Practical Experiences with NFC Security on mobile Phones

Gauthier Van Damme and Karel Wouters [⋆]

Katholieke Universiteit Leuven
Dept. Electrical Engineering-ESAT/SCD/IBBT-COSIC
Kasteelpark Arenberg 10, 3001 Heverlee-Leuven
{gauthier.vandamme,karel.wouters}@esat.kuleuven.be

Abstract. In this paper we present our practical experiences in implementing a secure NFC application on mobile phones. First, we present the characteristics of the NFC technology and its security features. Based on our practical, real-world offline NFC voucher payment application, using the Nokia 6313 and 6212 NFC enabled devices, we illustrate the possibilities of NFC applications on today's technology with an emphasis on the security aspects. In our experience, the current technology is not sufficient to provide for a completely secured system, which resulted in sub-optimal speed of our implementation. As the security protocols in our solution are PKI-based, we include some timings of the underlying cryptographic routines, to show that the actual slowdowns are not only caused by the heavy use of cryptography, but also by design decisions of the mobile phone manufacturer.

Key words: NFC, security, mobile payment, Nokia 6313

1 Introduction

Near Field Communication (NFC) is a short-range wireless communication technology standard that extends previous standards. It incorporates the ISO 14443 Type A and Type B standard for RFID technology [23] used by the NXP Mifare contactless cards [5], the ISO 15693 standard for Vicinity Cards [20] used in item management and FeliCa [1], the contactless card system from Sony that was later standardised to ISO 18092. The combination of all these standards into the NFC standard makes NFC devices backward compatible to existing RFID contactless cards, tags and infrastructures.

Furthermore the NFC technology makes it possible to combine the interface of a reader and a smart card in a single device. A NFC enabled mobile phone can then for example easily switch between both passive tag or active reader mode.

[⋆] 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 and by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy).

In the next sections the NFC standard and its security aspects will be discussed.

1.1 The NFC Standards

Near Field Communication technology is based on inductive coupled devices operating at the radio frequency ISM band of 13.56 MHz. The communication range of NFC devices is approximately 10 cm. The NFC interface and communication protocol between such closely coupled devices have been standardised in ECMA-340 [13] and ISO/IEC 18092 [21] called Near Field Communication Interface and Protocol-1 (NFCIP-1). In this standard three different bit rates are defined: 106, 212 or 424 kbit/s. Depending on the bit rate, different modulation and encoding schemes are used.

The standard also defines the communication modes. NFC devices can communicate in a passive or in an active way. Passive mode is equivalent to normal RFID type communications where one device acts as a reader (initiator) and the other as a passive tag (target). The initiator powers the target, generating a RF field. The target then responds by modulating the existing field. Active mode means that both devices should be internally powered and alternately generate their own inducted field, closing it when waiting for a response. This mode enables bit rates higher than the ones defined before (up to 6.78 Mbit/s).

Later, the NFCIP-1 standard was expanded to ECMA-352 [12] and ISO/IEC 21481 [22]: Near Field Communication Interface and Protocol-2 (NFCIP-2), integrating the NFCIP-1, the ISO 14443 and ISO 15693 standards into one common standard. This global standard specifies the mechanism to detect and select one communication type out of the three possible communication types defined in its three sub-standards. This way a NFCIP-2 compliant device is compatible with all existing radio-frequency (RF) devices communicating at 13.56 MHz.

These standards have led to three basic modes of operation for NFC devices:

– **Card Emulation Mode**: The NFC device acts like a normal passive contactless card, emulating a smart card. The device is passive so it does not generate a RF field.
– **Reader/Writer Mode**: The NFC device acts like a normal active contactless card reader. It can then generate RF fields to communicate with contactless cards, RFID tags or NFC Forum tags[1].
– **Peer to Peer Mode**: Two NFC devices can communicate together in both active or passive NFC mode. According to the master/slave principle, the initiator or master initiates a data transfer and waits for the target or slave to respond.

[1] The NFC Forum [9], a non-profit industry association started to promote the use of NFC, defined a common data format called NDEF to store different types of data on all kinds of tags. Tags compatible with this data format are called NFC Forum tags.

