Offline NFC Payments with Electronic Vouchers

Gauthier Van Damme Karel Wouters

Hakan Karahan Bart Preneel*

Dept. Electrical Engineering-ESAT/SCD/IBBT-COSIC
Katholieke Universiteit Leuven
Kasteelpark Arenberg 10, 3001 Heverlee-Leuven, Belgium

{gauthier.vandamme, karel.wouters, bart.preneel}@esat.kuleuven.be, hakan.karahan@rwth-aachen.de

ABSTRACT

In this paper a practical offline payment system based on digital vouchers using Near Field Communication (NFC) in mobile phones is presented. This work was performed within the scope of the IBBT NFC-Voucher project. The goal of the project is to assess the feasibility of such a system, from a technical and security perspective, using tangible NFC devices such as the Nokia 6131 NFC mobile phone. This involved an in-depth technical and security analysis of all actors in the system and a rigorous elaboration of the practical security requirements and assumptions. In the architecture implementing and connecting all the different actors of this voucher payment system, no compromises regarding security were made. At device level all sensitive data is stored in a Secure Element (SE) with limited access for non-authorised users. The backbone and voucher transfer system uses a classical Public Key Infrastructure (PKI), such that only trusted and registered parties can handle and transfer vouchers. After having implemented this system, we conclude that it is possible to build an off-line payment system for mobile phones without compromising security, but that it remains quite challenging, given the current limitations on speed, available memory and security functionality.

Categories and Subject Descriptors

K.4.4 [**Electronic Commerce**]: Payment schemes, Security

*These authors' work was supported by the Interdisciplinary institute for BroadBand Technology (IBBT) and in part by the Concerted Research Action (GOA) Ambiorics 2005/11 of the Flemish Government as well as the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy).

MobiHeld'09, August 17, 2009, Barcelona, Spain.

General Terms

Algorithms, Design, Security

Keywords

NFC, Security, Voucher, Nokia

1. INTRODUCTION

While mobile communications are growing far beyond classical wired ones, the world as a whole is getting connected. In this evolution Near Field Communications (NFC) [14] will play a central role. NFC is a short-range wireless communication technology that extends the ISO 14443 standard for RFID technology [15]. Therefore any NFC enabled device can communicate with other NFC devices and with any existing RFID infrastructure, such as readers and contactless cards. The range in which NFC devices can communicate is about 10 cm. Compared to RFID or even Bluetooth technology that have a much larger range, this provides NFC devices with a higher degree of security. This is because both sniffing communications and man-in-the-middle attacks are then harder if not impossible to accomplish.

Some of the main applications of NFC technology are those in which mobile phones are involved. Mobile phones already outgrew their original communication purpose, evolving into portable multimedia system, and are becoming an indispensable attribute for a growing community. NFC enabled phones, being able to run multiple applications, can replace a large part of the contemporary physical wallets, providing mobile ticketing, mobile payment, connection to smart posters, the use of electronic keys and so on.

To achieve an offline system for mobile payments, a highly secured architecture had to be developed, mainly to avoid voucher (and hence value) duplication, even in the very unlikely event of well-funded hardware attacks. As for the business case behind a potential pilot program, it should be noted that in Belgium alone, 250 million paper meal vouchers are issued per year at the moment[1]. Apart from these meal vouchers, the same infrastructure could be used for

[1]`http://www.standaard.be/Artikel/Detail.aspx?artikelId=BS1UJ9HN` In Belgium, 40% of all employees receive meal vouchers; meal vouchers make up for 4% of the total salary.

distributing service vouchers, gift vouchers and commercial coupons.

In the first part of this paper, related work is presented and compared with the newly developed system. Then the system architecture is presented, followed by the implementation aspects. Finally conclusions are made about the feasibility of the presented system.

2. RELATED WORK

In the last few years, numerous pilot projects have been executed in the area of NFC payment systems. Even prior to the introduction of NFC or RFID, contact smart cards existed for several decades.

Contact smart cards are frequently used for payments (credit and debit cards), authentication and identification, pre-pay household utilities, and public transport. Moreover, they can also be used as electronic purses; in this case, cryptographic protocols protect the exchange of money between the smart card and payment terminal, which also allows for offline transactions. Examples include Proton [8], Chipknip [2] and Geldkarte [3]. These cards are used for small payments and can support off-line transactions. One of their main disadvantages is that they do not support transfer of money between cardholders. The main reasons for this are security concerns and the fact that card readers must be involved. Moreover, to load money onto the card, direct interaction with a terminal is needed.

