Software Card Emulation in NFC-enabled Mobile Phones: Great Advantage or Security Nightmare?

Michael Roland
NFC Research Lab Hagenberg
University of Applied Sciences Upper Austria
Softwarepark 11, 4232 Hagenberg/Austria
michael.roland@fh-hagenberg.at

ABSTRACT

Software card emulation is a new approch to advance the interoperability of NFC with legacy contactless smartcard systems. It has been first introduced to NFC-enabled mobile phones by Research In Motion (RIM) on their BlackBerry platform. Software card emulation aims at opening and simplifying the complex and tightly controlled card emulation functionality. While this form of card emulation, that gets rid of the secure element (a device tightly controlled by the "big players"), is a great chance for development of innovative NFC applications, it potentially makes card emulation less secure and paves the way for interesting attack scenarios. This paper evaluates the advantages and disadvantages of software card emulation based on existing application scenarios and recent research results.

1. INTRODUCTION

With the emergence of Near Field Communication (NFC), more and more NFC devices and applications hit the market. However, the full potential of NFC is not available to all developers. Specifically, the secure element (SE), a smartcard microchip that is used to perform secure card emulation, is kept under tight control of device manufacturers and mobile network operators. Nevertheless, card emulation is required for interaction with many legacy RFID systems that are currently used for access control, ticketing and payment. On an NFC device, the secure element is used to store security critical applications like credit cards, access control credentials and public transport tickets. Through the device's NFC controller, the secure element can be accessed as if it were a regular contactless smartcard.

Especially the payment sector – the part of NFC that seems to generate the highest revenue – focuses on using the secure element for payment applications. Therefore, a lot of companies want to have access to secure elements to claim a share of that revenue. As a result, many developers call for easier access to card emulation capabilities.

An approach started by Research In Motion (RIM) on their BlackBerry platform is software card emulation (also known as "soft-SE" [7]). This mode allows interaction with legacy RFID reader infrastructures through applications on the mobile phone's application processor without using a secure element. At first glance this mode seems to be a great new feature for NFC devices. It opens up the – previously – tightly controlled world of card emulation to a wide range of developers. This will certainly lead to a number of new and innovative NFC applications. This increase in use-cases may, in turn, lead to an increased need for NFC devices and, consequently, help NFC to finally kick off as a mass-market technology. Besides software card emulation's benefits, there are, however, several downsides that come along with this new approach.

This paper starts with an introduction to NFC technology and its operating modes. The various types of card emulation and their availability in current NFC devices are explained. Based on existing application scenarios for the card emulation mode and based on recent research results, the advantages and disadvantages of software card emulation are evaluated.

2. NEAR FIELD COMMUNICATION

Near Field Communication (NFC) is a contactless communication technology first standardized by Ecma (ECMA-340, ECMA-352) and later adopted by ISO/IEC (ISO/IEC 18092, ISO/IEC 21481). It is an advancement of inductively coupled proximity Radio Frequency Identification (RFID) technology and smartcard technology. NFC is compatible to legacy contactless smartcard systems based on the standards ISO/IEC 14443 and FeliCa (JIS X 6319-4). Recent standardization activities aim at also adding compatibility to ISO/IEC 15693 vicinity coupling systems. Besides standardization through normative bodies like ISO/IEC and Ecma, further specification of data formats, protocols, interoperability requirements, device certification and NFC applications is driven by the NFC Forum[1].

A basic principle of the NFC technology is "*it's all in a touch*" [4]. This means that simply touching an object or an NFC device with another NFC device immediately triggers an action. Objects can be equipped with so-called NFC tags (simple contactless memory chips based on existing RFID transponders). These tags are used to store content like Internet addresses (URLs), telephone numbers, text messages (SMS) or electronic business cards. The user can access the information on a tag by simply touching it with an NFC device.

