Author Proposed authentication solution comparison.

	Advantages	Disadvantages
Single-Factor Authentication	<ul style="list-style-type: none"> <li>+ Simple</li> <li>+ Fast</li> <li>+ User friendly</li> <li>+ Single point of failure</li> <li>+ No service integrations</li> </ul>	<ul style="list-style-type: none"> <li>- Only one protecting pass-phrase</li> <li>- No security endpoint</li> <li>- Unsecure</li> <li>- Easy to bypass</li> </ul>
Two-Factor Authentication	<ul style="list-style-type: none"> <li>+ User friendly</li> <li>+ Fast</li> <li>+ Reliable</li> </ul>	<ul style="list-style-type: none"> <li>- One security endpoint</li> <li>- Complex user experience</li> </ul>
Three-Factor Authentication	<ul style="list-style-type: none"> <li>+ Multiple layered security</li> <li>+ Third biometrical factor</li> <li>+ Big future opportunities</li> </ul>	<ul style="list-style-type: none"> <li>- Slow</li> <li>- High computing requirements</li> <li>- Easy to bypass</li> <li>- Hard to develop</li> <li>- One security endpoint</li> </ul>
Author NFC Authentication	<ul style="list-style-type: none"> <li>+ 2 security passphrases</li> <li>+ 2 separate security endpoints</li> <li>+ User friendly</li> <li>+ Secure</li> <li>+ Reliable</li> </ul>	<ul style="list-style-type: none"> <li>- Complex user experience</li> <li>- Multiple service integrations</li> </ul>

## 2.5 Existing Authentication Solutions

In order to give a better estimate of the proposed solution's efficiency and profitability, there is a need for competition research. As this solution is one of a kind and has no direct competition solution wise, general estimation of authentication solutions must be provided and comparison between different existing authentication and signing solutions has to be made. After examining different solutions, a decision was made, if the proposed solution brings a better authentication solution to the table, or is it just another authentication solution similar to the variety of different solutions out there. In order to analyze the different authentication and signing solutions, a solution with different purpose and architecture was selected, focusing the selection on Two-Factor authentication, mobile devices and NFC.

### Image-Based Authentication

Image-based authentication relies on the preliminary authentication of the user, during which the user selects multiple different images from an image grid as a passphrase (user needs to remember all of the images without correct order). During the authentication request, the user is given a 3x3 or a 4x4 picture matrix [40] with some of the user authenticated pictures and along with random pictures. Within this picture matrix, user must select the correct (previously selected) images. The given solution is supported by Confident Technologies<sup>1</sup> and is relying on the fact that person's memory is better at remembering pictures, than random passwords and this solution can not be attacked with a key-logger, as the pictures occur in a random order and in random places. This also provides the security of detecting device authorization pattern using fingerprints left on the mobile device

<sup>1</sup> <http://confidenttechnologies.com/>

<sup>2</sup> <http://www.google.com/about/company/>

<sup>3</sup> <http://openid.net/>

screen [33]. To make remembering the pictures easier for the user during security solution initialization, the user can only select certain categories instead of exact pictures and later will need to identify correct category pictures during authentication, making it possible to add large selection of images to each category.

Table 5. Image based authentication pros and cons.

Pros	Cons
<ul style="list-style-type: none"> <li>- User do not have to remember passphrases</li> <li>- Not distinguishable fingerprints left on the screen</li> </ul>	<ul style="list-style-type: none"> <li>- 3x3 picture matrix with selected 3 categories only gives 84 possible passwords</li> <li>- Vulnerable for brute force attacks</li> <li>- Pictures must be unambiguous to everyone</li> </ul>

### SMS One-Time Password

SMS One-Time Password (OTP) authentication scheme uses SMS service to send a one-time password to authorize the user. This scheme uses known user devices to provide a second level of authentication to the user. In order to access the application, the user must have the passphrase and his/her mobile device. After entering the passphrase, the application send an SMS including OTP for single-usage login, and user must enter the received password into the application. This is the example usage of TFA described in section 2.4 in this thesis. The most widely known SMS one-time password solution is Google's<sup>2</sup> two-factor authentication solution. Another way to use the SMS OTP, is to use it directly from the mobile device, as the mobile application receives SMS from the server containing the password and uses this password directly with no user actions needed. This password is only valid for one single login session and is terminated after user logs out of the application or it expires due to user inactivity.

Table 6. SMS OTP pros and cons.

Pros	Cons
<ul style="list-style-type: none"> <li>- Second factor based on "Something you have" token</li> </ul>	<ul style="list-style-type: none"> <li>- SMS delivery time may vary</li> <li>- Cloned devices receive the same SMS</li> <li>- Usability</li> </ul>

### Device Generated One Time Password

Similar to the SMS one-time password solution (described in section 2.5.2), the device generated OTP solution uses a generated limited time single-login password for authentication, with the difference, that the OTP is generated within the user device. OTP generation is in this case software based and usually requires the device time to be synchronized with server's time [32]. The best example is yet again from Google called Google Authenticator - a mobile application generating 6 figure OTP codes to be used as second authentication factor for users. This is relying on the solution, that user first authenticates himself/herself with registration passphrase and after the password authentication has been successful, a second level of authentication is requested with the user's device and Google Authenticator mobile application. Device generated OTP is better than SMS OTP, as the

<sup>2</sup> <http://www.google.com/about/company/>

generated one time password will never travel over network between device and server. OTP is generated in the mobile application and user enters the generated code directly to the requesting application.

Table 7. Device Generated OTP pros and cons.

Pros	Cons
<ul style="list-style-type: none"> <li>- Second factor based on “Something you have” token</li> </ul>	<ul style="list-style-type: none"> <li>- Cloned devices can generate the same OTP</li> <li>- Usability</li> <li>- Internally attackable</li> </ul>

### Out-of-Band Authentication

Out-of-Band authentication is based on an automated phone call from the server to the user device in order to authenticate the user. This solution is using completely different channel (from the channel the authentication request has been started) to communicate with user and by doing that, eliminating the security issue of using the same channel for all authentication activity. The authentication flow is simple, as when user tries to login to application, the server automatically calls the user mobile device and dictates a pass phrase or code for the user to enter into the application in order to authenticate himself/herself [39]. This solutions security level is similar to the SMS OTP solution (described in section 2.5.2), as it uses different channel and OTP to authenticate the user.

Table 8. Out-of-Band authentication pros and cons.

Pros	Cons
<ul style="list-style-type: none"> <li>- Server can verify mobile device together with user authentication</li> </ul>	<ul style="list-style-type: none"> <li>- Voice channel is insecure</li> <li>- Cloned device can receive the call</li> <li>- Not suitable for high risk solutions</li> <li>- Hard to implement on server side</li> <li>- Expensive</li> </ul>

### Biometrics

Biometric authentication is using user’s unique biometrical fingerprint for authentication. The characteristics used are usually based on the user’s physiological appearance and not based on the behavior of the user. [33] The most common characteristic is fingerprint, but person has multiple different unique physiological characteristics that can be used for authentication. For example the user’s tone of voice, eye iris, face metrics etc. the main problem for using biometrics as a main authentication endpoint is the fact that not many devices provide biometric identification for their user, biometric authentication devices are expensive and the solutions are also expensive to develop and integrate. Biometric authentication provides third level to authentication patterns enabling Three-Factor Authentication [38] described in section 2.4 user’s biometric fingerprint can not be changed during theft either and therefore it is not suited to be the single point of user authentication.

Table 9. Biometric authentication pros and cons.

Pros	Cons
<ul style="list-style-type: none"> <li>- Unique to every user</li> <li>- Linking account to physical person</li> </ul>	<ul style="list-style-type: none"> <li>- Expensive</li> <li>- Irreplaceable</li> <li>- Very few devices support it</li> </ul>

### Another Application for Authentication

Similar to OpenID<sup>3</sup> solutions, another application provides user authentication for the required authentication. This solution can be used if the company's do not want to create their own authentication solution and are trusting another vendors to provide a secure and reliable authentication solution. This solution provides application full authentication flow without having to implement it themselves. Excellent examples of complete such solutions are Google OAuth<sup>4</sup> and Estonian ID card, where full authentication is handled by the vendors and either successful authentication or the unsuccessful authentication response is returned to the application. Google OAuth relies on generated tokens and returns a user session token to application that requested the authentication [34]. The Estonian ID card solution provides a unique user certification-based solution, where the user can login using their ID card solution [35].

Table 10. Another application for authentication pros and cons.

Pros	Cons
<ul style="list-style-type: none"> <li>- Single authentication endpoint</li> <li>- User must remember only one solution passphrase</li> <li>- Convenient for user</li> </ul>	<ul style="list-style-type: none"> <li>- Authentication control lost to vendors</li> <li>- No certainty, if solution is hacked</li> <li>- No information regarding the actual authentication flow</li> </ul>

### Authentication Using Mobile Device NFC Emulation

This authentication solution is based on the capabilities of the mobile device emulating a NFC card. Usually the security is stored on the mobile SIM card and the mobile phone simply communicates with the SIM card to access the user's private key in order to provide the needed cryptography and authenticate the user via mobile device, but there is also a possibility of adding an encrypted secure store within the mobile application to store the secrets and provide the needed cryptography. This solution works similar to the author proposed solution, but with the difference that the mobile device itself is the NFC card, providing card functionality to NFC readers that are communicating with application user is using within workstation [36]. The user must enter a PIN to authenticate himself within the mobile NFC security environment and after successful verification the device communicates with NFC reader to send authenticated user data over to the reader. This solution adds one extra layer to regular Two-Factor Authentication by requesting user PIN

<sup>3</sup> <http://openid.net/>

<sup>4</sup> <https://developers.google.com/identity/protocols/OAuth2>

within the mobile device, allowing this solution to have 2 passcodes, similar to author proposed solution. The only downside of this solution is, that the user does not have another extra security NFC card to separate it from the mobile device he/she is using. This solution also requires an additional NFC reader – a problem the author is trying to resolve, as users usually do not have NFC readers and they are reluctant to buy additional hardware.

Table 11. NFC card emulation authentication.

Pros	Cons
<ul style="list-style-type: none"> <li>- Easy to use</li> <li>- One device for authentication</li> <li>- Multiple solutions can use the same implementation and device application</li> </ul>	<ul style="list-style-type: none"> <li>- Personalizing each user device is hard and time consuming</li> <li>- Additional reader required</li> <li>- No separation between user device and security holder</li> <li>- No secure placement of keys, if SIM is not used</li> <li>- Requires special SIM from operators</li> </ul>

### RFID Rag As Additional Security Token

This security solution is based on the NFC reader mobile phones being able to read RFID (Radio Frequency Identification) tags from NFC cards and transmitting them to the server. This means each user must have their own NFC card with unique RFID. During authentication, the user receives a request to their mobile phone to read the RFID tag and authenticate himself/herself into the application [37]. This solution is again similar to the author proposed one, but it only read RFID from the NFC card and does not utilize all the possibilities an NFC card can provide. As anyone can read the RFID from this card, it means this card is not protected and if an attacker gets access to the card and the device, then after knowing the passcode, nothing is stopping the attacker from accessing the account. Therefore this RFID only provides one physical token that needs to be kept separate from the authentication device.

Table 12. RFID as security token.

Pros	Cons
<ul style="list-style-type: none"> <li>- Easy to use</li> <li>- Additional “something you have” token</li> <li>- Simple to distribute between users</li> <li>- Cheap to implement</li> <li>- Multiple usages to the same RFID card</li> </ul>	<ul style="list-style-type: none"> <li>- No security within the RFID card</li> <li>- Easy to replicate</li> <li>- No certainty, if solution is hacked</li> </ul>

### Password Authentication Solution

Password authentication solution is the most commonly used means of authentication out there. This is based on a single passphrase and a username a user must remember in order to authenticate himself/herself within a given application.

Table 13. Password authentication.

Pros	Cons
<ul style="list-style-type: none"> <li>- Easy to use</li> <li>- Fast</li> <li>- Easy to recover</li> <li>- Easy to distribute</li> </ul>	<ul style="list-style-type: none"> <li>- Insecure</li> <li>- Not scalable</li> </ul>

### Comparison

In order to compare these different solutions and to prove the validity of the author's proposed solution, there is a need to compare the security and usability of different solutions compared to the one the author is proposing. In order to do that, security and usability comparison tables are presented. In order to compare the different security solutions security, author examines the resistance of the solution to the following attacks: [36]

- Physical observation
- Targeted impersonation - Attacker tries to impersonate a user after observing the user's authentication procedure.
- Guessing attack - Attacker tries to guess the passphrases.
- Internal observation - Attacker can impersonate a user by intercepting the user's input from inside the user's device.
- Leaks from other verifiers
- Man-in-the-Middle attack

The security comparison of different existing authentication solutions is described in Table 14. If column is marked with 'O' then the given solution is resistant to the given attack. If the column is marked with 'X', then the solution is vulnerable to these attacks.

Table 14. Comparison of different solutions security.

	Author proposed solution	RFID tag as authentication token	NFC emulation authentication	Another application for authentication	Biometrics	Out-of-Band authentication	Device generated OTP	SMS OTP	Image based authentication	Password authentication
Physical observation	O	O	O	X	O	O	O	O	X	X
Targeted impersonation	O	O	O	X	O	O	O	O	O	X
Guessing attack	O	O	O	O	O	O	O	O	X	X
Internal observation	O	X	O	X	X	O	X	X	X	X
Leaks from other verifiers	O	O	O	X	X	X	X	X	X	X
Man-in-the-Middle	O	O	O	X	X	X	X	O	X	X

In order to give an appropriate usability comparison of different usability criteria must be made. Usability criteria used for comparison are [36]

- Memorywise effort – Measuring the amount of information the user has to remember
- Scalability place on users – The amount of information needed to remember each different solution using this sort of authentication
- Nothing to carry – The amount of extra devices/features needed from the user in order to authenticate himself/herself
- Physical effortless – How much physical effort the user must make within the authentication flow
- Efficiency – How long does the authentication process take
- Infrequent error – Reliability of the authentication solution
- Easy recovery from loss – Recoverability of the authentication solution, if a user forgets the required passphrases

For the usability comparison, the password authentication solution is taken as reference point and if the performance/usability of the given scheme is better than with the password authentication, then it will be marked with ‘+’ sign. If the performance is the same as with the password solution, it is marked as ‘=’ and if the performance/usability is worse than with the password scheme, then it is marked with ‘-’.

Table 15. Comparison of different solutions usability.