Since 2006 contactless payment technology has been emerging, including Visa payWave [10], MasterCard PayPass [6] and Chase's Blink [1]. These technologies mainly use NFC as a new communication means to the already available credit card network, and are online payment applications. Moreover, customer-to-customer transfer cannot be done. This trend is also noticeable in major consortiums focusing on mobile payments, such as the StolPaN [9] consortium and the GSM Association [11].

Finally, a large number of NFC trials are being conducted at the time of writing [7]. Most of them do not target offline payments nor customer-to-customer transfer, thereby making the system more manageable from a security point of view.

To summarise, our solution is different in the following ways:

- no security compromises - we do not rely on the phone's OS for security-sensitive operations,
- support for offline payments,
- offline customer-to-customer transfer of value,
- no reliance on Mifare,
- phone-based rather than card-based.

3. ARCHITECTURE

In this section, we present the security architecture for the NFC Voucher system. In this system, vouchers are made available in electronic format on a NFC-compatible device, allowing the user to pay electronically by touching a merchant's payment terminal with the NFC device. First, the basic structure with all its actors and possible transactions will be summarised. In the second part of this section, the security requirements of this system are discussed, followed by an overview of the key infrastructure. Finally, we present in detail the phone-to-phone transfer as this is one of the scenarios which is conducted completely offline.

3.1 The eVoucher transactions

The basic structure of the developed system is shown in Figure 1. The electronic voucher (eVoucher) system consists of three types of actors, namely eVoucher Issuers, eVoucher users (Beneficiaries) and owners of payment terminals (Affiliates).

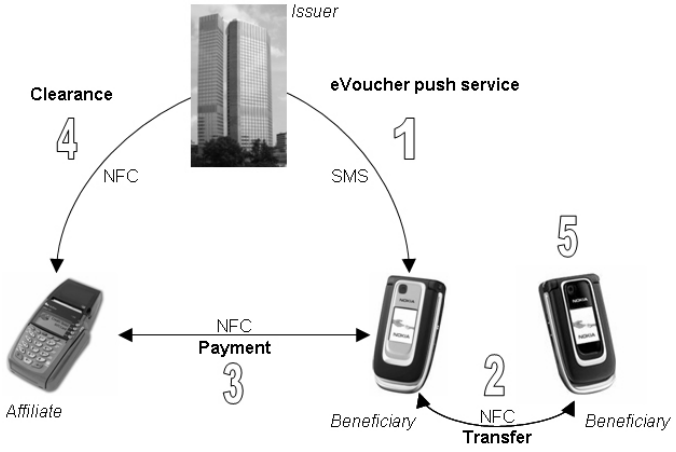

Figure 1: The eVoucher system

The system implements the following types of transactions between actors.

1. Receiving eVouchers from the issuer: SMS based, unconfirmed transaction from issuer to beneficiary.
2. Transferring eVouchers from device to device (beneficiary to beneficiary):
   - Using NFC, with confirmation of receipt.
   - SMS from one phone to another.
   - Through an intermediate RFID tag.
3. Payment: Transferring eVouchers from device (beneficiary) to a payment terminal (affiliate).
4. Clearance happens between the affiliate and the issuer, and happens over a secured (possibly dedicated) network.
5. Users can manage their eVouchers at any time. They can check their balance, check eVoucher expiration dates, retrieve their spending history, etc.

3.2 Security concerns, deployment and usage

The main concern when implementing any payment application should be the security of the transfer of value, especially in offline systems. This is because in payment schemes in which value can be transfered offline, there is no main server able to control and audit all transactions. Such a scenario closely resembles a paper-based system in which the issuer tries to protect his eVouchers (or money) by making them hard to copy. Similarly, digital value should be hard to copy. Another risk with digital value is that it might get

lost in transaction. Therefore, counterfeiting, copying and loss were the main risks to protect against, in our system. To achieve this, a circle of trust – described below – had to be constructed.

In a first stage, the beneficiary will deploy the NFC Voucher software on his phone. The software consists of two parts: a MIDlet[2], running in the phone's operating system, and a JavaCard Applet, running in a secured hardware component on the phone. This hardware, the so-called Secure Element (SE), will only accept new software from a Trusted Services Manager (TSM) holding a private key allowing authentication to the SE. Therefore, we can be sure that the NFC Voucher Applet will be deployed securely and can be fully trusted to handle valuable data such as eVouchers. Also note that the NFC Voucher MIDlet has to be code-signed, before the phone OS grants access to the SE (and hence the NFC Voucher Applet). The reason critical data is stored on a SE and not in a MIDlet, is that neither the MIDlets nor the phone's OS (S40) can be fully trusted [13].