NFC has three operating modes: peer-to-peer mode, reader/writer mode and card emulation mode:

- Peer-to-peer mode is an operating mode specific to

[1] http://www.nfc-forum.org/

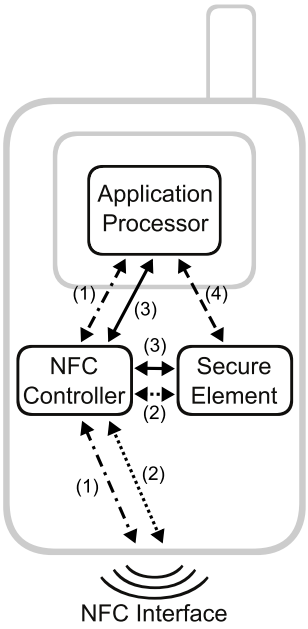

Figure 1: The paths of NFC data through the chipset of an NFC-enabled mobile phone.

NFC and allows two NFC devices to communicate directly with each other. For example, this mode can be used to exchange business cards between two NFC-enabled mobile phones or to exchange credentials for a Bluetooth or WiFi link.

- In reader/writer mode, NFC devices can access contactless smartcards, RFID transponders and NFC tags. Thus, this mode makes NFC devices compatible to existing contactless tokens.
- In card emulation mode, an NFC device emulates a contactless smartcard and, thus, is able to communicate with existing RFID readers.

Fig. 1 shows the various paths of NFC data trough the chipset of an NFC-enabled mobile phone. The application processor is the mobile phone's main processing unit. The NFC controller is the core component of the NFC functionality in a device. It contains an NFC modem and performs preprocessing of commands and data. The secure element is a smartcard microchip that is capable of performing secure card emulation. Path (1) routes commands and data between the application processor and the NFC interface. This path is used for peer-to-peer mode, reader/writer mode and software card emulation. Path (2) routes commands and data between the secure element and the NFC interface. This path is used for secure card emulation. In addition to external access through the NFC interface, the secure element is connected to the application processor. That way, content on the secure element can be managed from within the phone and through the cellular network. The secure element can be connected to the application processor either directly (4) or through the NFC controller (3). Typically, access to the secure element through pathes (2) and (3) is mutually exclusive and, therefore, those pathes cannot be active at the same time.

2.1 Card Emulation

There exist several possible options for NFC's card emulation mode. Emulation can differ in communication standards, in supported protocol layers, in supported command sets and in the part of the NFC device that performs the actual emulation.

Regarding the communication standard, there exist three possibilities: ISO/IEC 14443 Type A, ISO/IEC 14443 Type B and FeliCa (JIS X 6319-4). Support for either of these modes depends on the NFC controller, the secure element and typically the geographic region. For example, ISO/IEC 14443 Type A and Type B are the prevalent technologies in Europe while FeliCa is widespread in Japan.

Another difference is the part of the device that performs the actual emulation. On the one hand, a card can be emulated in software (on the device's application processor). On the other hand, card emulation can be performed by a dedicated smartcard chip – the secure element.

2.2 Secure Element

A secure element can be a dedicated microchip that is embedded into the NFC device. Such a chip could also be combined in a single package with the NFC controller. Another possibility is the integration of the secure element functionality in another smartcard/security device that is used within the NFC device. Such a combined chip can be the UICC (universal integrated circuit card; often referred to as Subscriber Identity Module/SIM card) or an SD (secure digital) memory card.

Many secure elements (e.g. NXP's SmartMX) are standard smartcard ICs as used for contact and contactless smartcards. They share the same hardware and software platforms. The only difference is the interface they provide: Instead of (or in addition to) a classic smartcard interface according to ISO/IEC 7816-3 (for contact cards) or an antenna (for contactless cards), the secure element has a direct interface for the connection to the NFC controller (e.g. NFC Wired Interface (NFC-WI) or Single Wire Protocol (SWP)).

Secure elements feature the same high security standards as regular smartcards. A secure element provides secure storage, a secure execution environment and hardware-based support for cryptographic operations. Secure element chips are protected against various attacks that aim at retrieval or manipulation of stored data and processed operations.

Smartcard chips, their operating systems and the design processes are evaluated and certified according to high security standards. Examples for such standards are the Common Criteria protection profiles for smartcard microchips[2]. Thus, the secure element fulfills the requirements necessary for security critical applications like payment and access control.

A major unsolved security issue with smartcards is the re-

[2]E.g. "Smartcard IC Platform Protection Profile", Version 1.0, July 2001; "Protection Profile Smart Card IC with Multi-Application Secure Platform", Version 2.0, November 2000; "Java Card™ System Protection Profile Collection", Version 1.0b, August 2003; "Java Card™ System Protection Profile Open Configuration", Version 2.6, April 2010.