	Author proposed solution	RFID tag as authentication token	NFC emulation authentication	Another application for authentication	Biometrics	Out-of-Band authentication	Device generated OTP	SMS OTP	Image based authentication	Password authentication
Memory wise effort	-	=	=	=	+	=	=	=	-	=
Scalability	+	+	+	+	+	=	=	=	=	=
Nothing to carry	-	-	-	-	-	-	-	-	=	=
Physical effort	-	-	-	-	-	-	-	-	=	=
Efficiency	-	-	-	-	-	-	-	-	-	=
Infrequent error	=	=	=	-	-	-	-	-	-	=
Recoverability	+	-	-	-	-	=	-	-	-	=

As it is visible from the usability comparison in Table 15, security always comes at a price for user experience and usability. The more difficult the authentication solution, the more it requires the user to make the physical effort and memorize different passphrases. The

author's solution is better for recoverability, when the user has forgotten his/hers PIN1, then he/she can recover it using PIN2 and can even do it offline without any outside connection whatsoever. In the case that the user has lost his/hers NFC card, then the recoverability is inconvenient for the user, as he/she needs a new card. When it comes to security, the author's proposed solution has similar security as the NFC emulated card does and is resistant to most attacks. Physical observation is not enough to bypass the security within author proposed security solution, as even if you have the passphrases user has, you need the user's device and the user's NFC card also. Impersonation is impossible, as private certificates are located on top of the NFC card and they are not extractable from it. Guessing attack is also impossible, as NFC PIN1 and PIN2 have only 3 tries before it locks the card. Internal observation would not be successful either, as it is impossible to observe NFC card internally. No other verifiers are user within the author's proposed solution, eliminating all other verification-based security risks. The impossibility of Man-In-The-Middle attack in the context of NFC was explained in section 2.2 of this thesis.



### 3 Proposed NFC Security Solution for Web Applications

The author's proposed NFC security authentication solution extends web application security to a new level, enabling user verification via the NFC Java Card solution. Each user receives their own security card, issued by authorized issuing authorities like the Estonian government. Which the card enables the user to log into different web applications using their personal passcodes.

#### 3.1 Platform Selection

Java Card platform was selected for the NFC smartcards, as it gives the possibility to add custom applications on top of smart cards, that can be accessed via contactless (NFC) or contacted reader interfaces. For the given prototype, a multi interface Java Card solution is selected, which means Java Card has both contactless and contact interface enabled at the same time, giving access to shared memory space on the card, using the same Java Card application. As NFC is an integrate part of the security solution, there is a need for an NFC compatible mobile device. Possible selections are Android OS (operating system), IOS or Windows OS based device. IOS devices do not have proper NFC support therefore we rule them out immediately. Selection between Windows and Android must be made. Based on Q4 2014 data, Android OS holds a market share of 76.6% compared to the 2.8% market share of Windows phone [19]. Due to the huge market gap, an Android OS based phone is selected.

The web application platform can be selected based on the creators liking and the given backend solutions (server solution) can be implemented with all the best-known web application solutions. The only requirement is, that the solution must be compatible with the Google Cloud Messaging service [18], used for Android communication. For the prototype, the author is using Java based web application platform called Grails<sup>5</sup>, an open source free to use framework for Java Virtual Machines. Allowing to build Java web solutions using Groovy, Grails and Hibernate. Grails has the whole Java web service development tools combined together with its integrated ORM<sup>6</sup>, domain specific Languages, meta- and asynchronous programming [17]. Grails has multiple connection adapters for different databases, but for this solution, MySQL<sup>7</sup> database engine is selected to work hand-in-hand with the Java Hibernate<sup>8</sup> database connector. One of the biggest benefits of the Grails web framework is its ability to create and distribute software plugins with ease, enabling the solution to be integrated into other web applications seamlessly.

#### 3.2 Architecture

The proposed NFC security solutions architecture can be divided into 2 separate sections, separated by a secure web layer. One part of the solution is the user side (left hand side of the Figure 4), where the user has his/hers workstation, mobile phone and NFC security card. The other side of the solution (right hand side of the Figure 4.) is the web application server side that handles the data sent to the user and the access given to the user. Between those sides is the vast open web (Internet) with a secure channel connecting both sides of

---

<sup>5</sup> <https://grails.org/>

<sup>6</sup> [http://en.wikipedia.org/wiki/Object-relational\\_mapping](http://en.wikipedia.org/wiki/Object-relational_mapping)

<sup>7</sup> <https://www.mysql.com/>

<sup>8</sup> <http://hibernate.org/>

this solution and a Google Cloud Messaging (GCM) service to handle the NFC authentication start.

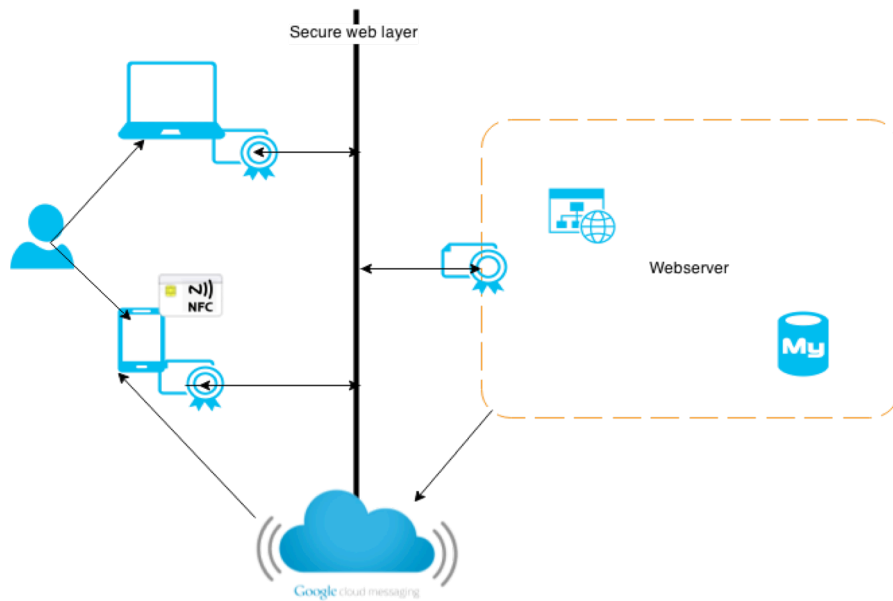


Figure 4. High-level solution architecture.

User side consists of user mobile device (android device in the prototyping phase), computer and NFC card. User operates on computer and is required for additional security authentication. User device receives a authentication required notification via GCM service and user is prompted into authentication application, where user must connect NFC card with the device and enter a PIN for the NFC card. The server side consists of a server solution hosting the web application the user is using and managing security levels together with user NFC card certificates. Server side is connected to GCM service to send notifications to user devices, requiring authentication from user.

### Server

Server architecture contains a local machine running Tomcat 8.0.15 web server and hosting a JVM website within this Tomcat web server. For the database, this solution is using MySQL database. Website is built on Grails Model View Controller (MVC) solution and is using hibernate and MySQL J connector [20] to connect with server MySQL database. The Web Server architecture is described in Figure 5.

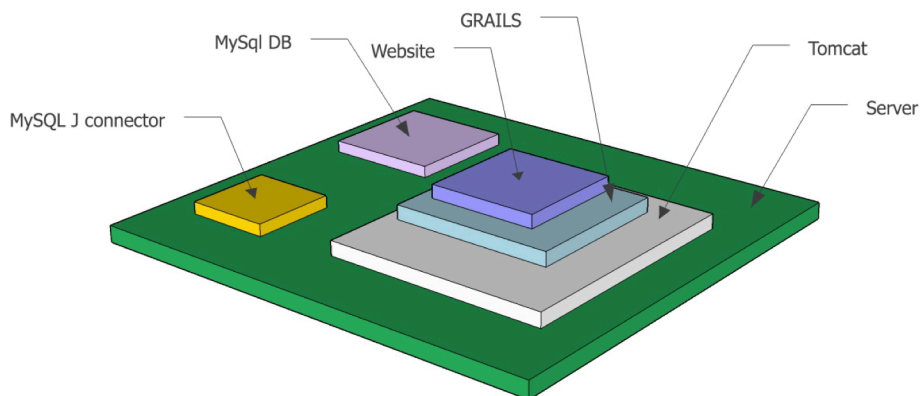


Figure 5. Server architecture.

This application is also integrated with Google Cloud Messaging service (GCM) [18], that has the capability to deliver notifications to client phones, waking them up from sleep and saving the effort to keep customer phone constantly connected to the web server. The GCM connection is established via regular HTTP [21] calls.

### Smartphone Application

The mobile application is based on Android mobile phones with NFC readers. This application is built using Android Software Development Kit (SDK) [22]. The mobile NFC app uses the device embedded NFC reader to access Java Card information and functionality. App also has the capabilities to use Android-embedded Key Store to keep web application certificate for TLS 1.2 communication. Google cloud messaging service is using android notification service, sending notifications to NFC Android application and starting application authentication flow. The Android application architecture is described in Figure 6.

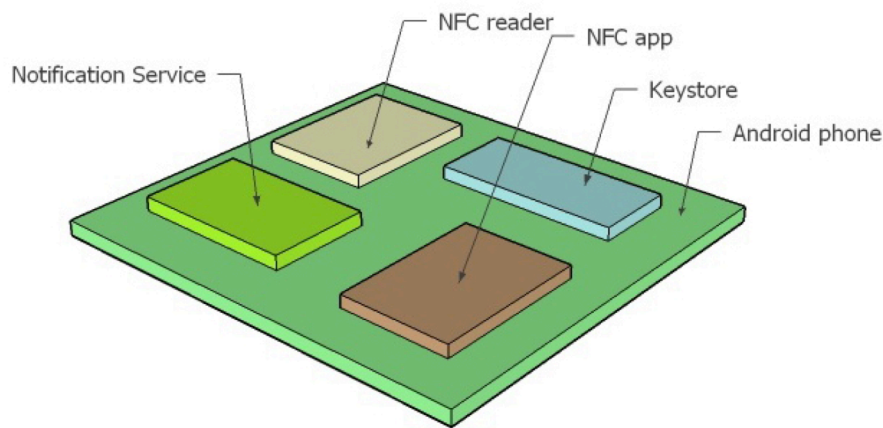


Figure 6. Smartphone application architecture.

Android application uses Android SDK embedded IsoDep<sup>9</sup> communication protocol to communicate with the Java Card application via the NFC reader. IsoDep protocol connects to Java Card and communicates via APDU (Application Protocol Data Unit) calls [11]. Possible APDU calls are described in Appendix 1.

### Java Card

The smart card used in this solution is dual interface Infineon jTop Java Card v3.0 with shared memory. The dual interface allows users to use both contactless and contact smart card interface enabling the use of this card with both NFC and regular smartcard readers. This card has shared memory between two interfaces, giving access to the same Java Card applet via both interfaces. Inside this shared memory is the Java Card applet containing APDU processing solution. Smart card architecture is described in Figure 7.

<sup>9</sup> <http://developer.android.com/reference/android/nfc/tech/IsoDep.html>

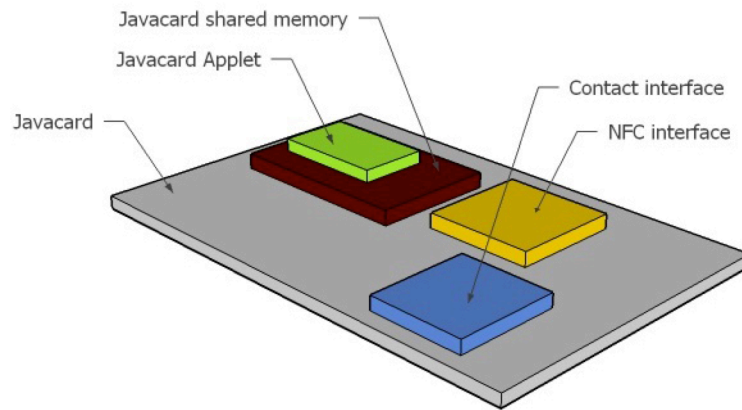


Figure 7. Java Card architecture.

The Java Card application is based on the Estonian ID card, using the same APDU's as the Estonian ID card does [5]. This gives the ability to directly use Estonian NFC ID card for authentication, when it is enabled in the near future. Commands, enabled in the NFC application and Java Card applet, are listed in Appendix 1.

### 3.3 Data Model

Application is using Read-, Write-, Update (RWU) database model, where new entries can only be added, read or updated, but not deleted. Database domain model is described in Figure 8.

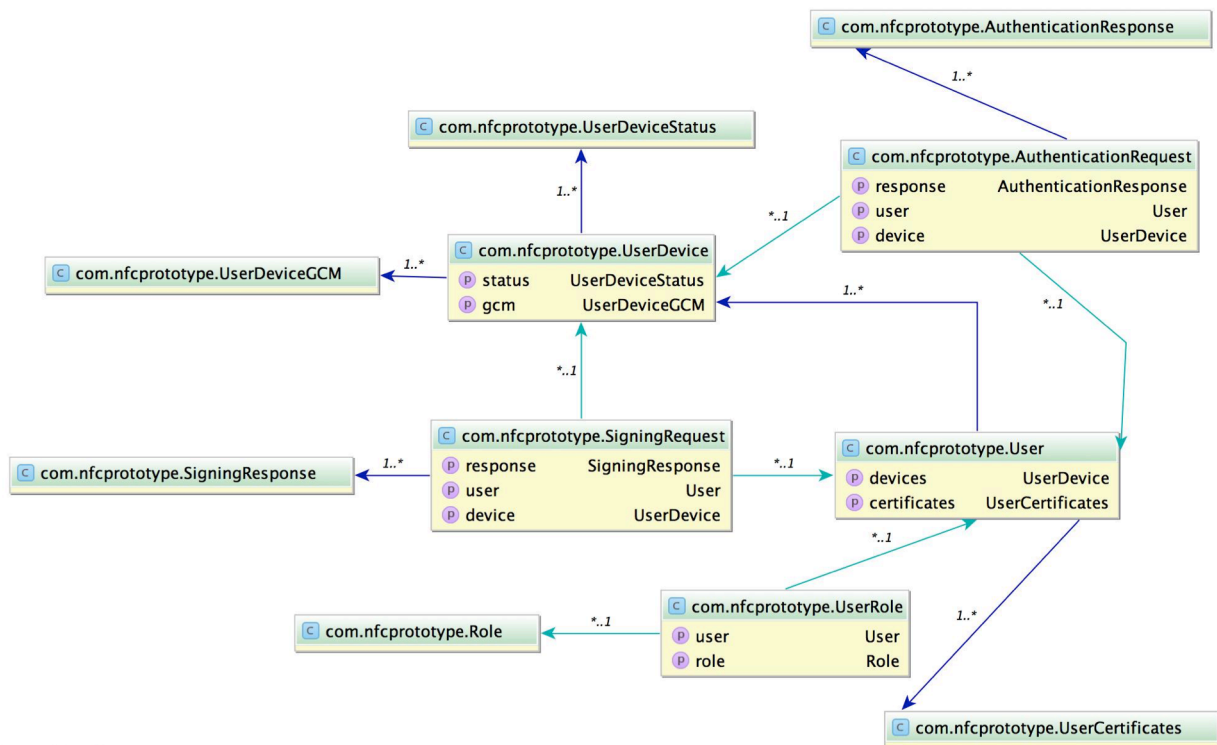


Figure 8. Server application domain model

The data model described in Figure 9 holds user data, user roles, user devices, user device statuses, user device Google Cloud Messaging tokens and user certificates. Regarding authentication and signing, the requests and results of authentication and signing, are also stored in MySQL database for better observation and debugging. In future development, authentication and signing requests and responses can be stored in memory cache based databases and final results can be added to relational database and not overloading the database. The full model of the NFC security prototype solution database is described in Figure 9.