First, the user will install the MIDlet, after which a connection to a TSM is established for installing the NFC Voucher Applet in the SE. When all software is installed, an initialisation protocol is executed with the Issuer. The public keys of the issuer are hard-coded in the applet, but the RSA key pair of the beneficiary, used to authenticate to other phones and payment terminals, is generated in the SE itself. During this initialisation phase, the cell phone number (or ID) and the public key (RSA) of the beneficiary are registered by the issuer: Upon reception of the public key of the beneficiary, the issuer generates a X.509 certificate for that public key and sends it to the SE. This is the only phase in which a network connection between the phone and the issuer is established.

In a later stage, eVouchers are produced by the issuer, essentially by generating signatures. The eVouchers are packed in a byte array and encrypted for the intended SE. The resulting blob is sent through SMS to the (registered) cellphone number. Upon reception of such a SMS, the phone starts the right MIDlet and transfers the encrypted blob to the SE, which decrypts it, unpacks the eVouchers and checks the signature on each. eVouchers are signed separately because they can be spent separately; leaving them unsigned in the SE would enable any SE to generate its own eVouchers. This would would lead to an uncontrollable and untraceable generation of fake eVouchers in the case of a breached SE[3].

An eVoucher consists of a serial number, a validity period, an amount in Eurocents, a status byte (dirty, spent, expired) and a 128 byte RSA signature by the issuer, yielding an eVoucher size of 148 bytes. Users can manage their eVouchers through the Voucher MIDlet. This allows them to check their balance and inspect their stored and spent eVouchers. They may also check when their next eVouchers will expire.

When transferring and paying with eVouchers, interruptions –malign or accidental– may occur. Therefore, an SE sending eVouchers will mark them as 'dirty' as soon as they have left the SE. This is a design choice in favour of security. If a transaction has failed, the beneficiary can reclaim dirty eVouchers at a point in time, when their validity date has passed.

3.3 Key Infrastructure

Given the description above, the NFC Voucher system features the following key infrastructure.

- The issuer holds two (RSA) keypairs: one signature keypair and one encryption keypair. The signature keypair is used to sign eVouchers and generate certificates for payment terminals and SEs. The encryption keypair provides confidential communication to applets wishing to register their keypairs during the applet initialisation phase. Both keys are hard-coded into the deployed applet, and can be updated by deploying a new applet through the TSM.
- The NFC Voucher Applet in the SE holds a single RSA keypair, for which it acquires a certificate, signed by the issuer. Using the certificate and the corresponding private key, the applet/SE can communicate in a secure and authenticated way with other applets/SEs and payment terminals. Standard PKI techniques can be employed to manage this keypair's lifecycle. For practical reasons, this keypair is used for signing as well as for encryption.
- Similar to applets, payment applications in payment terminals will also be equipped with a keypair and a corresponding certificate, provided by the issuer. Because the payment terminal will have a communication channel for eVoucher clearance, an initialisation phase, similar to the one with the SE, can be performed over that channel.

3.4 Protocols

To handle all the types of eVoucher transfers, different protocols were developed. We elaborate on the NFC phone-to-phone transfer of eVouchers, because it is performed completely off-line, without any trusted party interaction. Other transfer protocols (issuer-to-beneficiary, payment protocol) follow a similar but less complex protocol. Payment operations can be conducted on- or off-line, because of the underlying PKI.

In the phone-to-phone protocol, the SE of the receiving phone acts as a passive device (smart card), while the sender emulates a smart card terminal, with communication security starting in the SE. This terminal mode has to be facilitated by a MIDlet, running on the sender's phone, turning the SE at the sender's side into an *active* device. The MIDlet merely acts as a gateway, passing data between the receiver's SE and its own (sender's) SE.