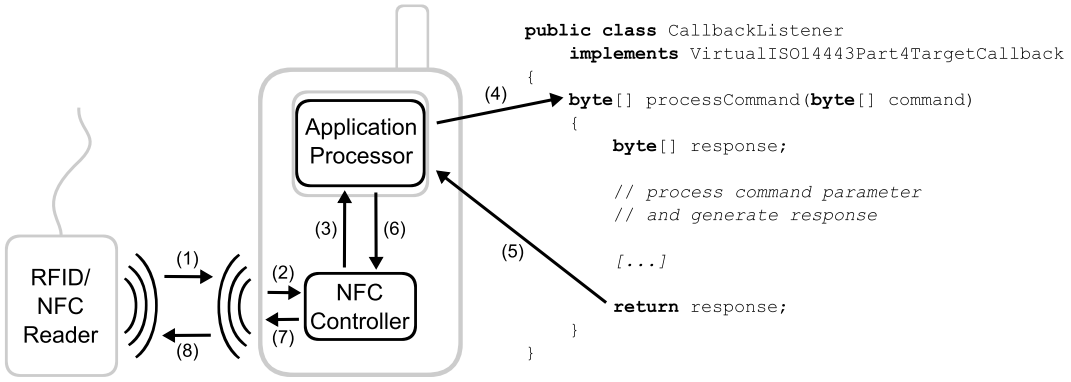

Figure 2: Command flow of software card emulation on BlackBerry 7 platform.

lay attack scenario. "Relay attack" refers to a scenario where the communication with a smartcard is relayed over longer distances through an alternative carrier. An attacker could abuse this to use a victims smartcard in remote places. The viability of relay attacks on contactless smartcards was first proven by Hancke [8] by relaying the signaling layer communication between a smartcard and an RFID reader. Kfir and Wool [11] could significantly improve communication distances for this scenario, which allows for easier access to smartcards without the victims knowledge. Roland et al. [14, 15] revealed that secure elements are not only prone to attacks through the contactless interface but relay attacks can even be performed through software on a mobile phone's application processor. Anderson [1] suggests that NFC-enabled mobile phones are an ideal platform for performing relay attacks on contactless smartcards. Francis et al. [6, 7] show that NFC peer-to-peer mode communication and contactless smartcard communication can be relayed via Bluetooth and other wireless communication channels by using two NFC-enabled mobile phones as attack platform.

3. SOFTWARE CARD EMULATION

Software card emulation (or "soft-SE" [7]) is a new approach to card emulation in NFC-enabled mobile phones. It was introduced to mobile phones by Research In Motion (RIM) on their BlackBerry platform. Besides support for different kinds of secure elements, the BlackBerry 7 platform introduces support for emulation of NFC tags and smartcards through software on the mobile phone's application processor [5, 13].

An application can emulate an NFC Forum type 4 tag by simply specifying an NDEF message that should be stored on the virtual tag. The type 4 tag protocol is handled automatically by the BlackBerry system. This mode could be used to exchange data with another NFC device that operates in reader/writer mode.

An application can also emulate a full ISO/IEC 14443-4 smartcard. Emulation is possible for both ISO/IEC 14443 Type A and Type B protocol variants. An application can specify static properties of the emulated smartcard (i.e. the unique identifier (UID) and the historical bytes for ISO/IEC 14443 Type A) and can exchange protocol data units on top of the block transmission protocol defined by ISO/IEC 14443-4. (While the API allows applications to freely define a UID, this feature was not implemented on existing devices due to security concerns [17].) When an application wants to act as a contactless smartcard, it registers itself with the BlackBerry system. As soon as a command is received from an external RFID/NFC reader, a callback method is executed. The received command is passed as a parameter to the callback method. The application can then process the command and supply a return value to be returned to the reader. Fig. 2 depicts the command flow for software card emulation. Arrows (1) to (4) show the flow of a command from the RFID/NFC reader to the application's registered callback method. Arrows (5) to (8) show the flow of the response generated by an application back to the RFID/NFC reader.