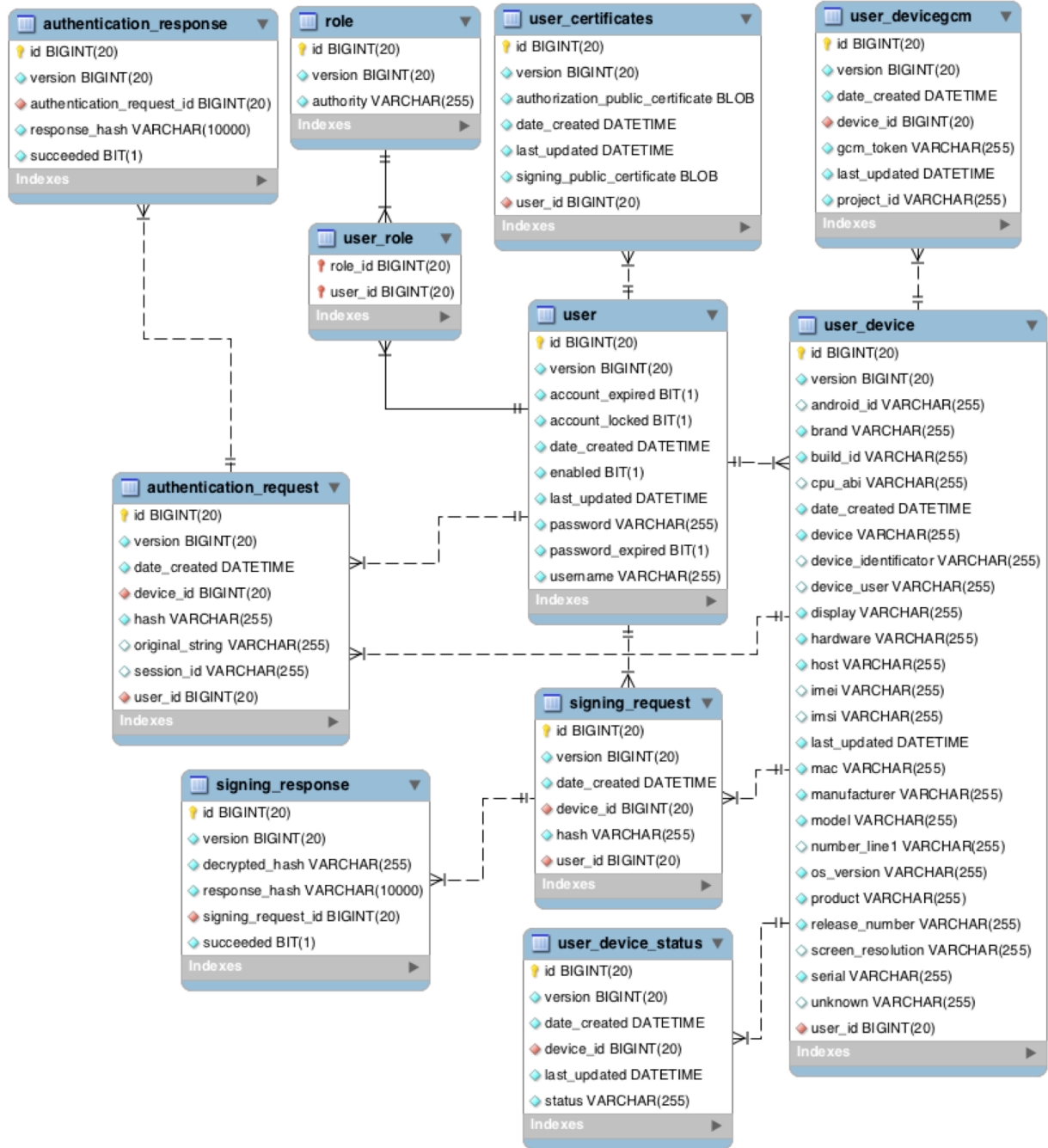


Figure 9. Server application data model

### 3.4 Security

#### Mobile Device Verification

During registration, the web application collects all possible data from the Android build file and sends it to the web application. Based on that data, a unique set of data is selected (data that will not change during device updates etc.) and based on this data, the application calculates user device fingerprint. The fingerprint is calculated based on 4 unique and unchangeable values, using device IMEI<sup>10</sup>, serial, display and MAC<sup>11</sup> values. Fingerprint calculation is based on SHA-512 hash and byte array XOR. Fingerprint is used as Base64 string [31]. Algorithm solution code sample is described below.

```
String getDeviceFingerprintFromDevice(UserDevice device) {
    MessageDigest mda = MessageDigest.getInstance("SHA-512");
    byte[] deviceBytes = mda.digest(device.imei.getBytes());
    byte[] tempBytes = mda.digest(device.mac.getBytes());
    deviceBytes = xorByteArrays(deviceBytes, tempBytes);
    tempBytes = mda.digest(device.screenResolution.getBytes());
    deviceBytes = xorByteArrays(deviceBytes, tempBytes);
    tempBytes = mda.digest(device.serial.getBytes());
    deviceBytes = xorByteArrays(deviceBytes, tempBytes);
    byte[] encodedBytes = Base64.encodeBase64(deviceBytes);
    return new String(encodedBytes);
}
```

This device fingerprint is the unique identification token for the users device during authentication and signing. Each time an authentication response is received, additional device data is received and fingerprint is calculated to make sure, the response is coming from a correct device. If the device fingerprint does not match to the customer allowed device fingerprint, authentication/signing will fail on the server side and the user will not be granted access.

#### Verify Authentication

In the prototyping phase, the authentication verification works similar to the signature verification of the user (described in section 3.4.3). User authenticates himself/herself via NFC card by simply signing a pre-generated hash with authentication certificate located on the NFC card. Public authentication certificate can also be extracted from the NFC card with simple APDU calls (described in Appendix 1). Private key of authentication certificate is pre-personalized into NFC card and can only be changed with master key that is kept secret from users. Therefore in order to authenticate, user must remember the PIN1 passphrase verified by the NFC card and after verification (security level changed in NFC card), user can encrypt web application generated hash with the authentication private key using RSA encryption algorithm (described in section 2.3.1 of this thesis). After encryption, the web application verifies the encryption with user public authentication certificate (pre-entered into web application) and if there is a match, the user gets authenticated into web application.

```
UserCertificates certs = authenticationReq.user.certificates.first();
CertificateFactory certFactory = CertificateFactory.getInstance("X.509");
InputStream inps = new
ByteArrayInputStream(certs.authorizationPublicKey());
X509Certificate cert = (X509Certificate) certFactory.generateCertificate(inps);
```

<sup>10</sup> <http://www.imei.info/>

<sup>11</sup> [http://en.wikipedia.org/wiki/MAC\\_address](http://en.wikipedia.org/wiki/MAC_address)

```

Security.addProvider(new BouncyCastleProvider());
Cipher asymmetricCipher = Cipher.getInstance("RSA/ECB/PKCS1Padding", "BC");
asymmetricCipher.init(Cipher.DECRYPT_MODE, cert.getPublicKey());
byte[] decrypted = asymmetricCipher.doFinal(encryptedHashBytesArr);
if(originalHashByteArray.encodeAsHex() == decrypted.encodeAsHex()) {
    return true;
}
return false;

```

For authentication, Grails uses Java security libraries and embedded BouncyCastle<sup>12</sup> security provider, RSA algorithm with ECB (Electronic codebook) cipher block mode [30] and PKCS1Padding for encrypted data padding. First a user authorization public certificate byte array is received from the database and generated into X509Certificate Java class, implementing certificate functionality. A security provider RSA decrypt is executed with user public key taken from the certificate and decrypted byte array is received. In order to do fast comparison between original hash byte array and decrypted hash byte array, the arrays are encoded to HEX<sup>13</sup> string and then compared. If the original hash HEX string matches the decrypted hash HEX string, the authentication has been successful. Depending on whether the authentication has been successful or not, a Boolean value “false” or “true” is returned accordingly.

### Verify Signature

In order to sign an action, task or a dataset made in the web application, a hash must be generated from the data that is about to be signed and a signing request must be sent to the mobile application. Mobile application authenticated the user with a PIN2 passphrase and by doing that the security level of the Java Card applet is changed to signing. After successful user verification, hash received from web application is signed using users signing private key stored in NFC card memory (APDU calls described in Appendix 1). After the hash has been signed, it is returned to the web application, where the signed hash is verified using users public signing certificate, pre-entered into the web application. If the signatures match, the user’s signing process was successful. Signature verification is handled with RSA public and private key cryptography (described in section 2.3.1 of this thesis). Signature verification is handled using the Java security implementation, included in the Java Development Kit.

```

UserCertificates certs = authenticationReq.user.certificates.first();
CertificateFactory certFactory = CertificateFactory.getInstance("X.509");
InputStream inps = new ByteArrayInputStream(certs.signingPublicCertificate);
X509Certificate cert = (X509Certificate) certFactory.generateCertificate(inps);
Security.addProvider(new BouncyCastleProvider());
Cipher asymmetricCipher = Cipher.getInstance("RSA/ECB/PKCS1Padding", "BC");
asymmetricCipher.init(Cipher.DECRYPT_MODE, cert.getPublicKey());
byte[] decrypted = asymmetricCipher.doFinal(encryptedHashBytesArr);
if(originalHashByteArray.encodeAsHex() == decrypted.encodeAsHex()) {
    return true;
}
return false;

```

For signing, Grails uses Java security libraries and embedded BouncyCastle security provider, RSA algorithm with ECB (Electronic codebook) cipher block mode [30] and PKCS1Padding for encrypted data padding. First a user signing public certificate byte array is requested from database and generated into X509Certificate Java class, implement-

<sup>12</sup> <https://www.bouncycastle.org/>

<sup>13</sup> <http://en.wikipedia.org/wiki/Hex>

ing certificate functionality. A security provider RSA decrypt is executed with the user's public key taken from certificate and decrypted byte array is received. In order to do fast comparison between original hash byte array and decrypted hash byte array, the arrays are encoded to HEX string and then compared. If the original hash HEX string matches the decrypted hash HEX string, the authentication has been successful. Depending on whether the authentication has been successful or not a Boolean value false or true is returned accordingly.

### Security Between NFC Card and Reader

Security between the NFC card and the NFC card reader (embedded into mobile device) is not secured in the first prototype iteration. During prototype phase, we are relying on NFC channel security and the fact that this solution is using one active and one passive device for NFC communication, therefore the NFC attack area is quite small and the attack can affect only one person personally, not the whole user base. As described in section 2.2 of this thesis, we can assume that NFC is open to neither Man-In-The-Middle nor eavesdropping attacks, therefore the passphrases are not easy to obtain. The most likely type of attack is data corruption that only prevents user from logging into desired web application, as data is corrupted during transmission between Java Card and NFC reader. It is possible to use a shared secret encryption on NFC communication further described in section 2.3 of this thesis this would make the NFC connection secure and resilient to all possible passphrase attacks.

### Security Between Device and Web Server

The communication between the device and the application server is secured with an HTTPS<sup>14</sup> connection using TLS 1.2 and a one-way certificate authentication [23]. Server is using a trusted certificate authority issued certificate and Android mobile device has this server certificate authority added to trusted CA<sup>15</sup> list in Android KeyStore<sup>16</sup>. TLS secure connection flow is described in Figure 10.

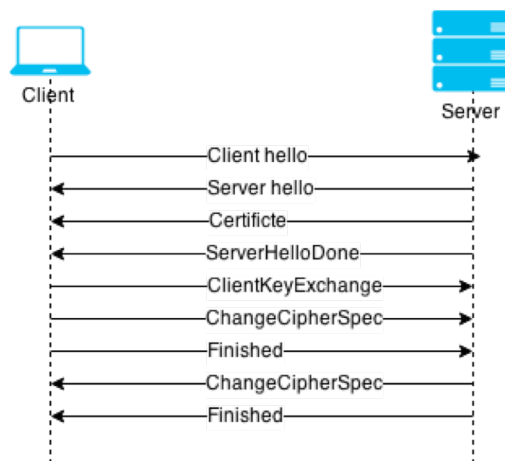


Figure 10. TLS handshake model

During the TLS connection establishing, server exchanges public certificate to client. Doing so, the client can verify that the connecting party is who he claims to be by verifying

<sup>14</sup> <http://en.wikipedia.org/wiki/HTTPS>

<sup>15</sup> [http://en.wikipedia.org/wiki/Certificate\\_authority](http://en.wikipedia.org/wiki/Certificate_authority)

<sup>16</sup> <http://developer.android.com/reference/java/security/KeyStore.html>



encrypted byte arrays with public key received from certificate. Both sides must have knowledge of trusted certificates and allowed connections. After the client verifies the server and keys have been exchanged, a unique session key is generated and shared. All future communication is encrypted with this key.

### **3.5 Application Flow**

Given NFC security solution has multiple flows that need to be handled separately. To start with, the user first needs to have an account in given web application - a prerequisite to start NFC authentication flow. Web application must have prior knowledge of users authentication and signing certificate public keys and need to know the username and passphrase of the user. If the prerequisites are handled, the user can enter the NFC security authentication application and it's flows. To start with, user must authenticate the device with given web application. To do so, the user must pass the Authorize device phase described in section 3.5.1 in this thesis. After successfully passing device authorization, user can then authenticate himself/herself with the mobile application or sign web application actions using given mobile applications. Authentication- and Signing phase of the NFC security solution are described in section 3.5.2 and 3.5.3 of this thesis. Final phase that is directly related to using proposed NFC solution mobile application and NFC card is unauthorize device phase, where the user unauthorizes device from web application.

#### **Authorise Device Flow**

In order to authorize the device, the user's device must first obtain a GCM device key and store it in device application local storage. This key is received on the first startup of the mobile application and is removed together with application uninstall. A new key is obtained after application reinstall. After the mobile application has received the GCM key, a device verification call is made to web application in order to verify (described in Appendix 4), if device is already activated by some user or not. If the device is not registered by any user - user can authorize this device as his/her authentication endpoint.

In order to authorize a device the user must enter their username and password, within the main view of the android application (described in Appendix 3), to authenticate himself/herself and register a new mobile device to the web application as an authentication device. After user has finished entering the username and passphrase, the application generates SHA-512 hash from user entered password and encodes it into Base64 string. After the hashing has finished, username, password hash Base64 string, GCM key and device data is sent to web application (API call described in Appendix 4). Full authentication flow is described in Figure 11.

If device authorization has been successful, device is added to users allowed device list and user can now use this device to authenticate himself/herself within the web application. If user has used this device before, then already existing device is reactivated as a valid authentication endpoint for the user. One device can be linked with only one user at a time. In order to use this device as an authentication point for another user, the user must first unauthorize the device (described in section 3.5 of this thesis) and the reauthorize the device as their device.