1.2 Security Issues of the NFC Standard

In RFID technology, user privacy [30] and man-in-the-middle (MITM) attacks [16] are major concerns. By using a shorter communication range and special communication protocols, NFC should be able to overcome these problems, at least partially. Compared to some RFID standards or even Bluetooth technology that have a much larger communication range, sniffing communications and MITM attacks on NFC communications are much harder to set up in practise.

As explained in [18], different types of attacks can be executed on the way NFCIP-1 compliant devices communicate:

Eavesdropping Eavesdropping or sniffing is the most obvious attack on wireless communications. Using an antenna and analysis equipment an attacker can listen to any data being sent between devices [17]. Due to the close proximity and therefore low power RF field of communicating NFC devices, it can be stated that NFC communications are harder to eavesdrop than other technologies with bigger wireless ranges. How hard it exactly is to eavesdrop a NFC conversation unfortunately depends on too many physical (RF signal emitted, antennas used, receiver used) and environmental (location, noise, position of the attacker) factors. There is also a difference between listening to a device in active or in passive mode. The passive device is much harder to listen to because the data is sent by using inductive coupling on the field generated by the active device. According to [18] a rule of thumb of 10 m for eavesdropping active devices and 1 m for passive devices can be used for communications based on the NFCIP-1 standard. The same rule can be used for ISO 14443 communications that are closely related to the NFCIP-1 standard. For communications based on the ISO 15693 standard eavesdropping will probably be possible over bigger distances because the communication range defined in the standard is bigger as well.

Data Modification Denial of service attacks using a jammer are easy to perform and almost impossible to prevent. The only thing a NFC device can do against such attacks is detect them. However, an attacker is interested in more advanced and useful attacks. The attacker will try to modify the data being sent out. In active mode the 106 kbit/s transfer rate defined by the NFCIP-1 standard uses 100% Amplitude Shift Keying (ASK) in combination with modified Miller coding. For higher bit rates 10% ASK in combination with Manchester coding is used. Both are represented in figure 1. As can be seen on the figure, the modified Miller code has the particularity that when a ‘10’ combination is send, the signal corresponding to the ‘0’ does not contain a pause, while for a ‘00’ combination both zero bit signals would contain a pause. In passive mode, response data is encoded using Manchester coding with 10% weak load modulation.

As described in [18], in the 106 kbit/s coding, it is ‘easy’ for an attacker to change a pause into a signal but impossible to change a signal into a pause. This is because adding a signal where there is no signal is easy but removing a signal by perfectly aligning the same signal on to it is practically impossible. In figure 1 one can then see that only changing a combination of ‘11’ into ‘10’ is feasible.

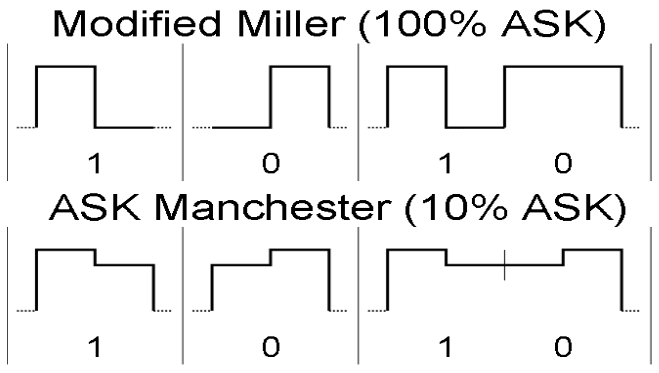

Fig. 1. The two coding schemes

For the higher bit rates 10% ASK means the signal level is either 82% or 100% of the chosen amplitude [21]. The decoder will measure both signal levels and compare them. In [18] the authors state that if the attacker adds up a signal to the 82% signal level he could trick the decoder in thinking the newly resulting signal is now the full amplitude signal and the old full amplitude signal is now the 82% signal, thus enabling him to change a '0' bit to '1' bit or vice versa.

To conclude, data modification in a NFCIP-1 communication is possible for bit rates higher than 106 kbit/s and possible only to some small extend for the 106 kbit/s bit rate. Similar conclusions can be drawn for ISO 14443 and ISO 15693 communications even if the latter uses different bit rates and encoding schemes.