At the moment, BlackBerry mobile phones are the only devices known to support software card emulation. However, recent patches [18, 19] to the CyanogenMod aftermarket firmware for Android devices will enable this type of card emulation on Android devices with NXP's PN544 NFC controller.

Besides mobile phones, other devices, like certain NFC readers, can also be used to perform card emulation without a secure element. An example is the ACS ACR 122U NFC reader. There also exist dedicated card emulators (e.g. Proxmark, OpenPICC, IAIK HF RFID DemoTag).

3.1 Advantages of Software Card Emulation

Card emulation mode is said to be the most promising mode of NFC [12]. A main reason for this is the revenue expected from card emulation compared to the revenue of NFC's other operating modes. Also many existing payment, ticketing and access control applications have a stationary reader infrastructure with users carrying smartcards/contactless tokens. Thus, adding the user side (i.e. smartcard/contactless token functionality) to a mobile phone requires card emulation mode.

However, while demanded by the NFC community, card emulation and, especially, the secure element are a complicated terrain. For now, embedded secure elements are usually under control of the handset manufacturer or a trusted service manager (TSM) who operates the secure elements for them. With the UICC as a secure element, it is the mobile network operator (MNO) who controls the secure element. "*Already a*

dispute is growing over control of the embedded secure chips expected to be included in most NFC phones [...]" [3]. So the first barrier towards access to a secure element will be that the various secure elements are operated by various different parties. The second barrier will be that it seems unlikely that a secure element operator who already provides a certain service will allow a competitor's similar service on their secure elements: "*[...] it's difficult to imagine a Google Wallet and Isis wallet residing on the same phone – especially if they are anchored to the same secure element*" [2]. The third barrier will be the cost of getting an application onto the secure element. Besides rental cost for space in the secure element, applications will most likely need some form of security certification if they should coexist with other security critical applications on one secure element.

All those barriers diminish the chances for average developers (or actually for all but the big players in the mobile phone and payment sectors) to get their applications onto the secure element. As a solution to this problem, RIM introduced a software card emulation mode with their BlackBerry phones. With this mode, any developer can create applications that use card emulation mode, even without access to a secure element. This opens development of applications based on existing stationary reader infrastructures. Thus, developers can create mobile phone based applications for access control, payment, public transport ticketing and event ticketing, where RFID tickets and smartcards are already in use.

Another advantage is that software card emulation mode can be used to communicate with NFC devices that do not (yet) support a full-fledged peer-to-peer mode. For instance, the Android system only supports Android Beam for peer-to-peer communication. However, Android Beam – which is based on Google's NDEF Push Protocol (NPP) and the Simple NDEF Exchange Protocol (SNEP) – can only be used to exchange one message into one direction per touch of the two NFC mobile phones[3]. Therefore, software card emulation can be used as an easy alternative to peer-to-peer mode for communication between NFC devices. Also many existing contactless smartcard readers for the PC platform do not support peer-to-peer mode. Examples are the Reiner SCT cyberJack RFID basic (used for the new German identity card) and the HID OMNIKEY 5321. Nevertheless, these devices can communicate with mobile phones in card emulation mode. Thus, software card emulation mode opens up for an easy interaction between mobile phones and PC systems without costs for additional NFC hardware.

A further advantage of software card emulation mode compared to peer-to-peer mode is the software and driver support for the PC platform. Reader/writer support for contactless smartcards is well standardized with PC/SC and integrated into most operating systems by default. Even platforms like Java SE have standardized APIs for access to contactless smartcards. Support for peer-to-peer mode is not that well-established and is only covered by some third party libraries like libnfc[4] and libnfc-llcp[5]. Besides that, the protocol stack for NFC peer-to-peer mode,

- application layer protocol (based on NDEF messages) on top of
- NPP, SNEP (or direct use of another application layer protocol) on top of
- LLCP (NFC Logical Link Control Protocol) on top of
- NFC-DEP (NFC Data Exchange Protocol, i.e. the low-level peer-to-peer communication protocol defined by ISO/IEC 18092),

is rather complex compared to that for reader/writer mode,

- application layer protocol (based on ISO/IEC 7816-4) on top of
- ISO-DEP (i.e. the communication protocol defined by ISO/IEC 14443-4).

Overall, software card emulation mode is certainly a great chance for developers and others than the "big players" to easily extend to applications beyond simple tagging[6].