The phone-to-phone protocol is depicted in figure 2. We briefly comment on the stages:

1. First, the sender selects the MIDlet to transfer eVouchers. After selection of the amount to be transferred

[2]A MIDlet (or Midlet) is an a Java ME program written for MIDP. Mobile Information Device Profile (MIDP) is a specification published for the use of Java on embedded devices such as mobile phones and PDAs. MIDP is part of the Java Platform, Micro Edition (Java ME) framework and sits on top of Connected Limited Device Configuration (CLDC), a set of lower level programming interfaces. Source: `http://wiki.forum.nokia.com`

[3]Although the SE is designed to protect against a certain level of attacks. Designing a system in which SEs have the power to generate value, would possibly attract well-funded hackers, deploying advanced hardware attacks such as chip decapsulation, probing, focused ion beam cutting and drilling at microscopic levels.

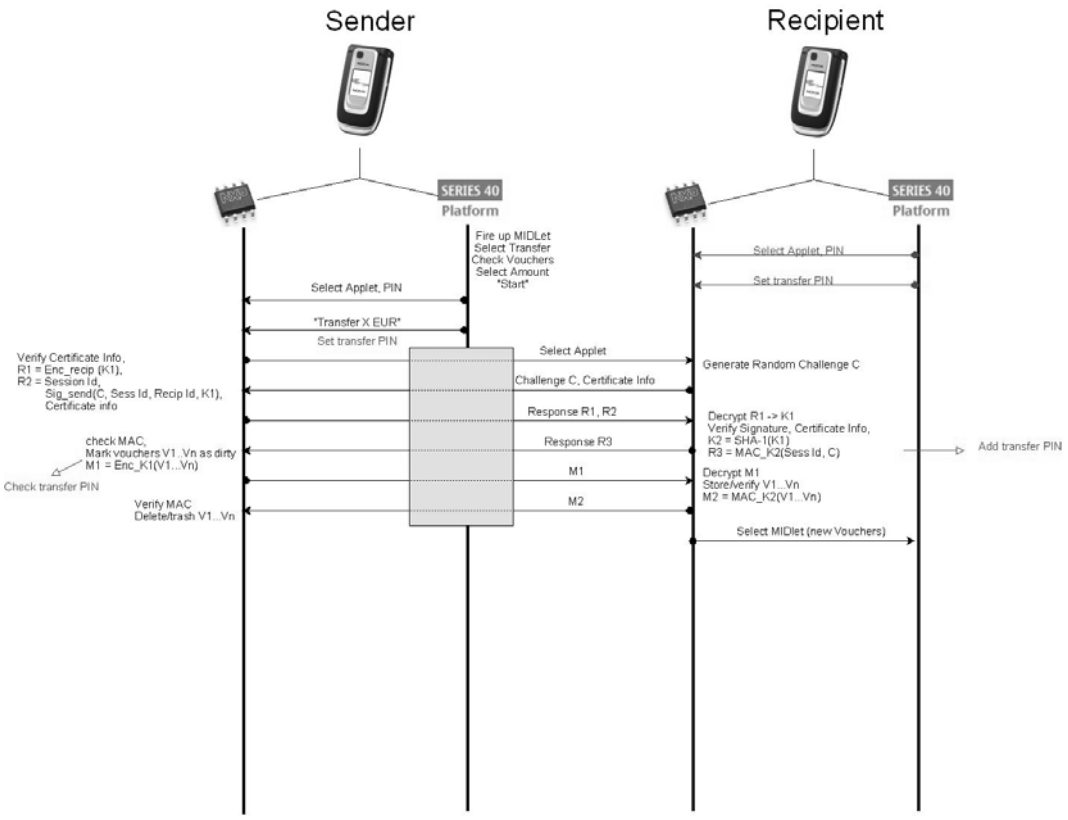

Figure 2: The phone-to-phone protocol

(which can be determined after some interaction with the eVoucher applet in the SE), the MIDlet selects the eVoucher transfer applet, which is protected by a PIN, stored by the MIDlet.

2. When the applet is successfully selected, the MIDlet instructs the transfer applet to transfer X EUR over NFC. Optionally, a transfer PIN has to be set, in order to limit the risk of accidentally transferring the eVouchers to non-intended devices. If this transfer PIN is used, the Recipient must set it and pass it on to his SE. The PIN is agreed upon by the devices' owners verbally.

3. When the applet in the SE is instructed to start the transfer protocol, it interacts with the SE at the receiver's side by selecting the transfer applet.

4. Upon selection, the SE at the receiver's side (R-SE) generates a random challenge C, and sends it to the sender's SE (S-SE), together with the PKI certificate of the key, associated with the applet (and therefore the Recipient). This PKI certificate contains the ID of the Recipient, and his public key.