Man-in-the-middle In MITM attacks, two parties are tricked into thinking they are communicating securely with each other, while the attacker actually sits in between them, communicating with both. The attacker catches messages sent by one device and then sets up a new communication with the second device. Then, he catches the response of the second device and sends his own response to the first device. To be sure both devices don't see the attacker, he needs to catch the data coming from one party while making sure the other party does not receive it. The attacker thus has to jam the signal from the first party while at the same time sending its own data out. The problem here is that NFC devices are able to receive and transmit data at the same time. Thus, they can check the radio frequency field and detect a collision if the received signal does not match with the transmitted signal and thus detect jamming or incoherent signals. Add to this the fact that NFC is a short range communication technology and MITM type of attacks are practically impossible to mount.

We can conclude that even if the NFC standard foresees some features that makes the attacker's life harder, perfect security can only be obtained when dedicated cryptography is used to establish a secure channel between communicating devices.

1.3 NFC Applications

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. Furthermore they enable NFC technology to be cheaply combined with a display, a keyboard and with all types of communication channels like the Internet or Bluetooth. NFC enabled phones, being able to run multiple applications, are able to 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.

Today a large amount of NFC trials is being conducted [7] and experts foresee that NFC mobile phones will eventually replace physical wallets, thereby improving the user convenience [2]. Obviously, for any NFC based payment system and especially an offline one, the basic security provided by NFC technology and discussed in the previous part is not sufficient.

In this paper we present our practical experiences in developing a specific secure offline payment application on NFC enabled mobile phones. Using the available Nokia NFC phones (Nokia 6131 [25] and Nokia 6212 [26]) we show today's cryptographic possibilities and limitations in real-life NFC applications. In the next section, the application and its security features will be discussed in more detail. Then we touch upon the obstacles we encountered when implementing this application. We conclude with our view on the state of practical security in current NFC technology.

2 A Secure NFC Offline Payment Application

Today, commercial smart card based electronic wallets suffer from their lack of usability. Users can not check their balance anytime anywhere, nor can they easily transfer money from one user to another without having to connect to a central server. It is obvious that the former can be solved by putting the wallet securely on a phone with display. Solving the latter on the other hand is a lot more complicated. In this paper we invest the feasibility of a secure, NFC based, offline payment application using digital vouchers on current NFC enabled phones that has both these properties and could therefore considerably improve user acceptance of electronic wallets. The users of the developed system have digital vouchers stored on their phones and can use them for payment at a cash terminal or they can transfer them to other users through NFC. For such a system, 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 on the mobile phone security components. Also, in payment schemes in which value can be transfered offline, there is no main server that is able to control and trace back all transactions. This creates the risk that digital vouchers might get lost in transaction. Therefore, copying and loss were the main risks to protect against. To achieve this, a circle of trust had to be constructed, which will be briefly described in the section below. A second section will then explain in more detail the cryptography used for this circle of trust.

2.1 The Application Backbone

NFC enabled phones use a secured piece of hardware called the Secure Element (SE). The SE is connected to the NFC module so it can be accessed in the passive NFC card emulation mode. In the Nokia phones we used, this SE is located in the phone but in the future it could be located in the SIM-card of the user [27]. The SE can only accept new software from a Trusted Services Manager (TSM), who holds the private key to authenticate to the SE. In our application we use this SE to host security-critical parts of the digital voucher application on the phone. Because of the TSM, we assume that this part of the application can be deployed securely, and can be fully trusted to handle valuable data such as digital vouchers. A MIDlet [6] running on the less protected Nokia phone Operating System S40 [15] has then limited access to this part of the application.

Knowing this we can then deploy a Public Key Infrastructure (PKI) using X.509 certificates for authentication to interconnect the voucher issuer and all users. The deployed PKI is also used to create secure channels for both voucher payments and voucher transactions. Furthermore, it enables to securely sign and verify vouchers to prevent attackers from generating fake digital vouchers.