3.2 Disadvantages and Security Impacts

All these advantages come, however, at a price. Besides some technical limitations in current implementations of software card emulation, there is a significant loss in security. Applications executed on a mobile phone's application processor do not benefit from the secure data storage and the trusted execution environment of a secure element. Unless, of course, the application processor itself provides some form of trusted computing technology. Though, this is not the case with most current mobile phones.

Without secure storage, it becomes difficult for card emulation applications to store sensitive data (e.g. credentials for access control systems, private signing keys for payment solutions, tickets ...) Moreover, the lack of a trusted execution environment could allow for (intentional) interference by other applications. For instance, recent vulnerabilities of the Google Wallet allowed an attacker to recover credit card numbers, account balance, card holder information and even the wallet's PIN code, because, even though Google Wallet has access to a secure element, this data was cached within the app's private data storage in the mobile phone memory [10, 16].

Depending on the value of the sensitive data, it might be okay to take this risk. On the one hand, there is ticketing,

[3]The ISMB-NPP-JAVA project (`http://code.google.com/p/ismb-npp-java/`) shows that this limitation can be overcome by turning off the initiator's electromagnetic field between exchange of each message. However, while this is possible with dedicate NFC reader devices, many mobile phones do not allow this.

[4]libnfc (`http://www.libnfc.org/`) provides, amongst others, low-level peer-to-peer communication for a broad range of NFC readers.

[5]Part of the nfc-tools project (`http://code.google.com/p/nfc-tools/`), libnfc-llcp provides an LLCP (logical link control protocol) implementation on top of libnfc.

[6]I.e. using the NFC device in reader/writer mode together with NFC tags.

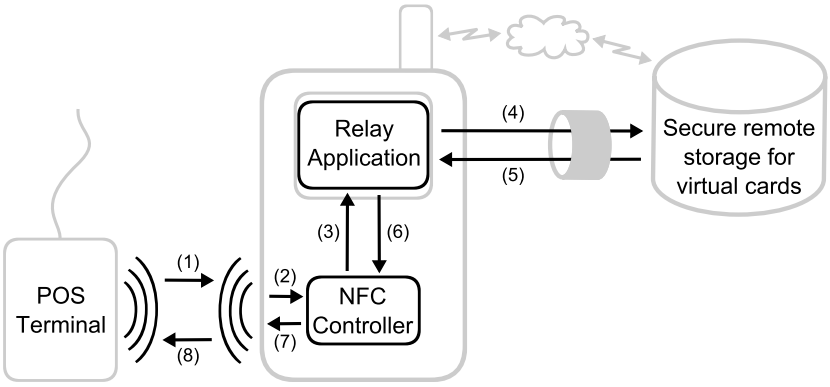

Figure 3: Credit card transaction with "virtual" card stored in a remote location.

where a single trip ticket for public transport or an entrance ticket for a theme park may be worth the risk. On the other hand, there is payment and access control. A credit card or the keys for access to a building are usually too sensitive to be stored on mobile phone memory from where they could potentially be skimmed.

Nevertheless, even with software card emulation, security critical applications are possible to some extent. The key is to store a "virtual card" in a secure remote location and use the mobile phone only as a proxy to access this card. Fig. 3 outlines this kind of online card emulation system for a virtual credit card. Commands received from the point-of-sale (POS) terminal are forwarded to a virtual credit card that is stored on a remote server. The response received from the virtual credit card is routed back to the POS terminal. Access to the virtual credit card must be secured against hijacking of ongoing communication and against unauthorized use of the credit card – e.g. by an encrypted and authenticated tunnel, and by a password entered by the user before use. Given that many mobile phones are prone to attacks which allow retrieval and manipulation of private information stored inside apps, creating a secure tunnel that cannot be hijacked by other apps can be a difficult task. Besides the security issues introduced by the secure connection to a remote service, this scenario requires a stable Internet connection during the whole transaction, which might not always be the case.