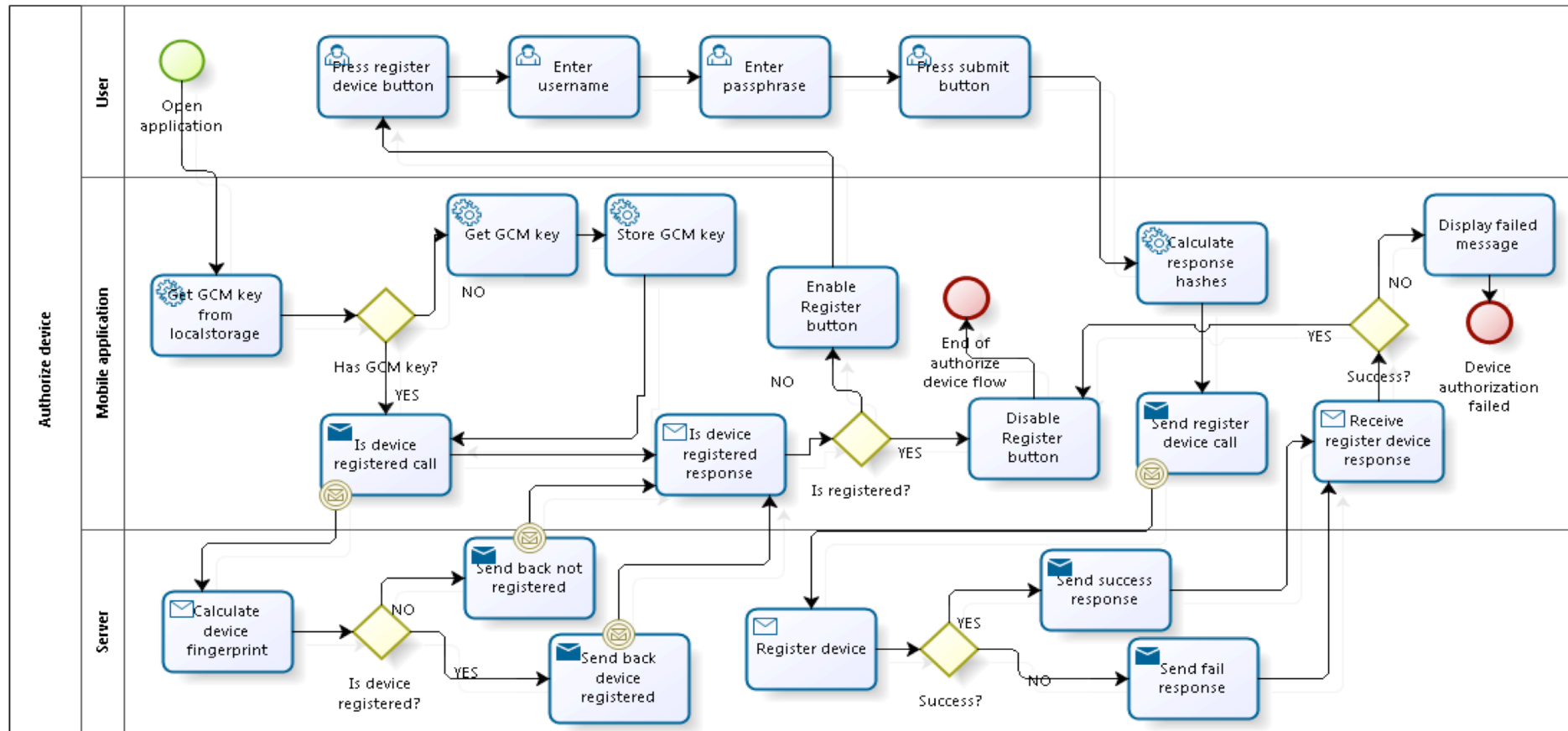


Figure 11. Device authorisation flow

## Authentication flow

To authenticate the user using the proposed NFC authentication solution, user must first try to login to web application. Upon arrival to web application user gets redirected to login page, where user must enter his/hers username and passphrase. After the sign-in button has been pressed, the web application first validates user username and password and if they are correct, it displays user the list of authenticated devices that can be used for NFC second layer authentication. At this point user must select a proper device – a device that he/she has switched on and connected to the network. After user has selected a device used for authentication, server generates a random SHA-512 hash and sends this hash to the selected mobile device via GCM service and waits for an answer from the device (maximum wait time is configurable, but is 120 seconds for prototype). At this point the authentication responsibility is delegated to the mobile device and the NFC Java Card.

When the mobile device receives an authentication request from server, the request is delivered to proper mobile application (NFC authentication application in this case) using `BroadcastReceiver`<sup>17</sup> and notification intents. A new Intent launches the NFC authentication application with authentication dialog box. While this dialog is open, user must pair his/hers device with NFC card – pairing is successful, when PIN1 retry counter is displayed to user. After successful pairing, user must enter PIN1 (distributed together with user personalized Java Cards) in order to log into web application. After user has entered PIN1 into dialog screen, authentication is handed over to Java Card application.

Mobile application uses an NFC card and authentication APDUs (described in Appendix 1) to encrypt the hash received from server with user authentication private key located inside NFC card. After hash signing is successful, mobile device sends back signed hash and original hash to server (using API call described in Appendix 4). The server validates the received hash against the users authentication public key (described in section 3.4 of this thesis) and if they match, then user gets authenticated and logged into web application. Detailed user authentication flow is described in Figure 12.

---

<sup>17</sup> <http://developer.android.com/reference/android/content/BroadcastReceiver.html>

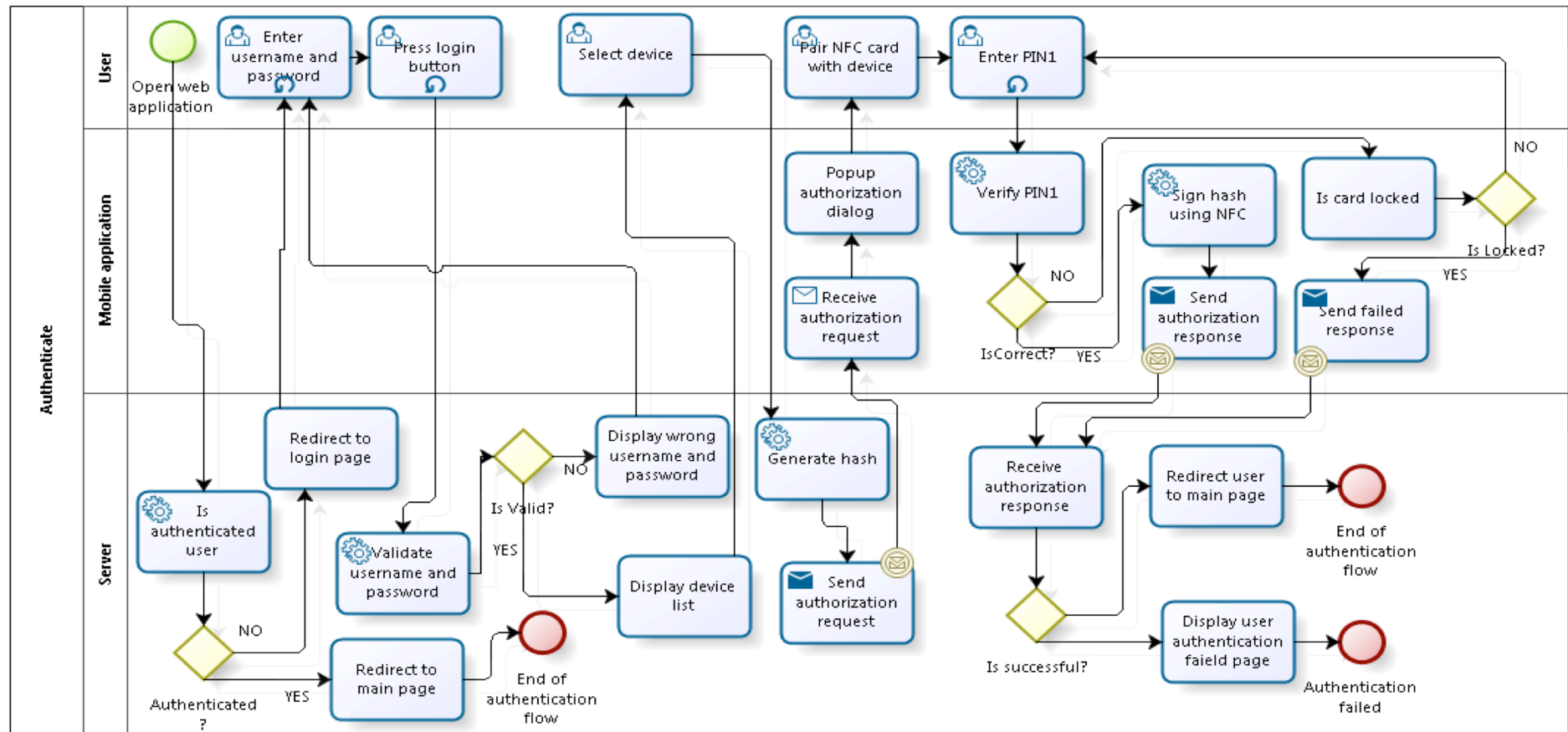


Figure 12. Authentication flow

## Signing flow

Signing flow is used to add an additional security verification to user tasks or activities within web application. Signing is used to maintain the integrity of the data (signed data can be validated, if someone has changed it or not) and to add an electronically proven signature, that this data has been authorized and validated by given user. This enables the web application to require signing from users to ensure the user is aware of the action and to tie this action with a certain user. Signing is relying on the fact, that the NFC card is accessible only for this given user and therefore this signature has the same effect as a written signature.

In order to start signing process, web application must require authorized user to select a signing device from the list of authorized user devices or use a device that was used to authenticate the given session he/she is using. When user has selected a device / web application has identified the authentication device used previously to authenticate the user, web application must calculate signing hash over all the data about to be signed. NFC prototype is using SHA-1 hash and hash byte array XOR to generate hash over all the fields if there is more than one field to be signed. In case there is only one field of data to be signed, web application will use SHA1 hash of this field as signing hash.

After general data signature hash has been calculated, web application sends hash together with signing request to user device via GCM service. User device receives a signing request via BroadcastReceiver and creates a new Intent launching the mobile application and shows a signing dialog to the user. While this dialog is open, user must pair his/hers device with NFC card pairing is successful, when PIN2 retry counter is displayed to user. After successful pairing, user must enter PIN2 of NFC authentication in order to sign the received data. After user has entered PIN2 into dialog screen, mobile application gives the signing over to NFC Java Card.

NFC card uses APDU's (described in Appendix 1.) to sign the hash received from server with user signing private key located inside NFC card. After hash signing is successful, mobile device sends signed hash and original hash back to server (using API call described in Appendix 4.), where server validates the received hash against user signing public key (described in section 3.4 of this thesis) and if they match, signing has been successful and signature is added to data/task. Signing flow is described in Figure 13.

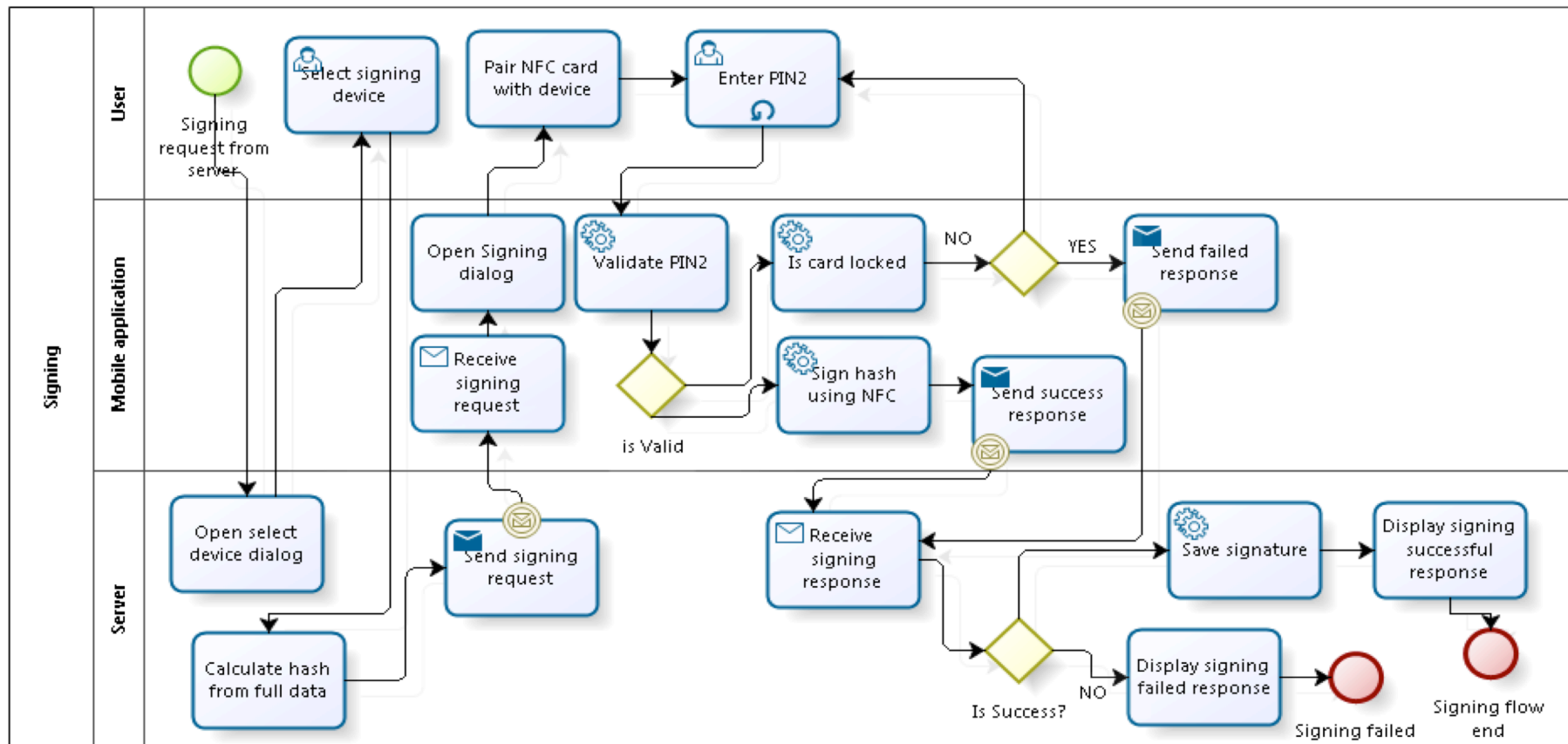


Figure 13. Signing flow

## **Unauthorize Device Flow**

Unauthorization of user device enables user to remove certain mobile device from allowed authorized device list by simply using NFC security mobile application. If user has unauthorized his/her device, they can later authorize it again and start using this device as an authorization endpoint for NFC security solutions.