2.2 Cryptographic components

To use the PKI to create a secure channel in every transfer, we use the cryptographic algorithms provided by the SE. In Figure 2 the phone-to-phone voucher transfer protocol, a three step handshaking protocol, is shown. The cryptographic algorithms used in this protocol are the same as in the payment protocol so only the phone-to-phone voucher transfer protocol is considered here. This protocol is also the most important one from a practical perspective because it enables secure voucher transfers without a trusted party being involved. More information on the protocols can be found in [11].

The SE on the Nokia NFC phones runs on Giesecke & Devrient's Sm@rtCafé Expert 3.1 operating system and consists of a Mifare 4K area and a Java Card which is compliant to Global Platform 2.1.1 [14] and Java Card 2.2.1 [4]. The system actually runs on a Philips/NXP SmartMX P5CN072 Secure Dual Interface PKI Smart Card Controller [8]. There is 65 KB memory space available, which is used to store applications (applets). The java card in the SE supports several common cipher, signature and message digest algorithms, like DES3, RSA and SHA-1. These algorithms where sufficient for our approach, but to further decrease computation time and public key and voucher size, AES and ECDSA

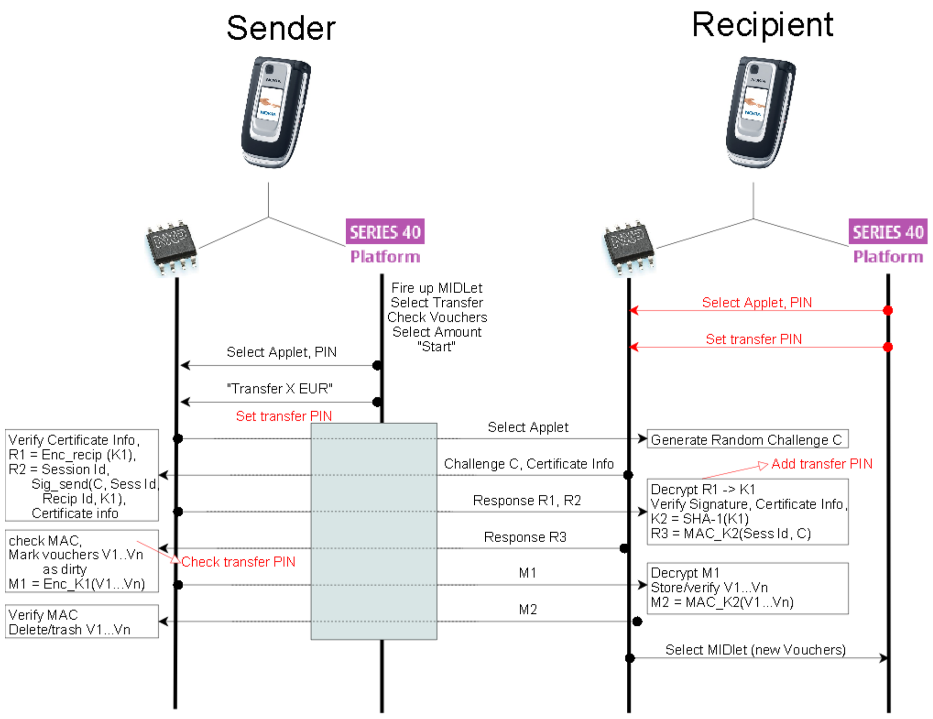

Fig. 2. The phone-to-phone protocol

would have been desirable, as these are more suitable for a limited environment. Regrettably, they are not supported on the SE of the Nokia phones.

More specifically, the cryptographic components we used are:

- A secure random number generator to generate challenges and session keys.
- A public key signature and encryption algorithm to execute the public key pair related actions. As said, we had to use RSA for this even if ECDSA was our first choice. More specifically for signatures, SHA-1 was used in combination with RSA.
- Public key pairs (1024 bit) for certificate, voucher and session specific data verification as well as for session key encryption and session data signature creation. The private RSA keys used are stored in their Chinese Remainder Theory (CRT) form to improve decryption and signature speed.
- The SHA-1 hash function to extract a new secure session key from a secure random session key.
- A symmetric key encryption algorithm to encrypt and decrypt data using a secure session key. We used DES3 in CBC mode with a random Initialisation Value (IV) for this.
- A Message Authentication Function (MAC) or key dependent hash to be able to proof session key knowledge and correct voucher reception to the other party. For this we use DES3 in CBC mode where we take the eight most significant bytes as MAC value.