Vulnerability of data used for software card emulation applications is not the only security impact. Another problem that gains increased importance due to software card emulation is the use of mobile phones as attack platform. Research by Francis et al. [6] shows that two NFC-enabled mobile phones can be used to relay peer-to-peer mode communication over long distances. Similar relay attacks are possible with contactless smartcards (cf. Hancke [8], Kfir and Wool [11], Hancke et al. [9]). In the past, an attacker had to prepare special equipment (e.g. a card emulator that proxies smartcard signals between an RFID/NFC reader and the relay carrier.) Ready-made kits exist for this purpose, however, these kits do not feature the form factor expected for NFC and contactless transactions (i.e. a plastic card or a mobile phone). A mobile phone that supports card software emulation mode has the ideal form factor and, furthermore, has various network interfaces (Bluetooth, WiFi, GSM, UMTS ...) to establish a relay channel. Francis et al. [7] prove that a contactless smartcard can easily be relayed using two NFC-enabled mobile phones: One in reader/writer mode that acts as a proxy between the smartcard and the relay channel. The other in software card emulation mode that acts as a proxy between the relay channel and the RFID/NFC reader (e.g. point-of-sale terminal). Roland et al. [15] even demonstrate that pure software on a victim's mobile phone is sufficient to proxy communication between the secure element and the relay channel.

Further disadvantages are the technical and security related limitations of software card emulation. With NXP's NFC controllers, software-emulated smartcards are only allowed to use a certain set of UIDs (specifically only those reserved for random UIDs). For the BlackBerry platform, a possibility to specify an arbitrary UID is included into the API, but this feature has been replaced on existing devices by a randomly generated UID due to security concerns [17]. While this limitation inhibits use of software card emulation with virtually any UID-based application, it also prevents card cloning attacks on purely UID-based authentication systems (e.g. certain access control systems).

Also, BlackBerry's software card emulation mode, as well as the software card emulation that has recently been added to the CyanogenMod aftermarket firmware for Android devices, only support emulation of ISO/IEC 14443-4 smartcards. Proprietary systems that operate on lower protocol layers (like NXP's MIFARE Classic) cannot be emulated. Thus, software card emulation is not usable for several legacy RFID systems.

4. CONCLUSION

This paper evaluates the advantages and disadvantages of software card emulation. It is shown that software card emulation provides a great opportunity to developers as it breaks the barriers that exist with secure element based solutions. Software card emulation allows for easy integration of NFC-enabled mobile phones into existing contactless smartcard systems. It also has benefits over peer-to-peer mode as devices in card emulation mode can easily communicate with devices in reader/writer mode. Reader/writer mode is more established than peer-to-peer mode. These benefits come at the price of security. It becomes more difficult to protect the

content of emulated cards. Moreover, software card emulation mode turns mobile phones into an ideal platform for conducting relay attacks on smartcards and other card emulation applications. Finally, limitations placed by handset and chipset manufacturers restrict the use of software card emulation for certain applications. Yet, many applications of software card emulation could be implemented with peer-to-peer mode. However, this would often require significant upgrades to existing infrastructures.

In my opinion, software card emulation mode is not necessary for NFC devices, as most of its applications could also use peer-to-peer mode, which was specifically created for easy communication between any NFC devices. Nevertheless, software card emulation is supported by current hardware. Just because developers may be tempted to create applications without sufficient security and just because this functionality could be abused for malicious use-cases, developers should not be barred from software card emulation. Security of applications should not be achieved by restricting technology that can potentially be used for attacks but by increasing robustness against attacks. After all, as seen with the patches to CyanogenMod, completely sealing off this functionality from beeing used is impossible anyways.

5. ACKNOWLEDGMENTS

This work is part of the project "4EMOBILITY" within the EU programme "Regionale Wettbewerbsfähigkeit OÖ 2007–2013 (Regio 13)" funded by the European regional development fund (ERDF) and the Province of Upper Austria (Land Oberösterreich).

6. REFERENCES

[1] R. Anderson. Position Statement in RFID S&P Panel: RFID and the Middleman. In *Financial Cryptography and Data Security*, volume 4886/2007 of *LNCS*, pages 46–49. Springer Berlin Heidelberg, 2007.

[2] D. Balaban. Telcos Close Ranks as Google Threat Looms. *NFC Times Blog*, July 2011. `http://www.nfctimes.com/blog/dan-balaban/telcos-close-ranks-google-threat-looms`.