In order to unauthorized a user's device, the device must first be authorized within the web application. This is verified via a device verification call (API call described in Appendix 4) and if the device is authorized to some user, username of this user is displayed within mobile application (described in Appendix 3). If user device is authorized and in order to unauthorize it, user must open mobile application and press unauthorize device button and confirm the unauthorization (described in Appendix 3). After the button click, mobile application makes an HTTPS connection to web application and sends a device unauthorization call, containing device data and GCM token (API call described in Appendix 4).

After user has unauthorized the device, this device can no longer be used for authentication or signing and it is not displayed as a login option for user. Device unauthorization flow is described in Figure 14.

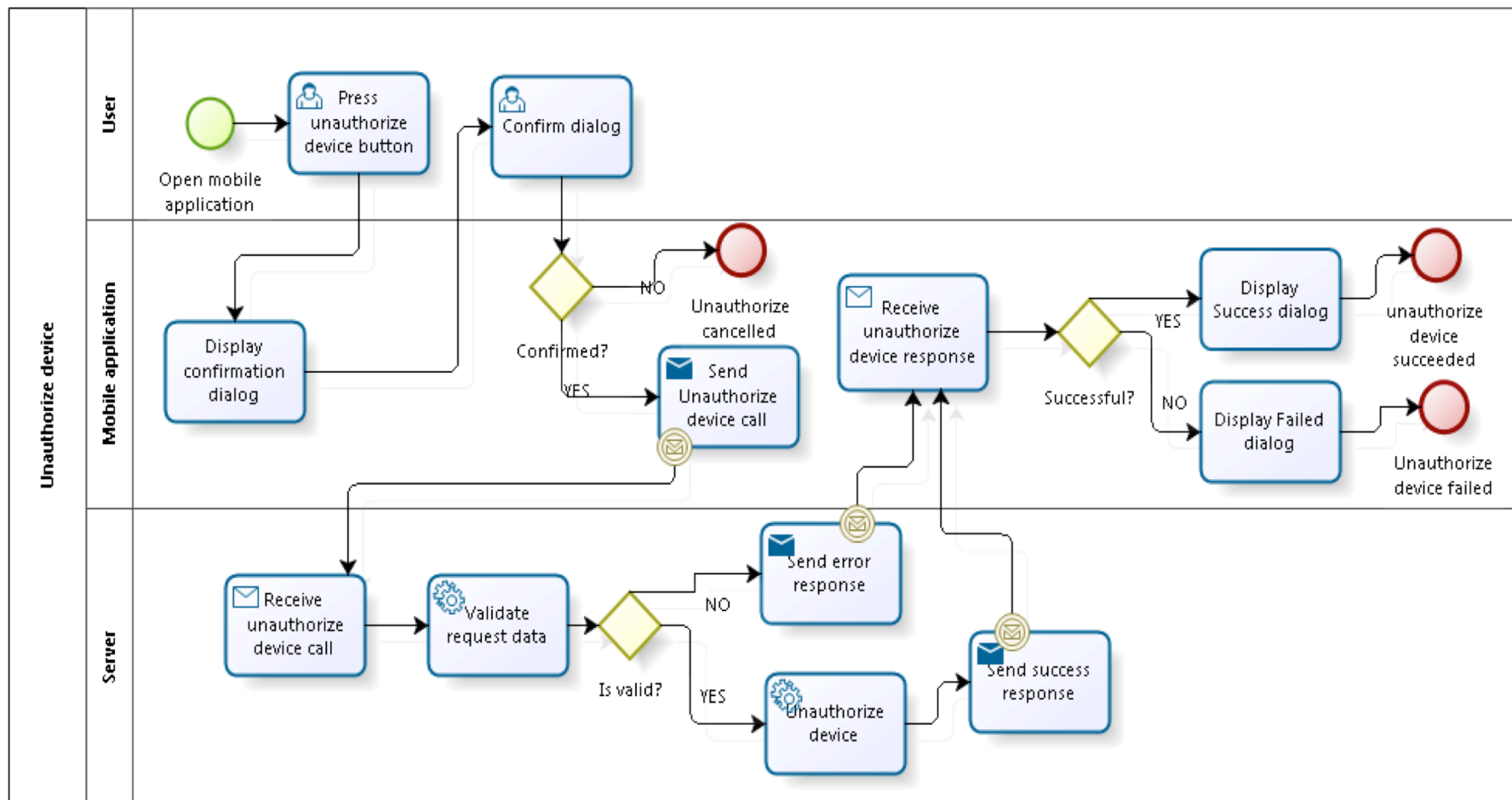


Figure 14. Unauthorize device flow



### 3.6 Future Opportunities

Author proposed NFC security solution for web application could be applied to solve multiple different authentication problems. Same solution, but with some additional development, can be easily converted into Mobile TFA, NO passwords solution, NFC card reader etc. All these solutions are enabled by the complex nature of the dual interface Java Card, enabling both contacted and contactless connection with same Java Card applet. Combining NFC card with Estonian ID card would open yet another possible future development direction, by enabling EstEID communication via NFC card reader and mobile device. This would allow full EstEID capabilities in mobile device without any extra ID card reader necessary. Possibilities for future development seem endless, therefore there is a need to narrow down the possibilities and thoroughly examine the more beneficial solutions.

#### Mobile Two-Factor-Authentication

As mobile is becoming a more and more independent device, enabling user to manage all necessary task using only his/her mobile device, the need for mobile security is on an uprise. Mobile device is an excellent second point of authentication device for TFA, but as mobile is turning into the main device, there is a need for another authentication point to rely on - to provide TFA on mobiles. Here is where NFC security offers an excellent solution with no additional development. Proposed NFC solution enables very hard to break two factor authentication for mobile devices, by using NFC smartcard as second authentication device for users. Mobile phones have direct communication capabilities (NFC reader integrated into most smartphones since 2006) [24] with NFC smartcards, no additional hardware or development is needed. Mobile application simply connects to NFC card, whenever a TFA authentication is needed and authenticates the user via an NFC card similar to the proposed NFC security solution for web application. Mobile TFA flow is described in Figure 15. where step 1 starts with authentication to mobile application, after what, in step 2 NFC PIN1 authentication request is generated to NFC authentication Android app (or NFC authentication embedded into an authorized application). User authenticates himself/herself with NFC card integrated PIN1 passphrase and system verifies authentication similar to section 3.4 of this thesis.



Figure 15. TFA from mobile devices

Proposed solution for web applications can be improved and converted into mobile TFA with ease. As mobile applications communicate with server via API calls, authentication can be provided for mobile application instead of web application.

### **Corporate Security**

Corporations could create their own NFC security cards that would enable users to access through different NFC enabled doors, but also use this card as a main authentication token for company systems, by forcing user to authenticate themselves via PIN codes and NFC cards for all corporate systems. As NFC cards hold user certificates, these certificates can be used for multiple purposes (VPN, Authentication etc.). Additional authentication can be requested via signing request, if user wants to access higher security areas or accept some tasks that need additional verification. All user decisions and actions within corporate system can be signed using an NFC card and user mobile device.

### **No Passwords Solution**

User can have multiple accesses to different solution by using only one personalized NFC card that has been issued to him/her. All solutions can be combined into using the same NFC authentication solution and the user does not need to remember all different login usernames and passwords. User can simply authenticate himself/herself via single PIN stored in NFC card. Creating a new system requiring a user authentication is also easier, as during development, developers can use pre made solutions to integrate with NFC security solution for web applications, removing the need to handle usernames and password within the system.

### **EstEID NFC Mobile Reader**

Proposed solution can be converted into EstEID NFC authentication solution, by simply enabling NFC on Estonian ID card. This would give users the capability of using their personal ID cards as additional authentication endpoints for mobile applications together with web applications and it also enables the use of mobile device as an ID card reader, disabling the need for separate ID card reader. It also adds an additional security layer on top of EstEID authentication, as mobile device as a reader, can also be identified and traced during authentication. It is possible to enable only some user devices to be allowed as NFC card readers. By doing that, the user has full control over what devices can and cannot be used to authenticate via NFC. EstEID can also add a possibility for the user to decide if NFC authentication is allowed or not with certain ID card, therefore disabling NFC attacks while NFC authentication is not used as a means of authentication. Also the list of APDU's enabled (APDU's described in appendix 1) via NFC interface can be limited to only the needed APDU's for authentication. PIN change and more secure APDU's can be enabled for contact interface only. By using these precautions, it is safe to say, that Estonian ID card with an NFC interface would be secure and safe to use as a main identification for Estonian citizens and e-residents.

This proposed solution would also change the authentication flow of the proposed prototype, by enabling TLS connection setup via NFC card (similar to working Estonian ID solution). Authentication would simply start a TLS connection where client certificate is needed and when client certificate request is received from the web application, client computer (used to access web application) picks up the request and forwards it to the user-selected mobile device. Mobile device then acts as an Estonian ID card reader and enables all ID card actions via NFC. Behind the scenes, mobile application sends back user public authentication certificate and signed TLS handshake hash to users computer and then user

browser will get full TLS authentication. Only restriction to this solution is, that user mobile device and user computer must be in the same private network (Wi-Fi, Ethernet or hotspot created with mobile device)

## 4 Conclusions

In conclusion, NFC security has proven to be secure enough to handle user authentication and signing via mobile phone and an internet connection. NFC can also be made more secure, by using a secure channel communication within the NFC communication relying on a shared secret to encrypt the communication data moving between the two devices. As author proposed solution is using one active device (mobile phone) and one passive device (dual interface NFC Java Card), the possible attack vector of NFC connection is reduced to eavesdropping and data corruption - the communication can be eavesdropped or disrupted within a small distance of the card itself. Adding additional APDU whitelisting to the Java Card APDU adds another security feature, enabling administrative actions of the card to be managed only via contacted interface. By enabling a secure channel between device and NFC card, plus adding whitelist to the Java Card applet, it is safe to claim that the proposed NFC solution device to Java Card communication is completely secure. Communication between mobile device and web application is ensured by TLS 1.2 cryptographic protocol and must also be considered as a secure channel between both endpoints. Taking into account all the security features added to the solution, the security of the solution is proven to be highly secured.

The author's proposed solution relies on the analysis made within this thesis to claim that NFC authentication solution for web applications can be created securely and can be used to authenticate user within web application, or sign user's activities using mobile device NFC. Prototype has proven the concept of how the solution should work and enables successful authentication and signing within created web application. Prototype has both dual password solution, where user authenticated himself/herself first with web application passcode and then via proposed NFC solution to get authenticated into web application and no passwords solution, where user simply enters their username and selects a device to authenticate himself/herself with. Both solutions work as described and prove the validity of the proposed solution. Author proposed NFC authentication has huge potential to become a two-factor authentication endpoint for mobile only solutions and has the possibility to use NFC enabled Estonian ID cards as authentication devices. Additionally, author proposed solution adds another level of security to already existing pile of security solutions, enabling new solutions to be built on top of the given solution.

## 5 References

- [1] H. O. Alanazi, B. B. Zaidan, A. A. Zaidan, H. A. Jalab, M. Shabbir, Y. Al-Nabhani, "New Comparative Study Between DES, 3DES and AES within Nine Factors," *Journal of computing*, vol. 2, no. 3, pp 152-157, 2010
- [2] AS Sertifitseerimiskeskus, (2014) "Architecture of ID-software, Version 0.4, ID-software version: 3.9," 28.04.2015 [Online] <http://open-eid.github.io/>
- [3] B. Chess and B. Arkin, "The Case for Mobile Two-Factor Authentication. Security & Privacy", *IEEE* vol. 9, no. 5 ), pp 81-85, 2011
- [4] D. S. A. Elminaam, H. M. A. Kader, M. M. Hadhoud, "Evaluating The Performance of Symmetric Encryption Algorithms," *International Journal of Network Security*, vol. 10, no. 3, pp 213-219, 2010
- [5] AS Sertifitseerimiskeskus, (2013) "EstEID v. 3.5 Estonian Electronic ID-card application specification," 28.03.2015 [Online] [http://id.ee/public/TB-SPEC-EstEID-Chip-App-v3\\_5-20140327.pdf](http://id.ee/public/TB-SPEC-EstEID-Chip-App-v3_5-20140327.pdf)
- [6] E. Haselsteiner and K. Breitfuß, "Security in Near Field Communication (NFC): Strengths and Weaknesses," *Workshop on RFID security*, pp 1-10, 2006
- [7] S. Hallsteinsen, I. Jørstad, D. V. Thanh, "Using the mobile phone as a security token for unified authentication." *Systems and Networks Communications*, pp 1-6, 2007
- [8] H. A. Housani, J. Baek, C. Y. Yeun, "Survey on Certificateless Public Key Cryptography," *ICITST Internet Technology and Secured Transactions*, pp 53 - 58, 2011
- [9] X. Huang, Y. Xiang, A. Chonka, J. Zhou, R. H. Deng, "A Generic Framework for Three-Factor Authentication: Preserving Security and Privacy in Distributed Systems," *IEEE Transactions On Parallel and Distributed Systems*, vol. 22, no. 8, pp 1390-1397, 2011
- [10] N. Jansma, B. Arrendondo, "Performance Comparison of Elliptic Curve and RSA Digital Signatures," *Technical Report. University of Michigan: Ann Arbor*, pp 1-11, 2004
- [11] Sun Microsystems Inc, (2011) "Java Card Platform Specification 3.0.4," 05.04.2015 [Online] <http://www.oracle.com/technetwork/java/javacard/specs-jsp-136430.html>
- [12] H. Ko, J. Kim, J. Jung, Y. Lee, S. Joe, Y. Chang, "A Study on the 128 bits SEED algorithm to apply in RFID tag," *Convergence Information Technology*, pp 406-411, 2007
- [13] T. Nie and T. Zhang, "A Study of DES and Blowfish Encryption Algorithm," *Tencon 2009 - 2009 IEEE Region 10 Conference*, pp 1-4, 2009
- [14] R. S. Pippal, C. D. Jaidhar, S. Tapaswi, "Highly Secured Remote User Authentication Scheme using Smart Cards," *Industrial Electronics and Applications (ICIEA)*, pp 1001 - 1005, 2012
- [15] D. Rinner, H. Witschnig, E. Merlin, "Broadband NFC - A System Analysis for the Uplink," *Information Forensics and Security*, pp 292-296, 2009
- [16] D. Stuttard and M. Pinto, "The Web Application Hacker's Handbook: Discovering and Exploiting Security Flaws," *Wiley Publishing Inc.*, pp 1-593, 2008
- [17] G. Rocher, P. Ledbrook, M. Palmer, J. Brown, L. Daley, B. Beckwith, L. Hotari, "The Grails Framework - Reference Documentation ver. 2.4.4," 18.04.2015 [Online] <http://grails.github.io/grails-doc/2.4.4/>
- [18] Google Inc., "Google Cloud Messaging for Android," 18.04.2015 [Online] <https://developer.android.com/google/gcm/index.html>