In the next part we will expand on the obstacles we encountered during the implementation of our application and the limitations of the current NFC environment for a secure application like the one presented above.

3 Practical experiences

In the offline voucher payment application we presented in the previous part, achieving the best possible security was one of the main goals. In contrast to other existing NFC applications, no compromise was made in trading security for improved speed or smaller memory usage. After implementation we can state that this zero tolerance security perspective causes some implementation issues and drawbacks on the final application characteristics. First the problems related to the implementation of the voucher payment application are discussed. Then the final application characteristics are presented.

3.1 Implementation Problems

The main problem when using a java card is the limited freedom Java Card 2.2.1 developers have. First of all only the types `short` and `byte` are defined. Moreover only one dimensional arrays of these types can be instantiated. And although Java Card 2.2.1 has a variety of different libraries defined, the impossibility to create your own and the shortcoming of some specific libraries makes efficient programming and code reuse very difficult. In our application for example a

X.509 certificate parser as defined in [19] to verify certificates would have been very useful, and would have reduced implementation time while improving code efficiency. Unfortunately there is no library yet available to do certificate parsing. Another example is when using the signature library, some unsafe padding schemes are still supported while more secure ones are not. In our application we had to use the insecure PKCS#1 v1.5 padding scheme [10] while it should have been the newer 2.0 version. A last example is the choice of the hash function. Next to the broken hash function MD5 [29] and the weak SHA-1 [28], the java card libraries also supports the secure RIPE MD-160 hash function. Unfortunately the Java standard libraries don't. This means we should use external libraries to make it possible to use RIPE MD-160 for payments at terminal site if standard Java is used there.

Another problem using a java card for our application is the problem of time. As every voucher has an expiration date, we should be able to use a trusted clock to check whether a voucher is expired or not. For now this is not possible on java cards and the only possible way to check time is when a trusted third party is present (e.g. during a payment at a terminal).

A limitation when using current java cards is also the maximum APDU [2] 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 which indicates if data is still pending or not. This decreases the transaction speed and raises the code complexity.

Our hope is that most of these problems will be solved in later Java Card versions so to bring java card programming closer to standard object oriented Java programming. Some of the missing features such as the impossibility to verify certificates, the absence of absolute time and the inefficient java card communication means will already be solved in Java Card 3.0 connected edition [3].

3.2 Implementation Results

When interaction between two phones' SEs is necessary, e.g. to transfer vouchers, the SE elements of both phones have to communicate. A SE being a passive device, some software running on the operating system of one of the phones is necessary between the two SEs. This software, in our case a MIDlet on the S40 system, is represented in Figure 2 as a grey box and operates as a APDU router. But as said before, the S40 operating system is far from being secure [15] so malware can be running on it. This brings us to the conclusion that even if MITM attacks are impossible to perform in between direct NFC communications, the possibility of malware makes MITM attacks possible when two SEs are communicating through the S40 Nokia OS.

The observation above lead us to develop a protocol to establish a secure channel between the two SEs, in which no valuable data leaves the SE unencrypted. Unfortunately the use of heavy cryptography and more specifically

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

RSA signatures and encryptions is very time-consuming. In table 1 we present timings for different operations such as RSA encryptions and signatures. As the SE is the same in both the Nokia 6313 and 6212, the java card timings are the same as well. In the table we can see that RSA operations are very expensive to perform. As we need them frequently in our PKI based protocol these operations will significantly decrease transaction speed. On the other hand, other operations like symmetric encryption using DES3 or SHA-1 hashing are are also not as fast as one would expect them to be. The P5CN072 Philips/NXP chip used in the SE has both a co-processor for DES3 as for PKI operations and should be able to compute a DES3 encryption in 0,05 ms and a 2048 bit modular exponentiation in less then 35 ms [8]. Compared to the actual results in table 1, this shows that there is a big overhead in translating high level operations onto the low level microcontroller. First the operations have to be translated from the Java Card environment onto the Java Virtual Machine (JVM), then from the JVM onto the OS and finally from the OS onto the actual hardware. A straightforward way of improving this in the future is by allowing applications for the SE or smart card to be written in native code such as C or assembly, so to avoid this top down code translation overhead. Another way to improve performance of these critical calculations is to have more efficient hardware co-processors, optimized for example for the faster AES and ECDSA algorithms. Both these solutions have already been considered for the Java 2 Micro Edition (J2ME), resulting in a significant performance gain [24], and could easily be extended to smart cards.