[3] D. Balaban. With Launch of Google Wallet, the Wallet War Begins. *NFC Times Blog*, June 2011. `http://www.nfctimes.com/blog/dan-balaban/launch-google-wallet-wallet-war-begins`.

[4] E. Chen. NFC: Short range, long potential. News Article, `http://www.assaabloyfuturelab.com/FutureLab/Templates/Page2Cols____1905.aspx`, Aug. 2007.

[5] S. Clark. RIM releases BlackBerry NFC APIs. *Near Field Communications World*, May 2011. `http://www.nfcworld.com/2011/05/31/37778/rim-releases-blackberry-nfc-apis/`.

[6] L. Francis, G. P. Hancke, K. E. Mayes, and K. Markantonakis. Practical NFC Peer-to-Peer Relay Attack Using Mobile Phones. In *Radio Frequency Identification: Security and Privacy Issues*, volume 6370/2010 of *LNCS*, pages 35–49. Springer Berlin Heidelberg, 2010.

[7] L. Francis, G. P. Hancke, K. E. Mayes, and K. Markantonakis. Practical Relay Attack on Contactless Transactions by Using NFC Mobile Phones. Cryptology ePrint Archive, Report 2011/618, 2011. `http://eprint.iacr.org/2011/618`.

[8] G. P. Hancke. A Practical Relay Attack on ISO 14443 Proximity Cards, Jan. 2005. `http://www.rfidblog.org.uk/hancke-rfidrelay.pdf`.

[9] G. P. Hancke, K. E. Mayes, and K. Markantonakis. Confidence in smart token proximity: Relay attacks revisited. *Computers & Security*, 28(7):615–627, 2009.

[10] A. Hoog. Forensic security analysis of Google Wallet. *viaForensics Mobile Security Blog*, Dec. 2011. `https://viaforensics.com/mobile-security/forensics-security-analysis-google-wallet.html`.

[11] Z. Kfir and A. Wool. Picking Virtual Pockets using Relay Attacks on Contactless Smartcard. In *Proceedings of the First International Conference on Security and Privacy for Emerging Areas in Communications Networks (SECURECOMM'05)*, pages 47–58, Sept. 2005.

[12] K. Ok, V. Coskun, M. N. Aydin, and B. Ozdenizci. Current benefits and future directions of NFC services. In *Proceedings of the 2010 International Conference on Education and Management Technology (ICEMT)*, pages 334–338, Nov. 2010.

[13] RIM. *Blackberry API 7.0.0: Package net.rim.device.api.io.nfc.emulation*, 2011. `http://www.blackberry.com/developers/docs/7.0.0api/net/rim/device/api/io/nfc/emulation/package-summary.html`.

[14] M. Roland, J. Langer, and J. Scharinger. Practical Attack Scenarios on Secure Element-enabled Mobile Devices. In *Proceedings of the Fourth International Workshop on Near Field Communication (NFC 2012)*, pages 19–24, Helsinki, Finland, Mar. 2012.

[15] M. Roland, J. Langer, and J. Scharinger. Relay Attacks on Secure Element-enabled Mobile Devices: Virtual Pickpocketing Revisited. IFIP International Information Security and Privacy Conference (pending publication), June 2012.

[16] J. Rubin. Google Wallet Security: PIN Exposure Vulnerability. *zveloBLOG*, Feb. 2012. `https://zvelo.com/blog/entry/google-wallet-security-pin-exposure-vulnerability`.

[17] M. Woolley. Response to thread "UID for NFC Mifare Tag emulation" on BlackBerry Support Community Forums. `http://supportforums.blackberry.com/t5/Java-Development/UID-for-NFC-Mifare-Tag-emulation/m-p/1575809`, Feb. 2012.

[18] D. Yeager. Added NFC Reader support for two new tag types: ISO PCD type A and ISO PCD type B. `https://github.com/CyanogenMod/android_packages_apps_Nfc/commit/d41edfd794d4d0fedd91d561114308f0d5f83878`, Jan. 2012.

[19] D. Yeager. Added NFC Reader support for two new tag types: ISO PCD type A and ISO PCD type B. `https://github.com/CyanogenMod/android_external_libnfc-nxp/commit/34f13082c2e78d1770e98b4ed61f446beeb03d88`, Jan. 2012.