- [19] IDC Corporate USA, "Smartphone OS Market Share, Q4 2014," 18.04.2015 [Online] <http://www.idc.com/prodserv/smartphone-os-market-share.jsp>
- [20] Oracle Corporation, "MySQL Connector/J Developer Guide," 18.04.2015 [Online] <http://dev.mysql.com/doc/connector-j/en/index.html>
- [21] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, (1999) "Hypertext Transfer Protocol -- HTTP/1.1," pp 1-176, 18.04.2015, [Online] <http://www.hjp.at/doc/rfc/rfc2616.html>
- [22] J. Steele and N. To, "The Android Developer's Cookbook: Building Applications with the Android SDK" *Pearson Education*, pp 1-400, 2010
- [23] T. Dierks and E. Rescorla, (2008) "The Transport Layer Security (TLS) Protocol Version 1.2" 18.04.2015 [Online] [http://tools.ietf.org/html/rfc5246?as\\_url\\_id=AAAAAAVBehpzRqATU5xWpMSTPjTY4oV6aOnai43OyHdsdcjqdSIYu0y-i\\_wtuyMcDhdfR\\_le\\_fBCnWW1xu50YwXZ7oot](http://tools.ietf.org/html/rfc5246?as_url_id=AAAAAAVBehpzRqATU5xWpMSTPjTY4oV6aOnai43OyHdsdcjqdSIYu0y-i_wtuyMcDhdfR_le_fBCnWW1xu50YwXZ7oot)
- [24] NearFieldCommunication.org "Development of NFC Compatible smartphones," 25.04.2015 [Online] <http://www.nearfieldcommunication.org/smartphone-development.html>
- [25] C. Lu, A. L. M. Santos, F. R. Pimentel, "Implementation of Fast RSA Key Generation on Smart Cards" *Proceedings of the 2002 ACM symposium on Applied computing*, pp 214-220, 2002
- [26] R. Rivest, A. Shamir, L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems" *Communications of the ACM* 21, pp 120-126, 1978
- [27] N. Koblitz, A. Menezes, S. Vanstone, "The State of Elliptic Curve Cryptography," *Kluwer Academic Publishers*, pp 173-193, 2000
- [28] M. Mostafa and A. Allah, "Strengths and Weaknesses of Near Field Communication (NFC) Technology" *Global Journal of Computer Science and Technology*, vol. 11, issue. 3 ver. 1.0, pp 1-6, 2011
- [29] A. Paus, "Near Field Communication in Cell Phones," *Seminararbeit Ruhr-Universität Bochum*, pp 1-19, 2007
- [30] A. J. Menezes, P. C. Oorschot, S. A. Vanstone, "Handbook of Applied Cryptography," *CRC Press*, pp 816, 1996
- [31] S. Josefsson, (2006) "The Base16, Base32, and Base64 Data Encodings," 08.05.2015 [Online] <https://tools.ietf.org/html/rfc4648>
- [32] A. Sethi, O. Manzoor, T. Sethi, "User Authentication on Mobile Devices," 10.05.2015, [Online] <http://www.cigital.com/wp-content/uploads/downloads/2012/11/mobile-authentication.pdf>
- [33] R. B. Davies, "Exclusive OR (XOR) and hardware random number generators," 10.05.2015, [Online] <http://www.robertnz.net/pdf/xor2.pdf>
- [34] Google Inc. "Using OAuth 2.0 to Access Google APIs," 10.05.2015, [Online] <https://developers.google.com/identity/protocols/OAuth2>
- [35] Sertifitseerimiskeskus AS, "Overview of the EstEID certification hierarchy," 10.05.2015 [Online] <http://id.ee/index.php?id=35774>
- [36] H. Lee, W. Hong, C. Kao, C. Cheng, "A user-friendly Authentication Solution using NFC Card Emulation on Android," *2014 IEEE 7th International Conference on Service-Oriented Computing and Applications*, pp 271-278, 2014
- [37] M. Massoth and T. Bingel, "Performance of different mobile payment service concepts compared with a NFC-based solution," *2009 ICIW Fourth International Conference on Internet and Web Applications and Services*, pp 205-210, 2009

- [38] C. I. Fan and Y. H. Lin, "Provably Secure Remote Truly Three-Factor Authentication Scheme With Privacy Protection on Biometrics," *IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY*, VOL. 4, NO. 4, pp 933-945, 2013
- [39] H. Fujii and Y. Tsuruoka, "SV-2FA: Two-Factor User Authentication with SMS and Voiceprint Challenge Response," *ICITST -2013*, pp 283-287, 2013
- [40] M. Sreelatha, M. Shashi, M. R. Teja, M. Rajashekar, K. Sasank, "Intrusion Prevention by Image Based Authentication Techniques," *ICRTIT – 2011*, pp 1239 – 1244, 2011
- [41] R. Foster, "Manchester encoding: opposing definitions resolved," *Engineering Science and Education Journal*, Vol. 9, Issue. 6, pp 278-280, 2000
- [42] V. Lalitha and S. Kathiravan, "A Review of Manchester, Miller, and FM0 Encoding Techniques," *Smart Computing Review*, vol. 4, no. 6, pp 481-490, 2014

## Appendix

### I. Java Card Application APDU's

**Abstract:** Appendix 1 describes APDUs sent via IsoDep protocol within mobile application in order to provide authentication functionality to web application. This APDU list is based on Estonian ID card possible commands list and provides exactly the same functionality as EstEID card does via contacted interface.

#### 1. Read PIN retry counters

##### 1.1. Select Master File directory

CLA	INS	P1	P2	Le
00	A4	00	0C	00

##### 1.2. Select counter file

CLA	INS	P1	P2	Lc	Data
00	A4	02	0C	02	0016

##### 1.3. Read counters

###### PIN1 counter

CLA	INS	P1	P2	Le
00	B2	01	04	00

###### PIN2 counter

CLA	INS	P1	P2	Le
00	B2	02	04	00

###### PUK counter

CLA	INS	P1	P2	Le
00	B2	03	04	00

#### 2. Read certificates

##### 2.1. Select Master File directory

CLA	INS	P1	P2	Le
00	A4	00	0C	00



## 2.2. Select EEEE file

CLA	INS	P1	P2	Lc	Data
00	A4	01	04	02	EEEE

## 2.3. Select certificate

### Authentication certificate

CLA	INS	P1	P2	Lc	Data
00	A4	02	04	02	AACE

### Digital signature certificate

CLA	INS	P1	P2	Lc	Data
00	A4	02	04	02	DDCE

## 2.4. Read certificate

CLA	INS	P1	P2	Le
00	B0	01 - FF	00	00

## 3. Verify PIN1

### 3.1. Select Master File directory

CLA	INS	P1	P2	Le
00	A4	00	0C	00

### 3.2. Select EEE file

CLA	INS	P1	P2	Lc	Data
00	A4	01	04	02	EEEE

### 3.3. Set Security environment

CLA	INS	P1	P2	Le
00	22	F3	01	00

### 3.4. Verify PIN1

CLA	INS	P1	P2	Lc	Data(PIN1 as ASCII)
00	20	00	01	04	31323334

#### 4. Sign hash using PIN1

PIN1 verification (described in section 3 of Appendix 1) required before executing this APDU

CLA	INS	P1	P2	Lc	Data	Le
00	88	00	00	Hash array length	Hash array	00

#### 5. Verify PIN2

##### 5.1. Select Master File directory

CLA	INS	P1	P2	Le
00	A4	00	0C	00

##### 5.2. Select EEEE file

CLA	INS	P1	P2	Lc	Data
00	A4	01	04	02	EEEE

##### 5.3. Set security environment

CLA	INS	P1	P2	Le
00	22	F3	01	00

##### 5.4. verify PIN2

CLA	INS	P1	P2	Lc	Data(PIN2 as ASCII)
00	20	00	02	05	3132333435

#### 6. Sign hash using PIN2

PIN2 verification (described in section 5 of Appendix 1) required before executing this APDU

CLA	INS	P1	P2	Lc	Data	Le
00	2A	9E	9A	Hash array length	Hash array	00

## II. Cryptographic Algorithm Benchmarking

**Abstract:** Appendix 2 describes different cryptographic algorithm speeds and compared them with each other. Main focus is comparing symmetric and asymmetric cryptography algorithms, with different key and block sizes and to identify the best suitable cryptographic algorithms for NFC security solution proposed in this thesis.

Hardware used for testing cryptography speeds

Operating system	OS X Yosemite
Processor	2.2 GHz Intel core I7
Memory	16 GB 1600 MHz DDR3
Graphics	Intel Iris Pro 1536 MB

Performance analysis done using OpenSSL built in speed testing tools, with minimized operating system load on the hardware.

### Hash generation

Block size	SHA1 (Hash/s)	SHA256 (Hash/s)	SHA512 (Hash/s)	MD5 (Hash/s)
16	2927071.3	1939447.3	1453248.7	2619817.3
64	2051071.3	1168507.0	1451481	2002429
256	1161599.7	552187.7	666911.7	1077459
1024	416352.3	171633.7	258243	365903.7
2048	57689.7	22411.7	37263	52981

### Symmetric Encryption algorithms

Block size	3DES (Enc/s)	DES (Enc/s)	AES128 (Enc/s)	AES192 (Enc/s)	AES256 (Enc/s)
16	1724741.7	4482440	8488211.7	7102791	6279469.3
64	419486	1113314.3	2141398	1897358	1722419.7
256	109878.7	273940	535701	451289.3	434633
1024	27567.3	67393.3	138249	112720.7	108602.7
2048	3465	8492.7	16524	14815	13540.3

## Asymmetric Encryption algorithms

### RSA

Key size (bit)	RSA (Sign/s)	RSA (Verify/s)
512	7334.6	101908.4
1024	2099.3	43349.3
2048	397.3	15873.5
4096	64.9	4708.0

### Elliptic Curve

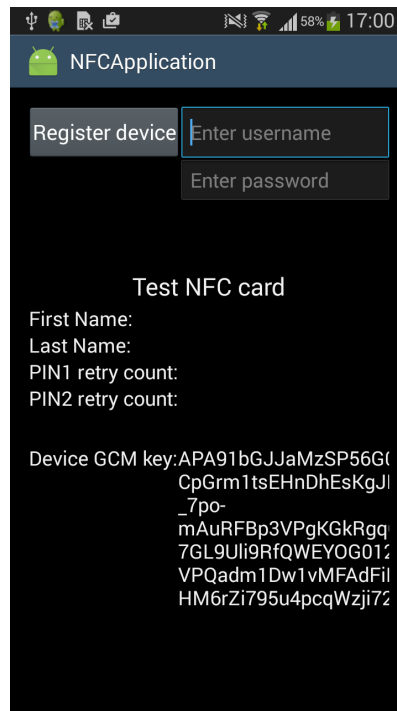
Key size (bit)	EC (Sign/s)	EC (Verify/s)
160	9483.5	2155.7
192	9186.5	2138.2
224	6317.7	1432.6
256	5132.4	1158.6
384	2660.7	552.0
512	2453.6	504.5
571	142.0	69.5

### III. NFC Security Solution for Web Application Prototype

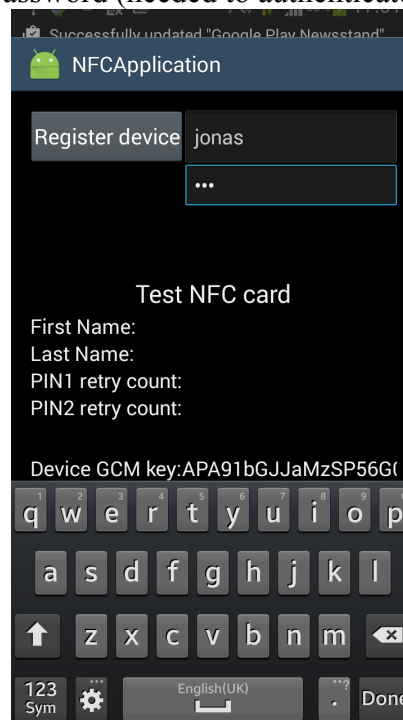
**Abstract:** Appendix 3 describes the different user activity flows within NFC security solution for web application prototype. 5 different flows - device authorization, user authorization, signing, device unauthorisation and NFC card test are described with visual aids from web application and mobile application.

#### 1. Device authorization

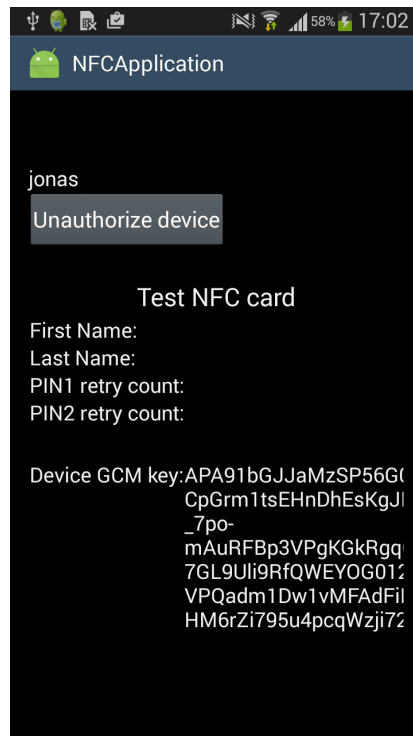
Start NFC mobile application for the first time on a new device (unauthorized device)



Fill in server username and password (needed to authenticate user for web application)




Press register device button, to authenticate this device as you NFC authentication end-point



Your device has been successfully authenticated


## 2. User authorization

 NFC Authentication Prototype

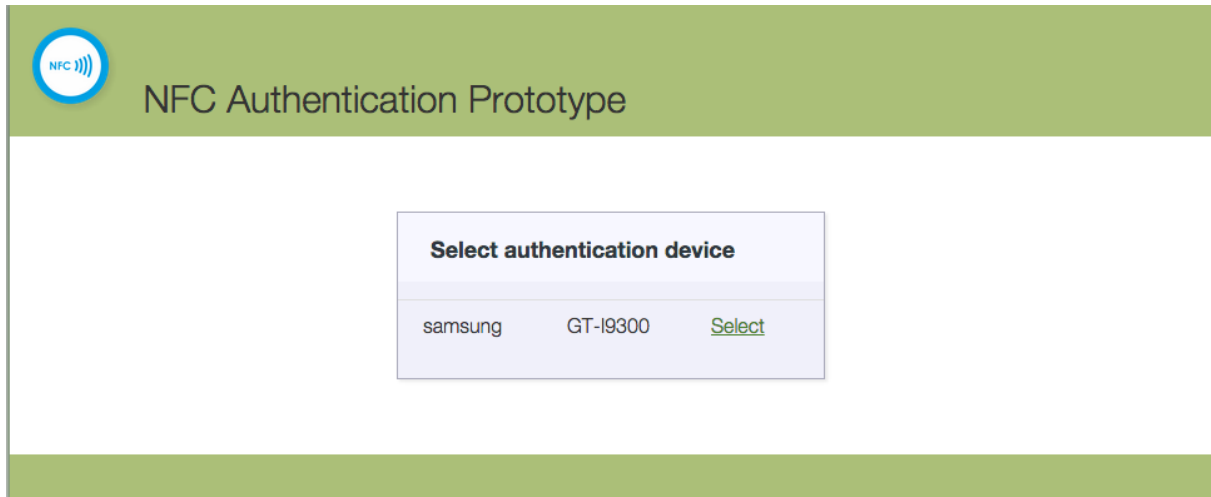
**Please Login**

Username:

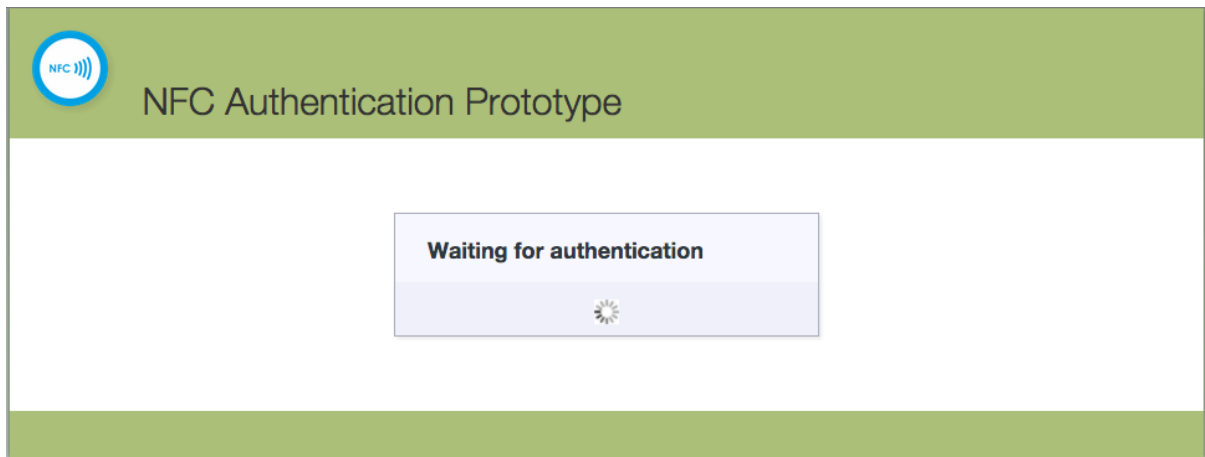
Password:



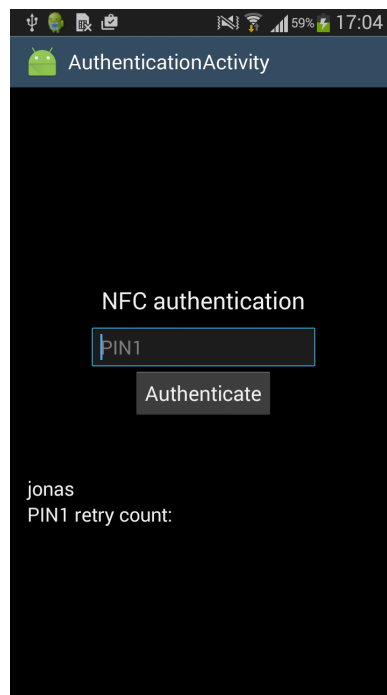
Open web application and authenticate user with username and password



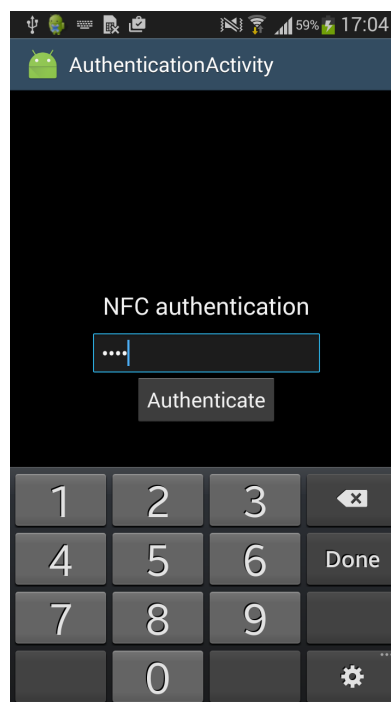
After successful username and password authentication, user gets redirected to select NFC authentication device page, where user must select one device to authenticate with.



When user has selected the NFC authentication device, an authentication request is sent to selected device via GCM service.

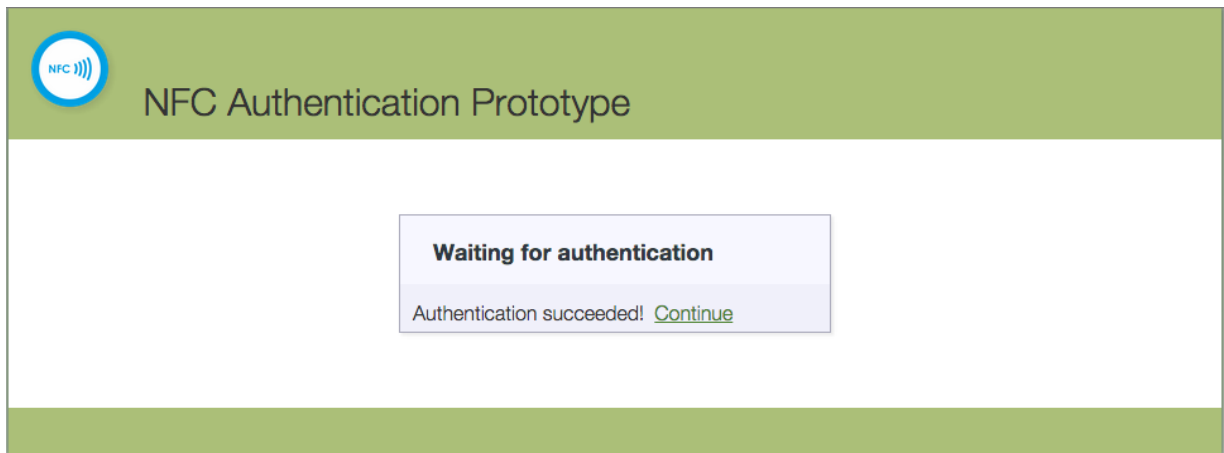


New authentication request is shown to user (Requiring user to authenticate himself/herself using NFC card and PIN1 passphrase).

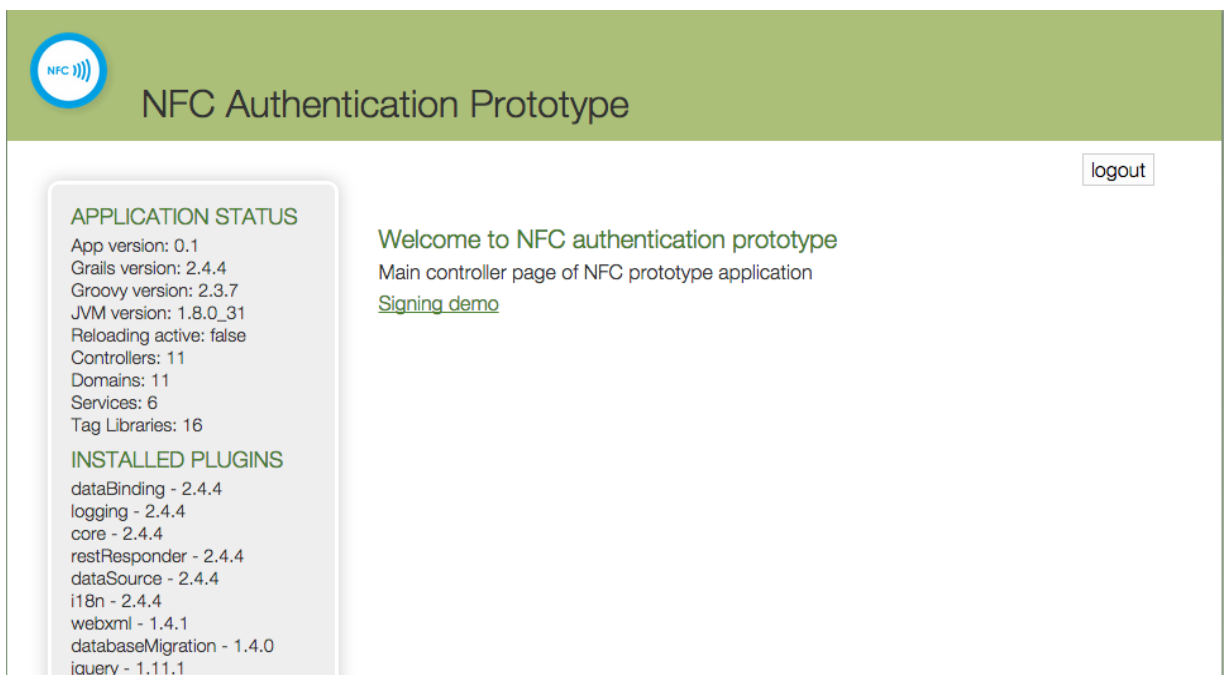


User must fill in the PIN1 passphrase with 4 digits, matching the personalized PIN1 on the NFC card.



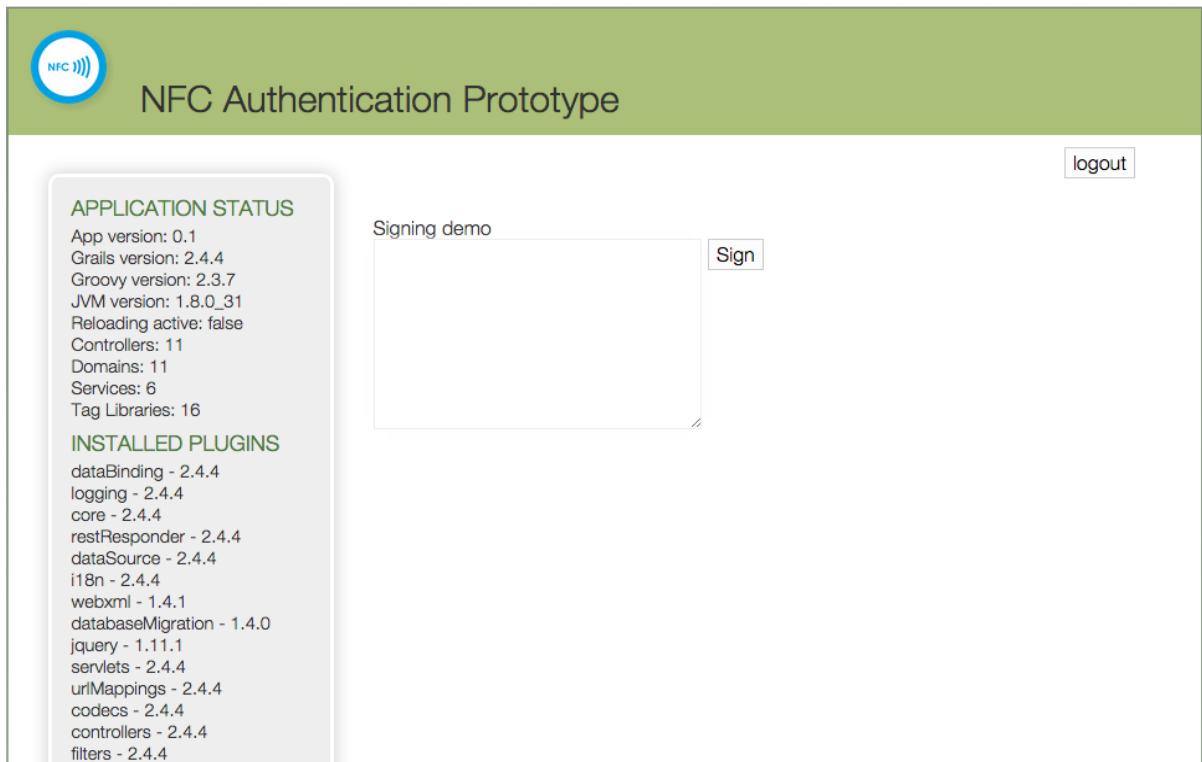


User authentication was successful, and user can continue to using the web application

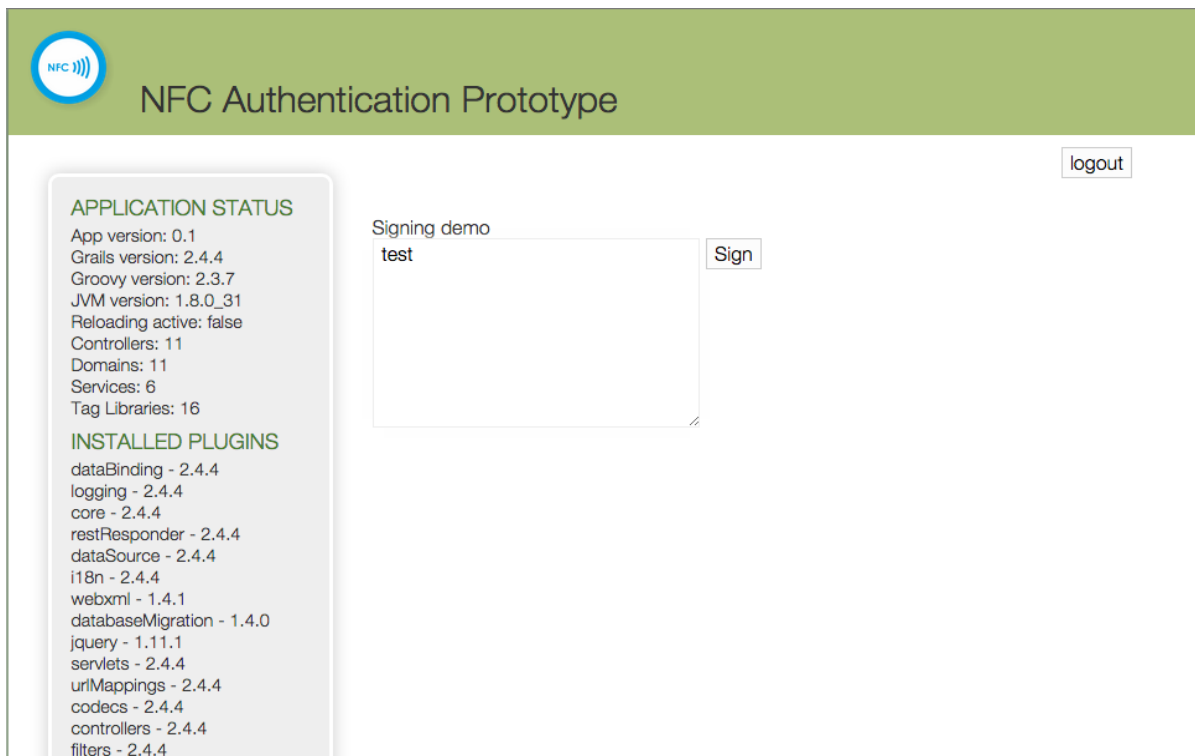


After user has agreed to continue, he/she gets redirected to main page of the web application