Operation	Data Length	Time
Public Key Encryption	100 byte	98.8 ms
Private CRT Key Signature	100 byte	287.9 ms
Triple DES in CBC mode	100 byte	34.3 ms
SHA-1 Hashing	100 byte	29.5 ms

Table 1. Timing of different Java Card operations.

Another issue related to the use of software on the Nokia NFC phones to route APDUs is caused specifically by the way these phones implement their interface to SEs. As can be seen in Figure 2 the routing software at sender side has to switch continuously between an external NFC connection with the receiver SE and an internal connection with his own SE. As on the current Nokia phones these two NFC connections can not be open at the same time, at every step the current connection needs to be closed and a new connection has to be opened. We know that opening an external NFC connection is a time consuming task due to the fact that at sender side a Target Listener has to constantly poll for external connections. In table 2 one can see that opening an external connection, closing it and then opening an internal connection can take approximately 0.68 to 0.48 second on the Nokia 6131 phones when phones are hold close together. The results get far worse if like in real life the phones are not necessarily hold close without moving. The same operation can then vary between 0.8 and 1 second when the phones are hold approximately 0.5 cm from each other. On the Nokia 6212 phones where the antenna is not focused on a specific point (it sits all around the phone) even worse results are obtained because the phones do not always respond immediately to each other when brought into close range. For those phones timings can then vary from 1.62 up to 2.74 seconds. As in our protocol three such connections are needed, the time consumed by this operation is non-negligible. A first solution to this overhead would be to make it possible to have both an internal and external connection open at the same time.

Phone	External Phone Status	Time
Nokia 6313	Off and zero distance	0.46 s
Nokia 6313	On and zero distance	0.68 s
Nokia 6313	On and 0,5 cm distance	0.8 - 1,0 s
Nokia 6212	On and zero distance	1.62 - 2.74 s

Table 2. Timings for opening a connection to an external SE, followed by opening a connection to the internal SE.

A final issue with our implementation is memory-related: The java card in the SE has 65 KB of memory that should be able to contain different types of applications (payment, ticketing, a.s.f.). Unfortunately our application with 20 vouchers and the necessary certificates already requires about 30 KB of memory. This means that for different secure NFC applications to run on the same phone, a smart card with larger memory is necessary.

4 Conclusion

To conclude we can state that today's NFC technology in mobile phones can already be used for secure applications but still suffers some drawbacks that are show-stoppers for commercial use. The heavy cryptography and the relative immaturity of available NFC phones, makes for example offline payments on these phones a time consuming task. A complete transaction with 10 vouchers from one phone to another takes about eight seconds. Increasing the number of vouchers, from 10 to 20 only slightly increases the transaction time. This confirms our claim that, independently of the number of vouchers send, there is a large constant communication overhead. In real life situation eight seconds per transaction is far too slow so a secure offline NFC voucher payment system can not be brought to the consumer yet.

On the bright side, the availability of a SE in which only a Trusted Services Manager can upload software and with controlled access for the phone users is already a big step towards more mobile security. By shifting security from unsafe platforms such as the S40 OS towards a more secure hardware module like the java card used by Nokia, the user and developer trust can only be improved, opening the path to newer and better NFC enabled devices and systems.