5. Upon reception of C and the certificate, the S-SE does the following:

   - check the certificate info, extract the Recipient ID and the public key of the R-SE.
   - generate a random symmetric (3DES) session key K1
   - RSA-encrypt K1 with the public key of the R-SE. The result is R1
   - generate a session ID
   - RSA-Sign (C, Session ID, Recipient ID, K1) with the private RSA key of the S-SE. Together with the Session ID and the certificate info of the S-SE, this results in R2
   - send R1,R2 to the R-SE

6. Upon reception of (R1, R2), the R-SE performs the following steps:

   - RSA-decrypt R1 with the private key of the R-SE. This results in K1
   - store the Session ID
   - verify the certificate of the S-SE
   - verify the signature of the S-SE over C, Session ID, Recipient ID and key K1
   - compute the MAC (Message Authentication Code) key K2=SHA-1(K1)
   - Compute a 3DES MAC over the Session ID and C, with key K2; this is message R3. Optionally, the transfer PIN is also added to this MAC
   - send R3 to the S-SE

7. The S-SE receives R3 and checks the MAC. Then, it marks the eVouchers to be transferred as dirty, 3DES-encrypts them under key K1 (resulting in M1) and sends them over to the R-SE.

8. The R-SE decrypts M1, and stores and verifies the eVouchers included. Then, it computes a 3DES MAC M2 over the received eVouchers under key K2, and sends it to the S-SE.

9. Upon reception of the M2, the S-SE checks the MAC and if successful, it deletes/trashes the eVouchers that were sent.

10. The R-SE can then notify the user of the arrival of new eVouchers.

In this protocol the symmetric key K1 is guaranteed to be generated within a SE because it is signed with a certified keypair. Also, after response R3, the S-SE and the R-SE are assured they are communicating with legitimate parties: The R-SE proves it possesses the private key linked to its certificate by correctly recovering key K1. The S-SE shows its possession of the private key by correctly signing C, Session ID, Phone ID and K1, establishing that C was received correctly, the Session ID is genuine, K1 is genuine and the message is actually meant for this R-SE. Also, because of the signature, the R-SE assumes that the key K1 is fresh (because SE's are trusted).

4. IMPLEMENTATION AND LIMITATIONS

In this section we describe hardware and software used and provide details of implementation limits and runtime tests. For our approach we used the Nokia 6131 NFC [17] and its successor, the Nokia 6212 NFC [18] which are the only two phones by Nokia with NFC capabilities at the time of writing.

The SE on the Nokia 6131 runs on Giesecke & Devrient's Sm@rtCafé Expert 3.1 operating system and consists of a Mifare 4K[4] area and a Java Card compliant to Global Platform 2.1.1 [12] and Java Card 2.2.1 [5]. There is 65 KB memory space available, which is used to store applets. For deployment on the phone we used the open source tool GP-Shell [4] which enables the installation and removal of applets. It also offers a simple way to communicate with stored applets through scripts or direct use of command shell. For transaction testing we used a standard PC with an Omnikey Cardman 5321 Reader to emulate a merchant's payment terminal with NFC interface. The software for this was written in Java using the Java Card Development Kit [19].

When developing our implementation, we encountered some limitations: The SE supports several common cipher, signature and message digest algorithms, such as 3DES, RSA and SHA-1. These algorithms where sufficient for our approach, but to further decrease computation time and eVoucher size, AES and ECDSA would have been desirable, as these are more suitable for a limited environment such as a SE. Regrettably, they are not supported on the Nokia phones in our project.

Obviously, it is not possible to store arbitrarily many eVouchers in the SE. Our application requires about 30 KB of memory using 20 eVouchers – including the certificates for the phone and the public key of the issuer. As an eVoucher consists of 148 bytes in total, a maximum of 262 vouchers can be stored on the phone. This is no drawback for the application, since the vouchers will have limited validity similar to their paper-based counterparts.

[4]`http://mifare.net/`

Another limitation is the maximum APDU[5] size of 256 bytes. This causes some overhead since for most steps in the transaction more than 256 bytes have to be sent. Therefore, the payload is segmented and marked with a status word indicating whether data is still pending or not. This decreases the transaction speed.

Finally we benchmarked some eVoucher transaction scenarios between two phones (phone-to-phone transfer) with the two Nokia phone types. Timing results were similar for both phones, even if in our opinion transactions with the newer Nokia 6212 phone are more difficult to perform than with the Nokia 6131. The first reason for this is that the antenna is not focused on a specific point and secondly the newer phones do not always respond immediately to each other when brought into close range.