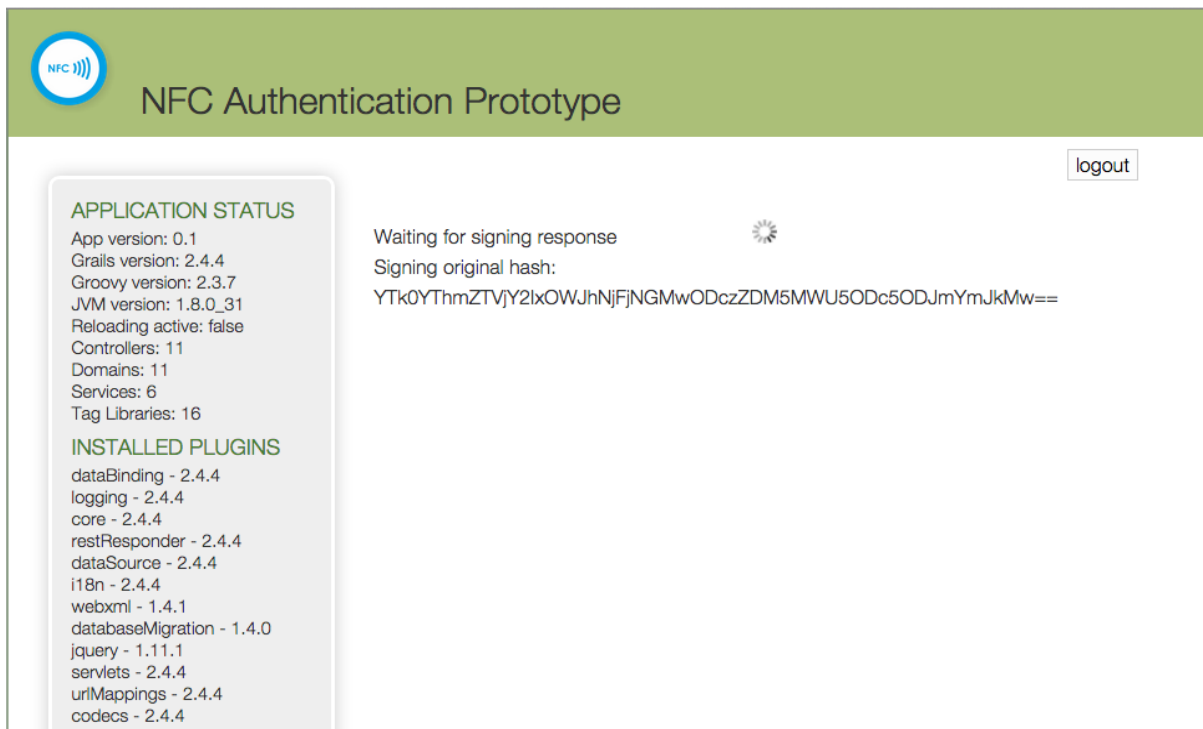
### 3. Signing



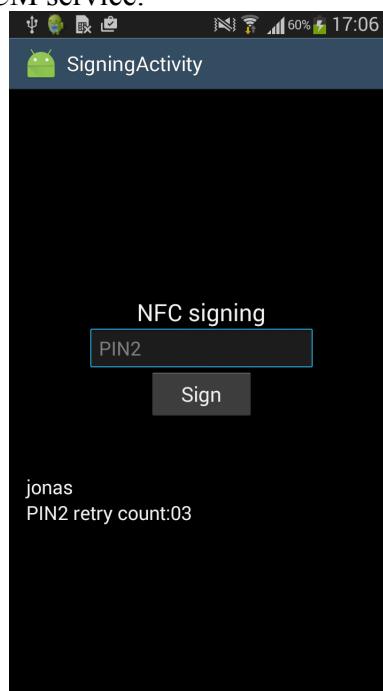
In order to start signing prototype, user must open signing demo view



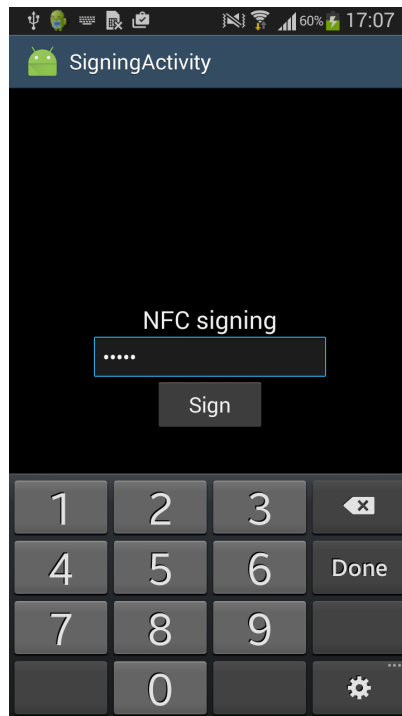
User must enter the text phrase he/she wants to sign and press “Sign” button



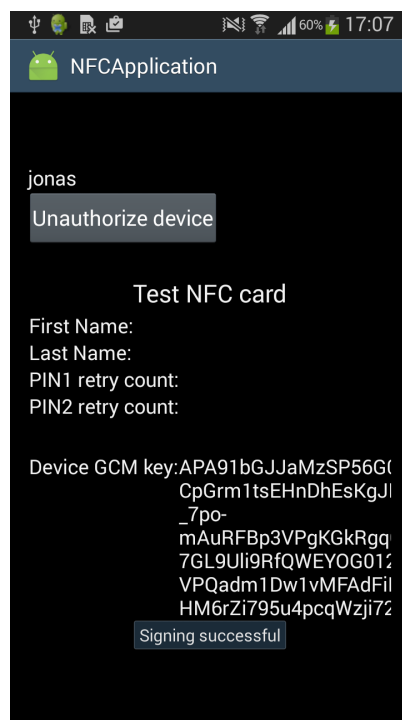
A new signing request is forwarded to user authenticated mobile device (same device used in authentication flow) via GCM service.




A new signing request is received in user mobile device and signing flow has been started in NFC mobile application



User must fill in the PIN2 passphrase, matching the PIN2 passphrase personalized into NFC card.



Signing has succeeded and signing request is closed.



# NFC Authentication Prototype

logout

## APPLICATION STATUS

App version: 0.1  
Grails version: 2.4.4  
Groovy version: 2.3.7  
JVM version: 1.8.0\_31  
Reloading active: false  
Controllers: 11  
Domains: 11  
Services: 6  
Tag Libraries: 16

## INSTALLED PLUGINS

dataBinding - 2.4.4  
logging - 2.4.4  
core - 2.4.4  
restResponder - 2.4.4  
dataSource - 2.4.4  
i18n - 2.4.4  
webxml - 1.4.1  
databaseMigration - 1.4.0  
jquery - 1.11.1  
servlets - 2.4.4  
urlMappings - 2.4.4  
codecs - 2.4.4  
controllers - 2.4.4  
filters - 2.4.4

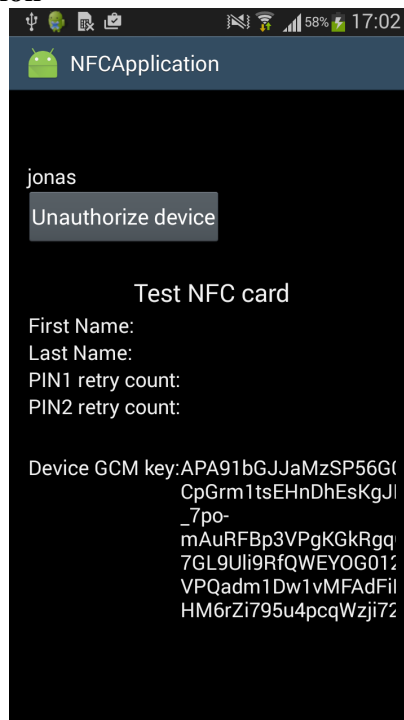
Signing succeeded!

Signing original hash:  
YTk0YThmZTVjY2lxOWJhNjFjNGMwODczZDM5MWU5ODc5ODJmYmJkMw==

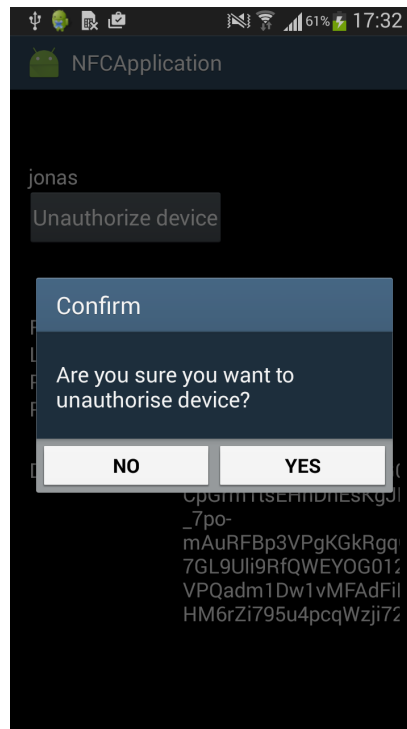
Signing decrypted hash:  
YTk0YThmZTVjY2lxOWJhNjFjNGMwODczZDM5MWU5ODc5ODJmYmJkMw==

Signing response is received by web application and the result is displayed to the user.

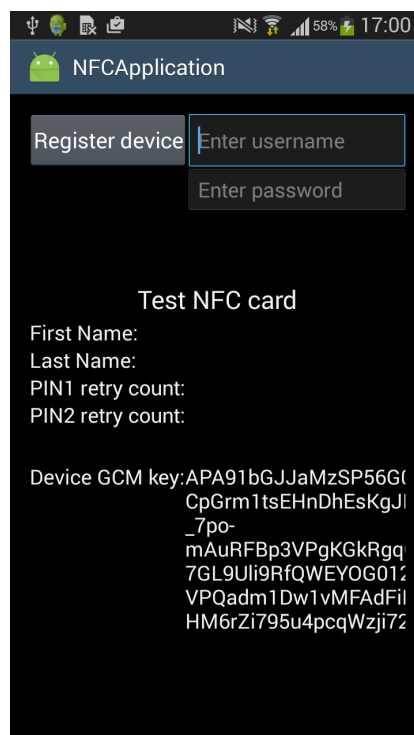
#### 4. Device unauthorisation



In order to authorize the device, user must open the NFC application in mobile device and press “Unauthorise device” button

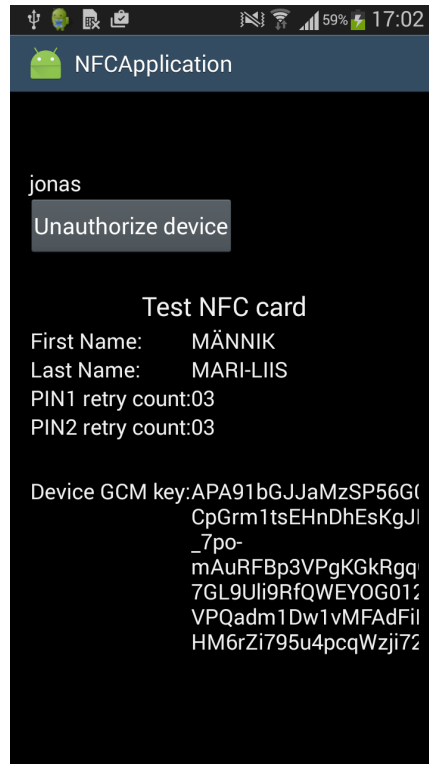


Unauthorisation verification is requested from the user. If the user decides to unauthorize the device, then after pressing the “YES” button, the device gets unauthorized.



When device is unauthorized, it opens the possibility to register the device again with web application username and password.

## 5. NFC card test



To test the NFC connection, there is the possibility to open NFC mobile application and simply pair the device with NFC card. If connection is successful, NFC card personalized first name, last name, PIN1 retry count and PIN2 retry count is displayed.

#### IV. Rest API calls

Challenge name	Direction	Part	Value
1. Verify device	POST	URL	https://localhost:8080/api/verify-device
		JSON	{ device_key: "ewjoi4jio4j34oij4oi34jo3j43o5j35", device: { osversion:"test", release:"test", device:"test", model:"test", product:"test", brand:"test", display:"test", cpuabi:"test", unknown:"test", hardware:"test", buildid:"test", manufacturer:"test", serial:"test", deviceuser:"test", host:"test", imei:"test", imsi:"test", numberline1:"37255526262", mac:"test", screen " : "345x345", androidid:"test", deviceid ":"test", } }
	GET	JSON	{ status:"OK" or "NOK", message: 1-8, username: "jonas" }
2. Add device	POST	URL	https://localhost:8080/api/add-device
		JSON	{



			<pre> device_key: "ewjoi4jio4j34oij4oi34jo3j43o5j35", username: "jonas", device: {   osversion:"test",   release:"test",   device:"test",   model:"test",   product:"test",   brand:"test",   display:"test",   cpuabi:"test",   unknown:"test",   hardware:"test",   buildid:"test",   manufacturer:"test",   serial:"test",   deviceuser:"test",   host:"test",   imei:"test",   imsi:"test",   numberline1:"37255526262",   mac:"test",   screen : "345x345",   androidid:"test",   deviceid:"test", } } </pre>
	GET	JSON	<pre> {   status:"OK" or "NOK",   message: 1-8,   username: "jonas" } </pre>
3. Unauthorize device	POST	URL	https://localhost:8080/api/unauthorize-device
		JSON	<pre> {   device: {     osversion:"test",     release:"test",     device:"test",     model:"test",     product:"test", </pre>

			<pre> brand:"test", display:"test", cpuabi:"test", unknown:"test", hardware:"test", buildid:"test", manufacturer:"test", serial:"test", deviceuser:"test", host:"test", imei:"test", imsi:"test", numberline1:"37255526262", mac:"test", screen" : "345x345", androidid:"test", deviceid":"test",  } } </pre>
	GET	JSON	<pre> {   status:"OK" or "NOK",   message: 1-8,   username: "jonas" } </pre>
4. Authentication result	POST	URL	https://localhost:8080/api/authentication-result
		JSON	<pre> {   username: "jonas",   original_hash:"Base64 generated hash",   encrypted_hash:"Base64 encrypted hash" } </pre>
	GET	JSON	<pre> {   status:"OK" or "NOK",   message: 1-8,   username: "jonas" } </pre>

5. Sign result	POST	URL	https://localhost:8080/api/sign-response
		JSON	{ username: "jonas", original_hash:"Base64 generated hash", encrypted_hash:"Base64 encrypted hash" }
	GET	JSON	{ status:"OK" or "NOK", message: 1-8, username: "jonas" }

## **VI. Source code**

The source code of the NFC web application solution can be downloaded from bitbucket git repository <https://username@bitbucket.org/jonx/nfc-grails.git>

The source code of the Android application can be downloaded from bitbucket git repository <https://username@bitbucket.org/jonx/nfc-android.git>

## **VII. License**

### **Non-exclusive licence to reproduce thesis and make thesis public**

I, **Jonas Kiiver** (date of birth: 23.01.1990),

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:
  - 1.1. reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and
  - 1.2. make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

of my thesis

**NFC Security Solution for Web Applications,**

supervised by Professor Eero Vainikko,

2. I am aware of the fact that the author retains these rights.
3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, **21.05.2015**