References

1. FeliCa contactless IC card system. `http://www.sony.net/Products/felica/`.
2. Japans Year of the FeliCa eWallet Phone. `http://wirelesswatch.jp/2005/11/11/2006-japans-year-of-the-felica-ewallet-phone/`.
3. Java Card 3.0 Platform Specification. `http://java.sun.com/javacard/3.0/specs.jsp`.
4. Java card platform specification 2.2.2. `http://java.sun.com/javacard/specs.html`.
5. MIFARE contactless smart card from NXP. `http://www.nxp.com/products/identification/mifare/`.
6. Mobile Information Device Profile (MIDP). `http://java.sun.com/products/midp/`.
7. NFC payment trials. `http://www.contactlessnews.com/tag/Payment`.
8. P5CN072 Secure Dual Interface PKI Smart Card Controller. `http://www.nxp.com/acrobat_download/other/identification/SFS107710.pdf`.
9. The Near Field Communication (NFC) Forum. `http://www.nfc-forum.org/`.
10. Jean-Sébastien Coron, Marc Joye, David Naccache, and Pascal Paillier. New Attacks on PKCS#1 v1.5 Encryption. In *EUROCRYPT*, pages 369–381, 2000.
11. Gauthier Van Damme, Karel Wouters, Hakan Karahan, and Bart Preneel. Offline NFC Payments with Electronic Vouchers. In *ACM SIGCOMM Workshop on Networking, Systems, Applications on Mobile Handhelds MobiHeld 2009*, aug 2009.
12. ECMA. *ECMA-352: Near Field Communication Interface and Protocol-2 (NFCIP-2)*. pub-ECMA, pub-ECMA, dec 2003.
13. ECMA. *ECMA-340: Near Field Communication - Interface and Protocol (NFCIP-1)*. pub-ECMA, pub-ECMA, dec 2004.
14. GlobalPlatform. GolbalPlatform card specification, version 2.1.1. `http://globalplatform.org`, mar 2003.
15. Adam Gowdiak. J2ME security vulnerabilities 2008. `http://www.security-explorations.com/n2srp.htm`.
16. Gerhard Hancke. A practical relay attack on ISO 14443 proximity cards. Technical report, University of Cambridge, 2005.
17. Gerhard Hancke. Eavesdropping attacks on High-Frequency RFID tokens. In *Workshop on RFID Security RFIDSec 08*, jul 2008.
18. Ernst Haselsteiner and Klemens Breitfuss. Security in near field communication (NFC), Philips Semiconductors. In *Workshop on RFID Security RFIDSec 06*, jul 2006.
19. Olaf Henniger, Karim Lafouand, Dirk Scheuermann, and Bruno Struif. Verifying X.509 certificates on smart cards. In *International Conference on Computer and Information Science and Engineering (CISE)*, 2006.
20. International Organisation for Standardisation. *ISO/IEC 15693-1. Identification cards – Contactless integrated circuit(s) cards – Vicinity cards –* , 2000.
21. International Organisation for Standardisation. *ISO/IEC 18092-4. Information technology – Telecommunications and information exchange between systems – Near Field Communication – Interface and Protocol (NFCIP-1)*, 2004.

22. International Organisation for Standardisation. *ISO/IEC 21481. Information technology – Telecommunications and information exchange between systems – Near Field Communication Interface and Protocol -2 (NFCIP-2)*, 2005.
23. International Organisation for Standardisation. *ISO/IEC 14443-1. Identification cards – Contactless integrated circuit(s) cards – Proximity cards –* , 2008.
24. Yusuke Matsuoka, Patrick Schaumont, Kris Tiri, and Ingrid Verbauwhede. Java cryptography on kvm and its performance and security optimization using hw/sw co-design techniques. In *CASES '04: Proceedings of the 2004 international conference on Compilers, architecture, and synthesis for embedded systems*, pages 303–311. ACM, 2004.
25. Nokia. Nokia 6131 NFC technical product description.
26. Nokia. Nokia 6212 classic, specifications.
27. Marie Rveilhac and Marc Pasquet. Promising Secure Element Alternatives for NFC Technology. In *1st International Workshop on Near Field Communication - NFC 09*, feb 2009.
28. Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding Collisions in the Full SHA-1. In *In Proceedings of Crypto*, pages 17–36. Springer, 2005.
29. Xiaoyun Wang and Hongbo Yu. How to break MD5 and other hash functions. In *In EUROCRYPT*. Springer-Verlag, 2005.
30. Stephen A. Weis, Sanjay E. Sarma, Ronald L. Rivest, and Daniel W. Engels. Security and privacy aspects of low-cost radio frequency identification systems. In *Security in Pervasive Computing*, volume 2802 of *Lecture Notes in Computer Science*, pages 201–212, 2004.