A complete transaction with 10 eVouchers from one phone to another takes about eight seconds. This is unfortunately not as fast as we had hoped for; satisfactory timings would be situated around 1 second. Increasing the number of eVouchers, from 10 to 20 only slightly increases the transaction time. This shows that, independently of the number of eVouchers sent, there is a large overhead. Further measurements confirmed this. The main reasons for this overhead lie in the cryptographic protocols used and in the way the phones communicate with each other. In both Nokia NFC phones it is impossible for a MIDlet to have an open connection to the internal SE and to an external SE at the same time. This forces the MIDlet that routes the APDUs from the internal SE to the external SE and back to close a connection and reopen it at every step of the protocol. The overhead that this switching between internal and external connections to the SEs causes is reinforced by the fact that the MIDlet uses a targetListener to detect possible external connections before one can be set up. This obviously is a time consuming task, especially when for every transaction, the external SE has to be accessed three times. Fortunately, this overhead is not present in phone to terminal payment transactions, because a terminal can keep a single external connection open for the whole transaction time. For this reason, terminal payments are faster than phone to phone transactions. The overhead of using RSA is another significant reason for the unsatisfactory timings. After some timing experiments, we concluded that half of the transaction time is spent to perform asymmetric key operations (in our case RSA). Although the SE features a cryptography coprocessor, able to perform raw 1024-bits RSA operations in less than 20ms, the actual RSA performance seems to be 10 times slower. This can be because of the SE operating system and the JavaCard Virtual Machine running on top of the hardware. Similar experiences with Java Virtual Machines running on fast cryptographic coprocessors – in different settings – have been noted by other authors [16].

5. CONCLUSION

In this paper, a highly secure, offline NFC based eVoucher payment system using NFC enabled mobile phones was presented. A major advantage of the presented system when compared to others is the possibility for users to perform and complete offline user-to-user transactions and to be able to consult their actual balance and other information whenever

[5]APDU stands for Application Protocol Data Unit; it is the communication unit between a reader and a SE.

and wherever they want, without having to connect to an external server and thus without any extra costs. The concrete implementation of the system showed some limitations when deployed on current technology. The specific design of the used cellphone and the speed of cryptographic operations are major concerns. While the design can be changed and optimised, it remains unclear if the full potential of the cryptographic coprocessors can be used in future devices.

6. REFERENCES

[1] Chase's Blink. http://www.chaseblink.com.

[2] Chipknip: An offline smartcard payment system. http://www.chipknip.nl.

[3] Geldkarte: An offline smartcard payment system. http://www.geldkarte.de.

[4] Globalplatform C library and command shell. http://sourceforge.net/projects/globalplatform.

[5] Java card platform specification 2.2.2. http://java.sun.com/javacard/specs.html.

[6] MasterCard PayPass. http://www.paypass.com/.

[7] NFC payment trials. http://www.contactlessnews.com/tag/Payment.

[8] PROTON: An offline smartcard payment system. http://www.banksys.be.

[9] StolPaN – Store Logistics and Payment with NFC. http://www.stolpan.com.

[10] Visa payWave. http://usa.visa.com/personal/cards/paywave/index.html.

[11] GSM Association. Mobile NFC Sevices. http://www.gsmworld.com/newsroom/document-library/technical_documents.htm, 2007.

[12] GlobalPlatform. GolbalPlatform card specification, version 2.1.1. http://globalplatform.org, mar 2003.

[13] Adam Gowdiak. J2ME security vulnerabilities 2008. http://www.security-explorations.com/n2srp.htm.

[14] International Organisation for Standardisation. *ISO/IEC 18092-4. Information technology – Telecommunications and information exchange between systems – Near Field Communication – Interface and Protocol (NFCIP-1)*, 2007.

[15] International Organisation for Standardisation. *ISO/IEC 14443-1. Identification cards – Contactless integrated circuit(s) cards – Proximity cards –* , 2008.

[16] Y. Matsuoka, Patrick Schaumont, K. Tiri, and Ingrid Verbauwhede. Java cryptography on kvm and its performance and security optimization using hw/sw co-design techniques. In *Proc. Int. Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES 2004)*, pages 303–311, 2004.

[17] Nokia. Nokia 6131 NFC technical product description.

[18] Nokia. Nokia 6212 classic, specifications.

[19] SUN. Java card technology development kit. http://java.sun.com/javacard/devkit/.