Mobile smart card reader using NFC-enabled smartphones

Frank Morgner[1,2], Dominik Oepen[1], Wolf Müller[1], and Jens-Peter Redlich[1]

[1] Humboldt-Universität zu Berlin, Institut für Informatik, Lehrstuhl für Systemarchitektur, Unter den Linden 6, 10099 Berlin, Germany,
{morgner,oepen,wolfm,jpr}@informatik.hu-berlin.de
[2] Bundesdruckerei GmbH, Oranienstraße 91, 10969 Berlin, Germany,
Frank.Morgner@bundesdruckerei.de

Abstract. Due to the increasing use of electronic systems in all fields of everyday life, users are now having to deal with electronic identification and authentication practically every day. Password based authentication systems are neither secure nor particularly convenient for users. Here, we are presenting the idea of using an NFC-enabled mobile phone as a chip card reader for contactless smart cards. A mobile phone can be used to visualise, inspect and control electronic transactions. This mobile smart card reader implementation enables ubiquitous, secure and convenient two-factor authentication, the mobile phone being a very personal device which users guard carefully and with which they are particularly familiar. In this paper, we discuss the concept and implementation details of the mobile reader and present a use case for the German electronic identity card.

Key words: two-factor authentication, mobile phones, mobile identity management, USB CCID, skimming, trusted user interface

1 Introduction

Simple authentication schemes use only one single factor to authenticate the correct user: this can be a key that opens a door, a password to access an email account or a signature on a remittance form, for example. Obviously, authentication is broken when the authenticator is stolen or compromised. An additional authentication factor can mitigate this problem if the authenticators are kept separate from each other. Nevertheless, two-factor authentication systems are susceptible to active attacks [1]:

– Even if a service protects the user against phishing attacks, e.g. by introducing authenticators which are bound to a single session or transaction, a *man-in-the-middle attacker* (MITM) may still be able to mount a *relay* attack. For example, a fraudster could fake an online banking website to intercept a transaction, possibly modify this and then forward the modified transaction to the real bank. The attacker accesses the legitimate service using the authenticators

provided by the user. This may be a simple password, a passphrase generated by a token or a signature on a smart card.
- With a *Trojan*, an attacker can read the user's input (e.g. passwords) and modify the displayed output (e.g. beneficiary of a credit transfer). Effectively, the Trojan is a MITM which has full control over the input and output of a user's computer. Even worse, the Trojan can also start transactions the computer's peripheral devices. A Trojan can, for instance, use a smart card for authentication when this is inserted into the reader. The legitimate user might only see an LED blinking on the reader—if he notices anything at all.
- We often need to use public terminals. ATMs or ticket machines can be manipulated (*skimming attack*). Similar to a Trojan attack, the attacker has full control over which actions are displayed to the user and which actions the terminal is actually carrying out.

Regardless of what authentication mechanism is actually used, many systems have one fundamental shortcoming—the lack of a trusted user interface [2].

Some chip card readers have a secure PIN pad. However, the reader's display is not often used as a secure output device. Even if it displays details of the transaction, the screen, typically very small, is not able to display all the necessary. In effect, the user will often not read all the information shown on the card reader's display. Furthermore, two-factor authentication using smart cards requires users to buy an additional device, which they would have to carry around with them permanently in order to be able to log in securely at any time. Even if users actually did this, they would not be able to use the smart card reader at public terminals, so this would not protect them against skimming attacks.

More and more new smartphones are being equipped with Near Field Communication (NFC) hardware. To a large extent, this wireless communications technology is compatible to the ISO 14443 series of standards for contactless smart cards, which means that smartphones can communicate with such cards.

In this paper we are presenting the concept of a *mobile smart card reader*: an NFC-enabled smartphone used as a chip card reader for contactless smart cards. Smartphones offer rich input and output options which would allow secure user authentication and transaction verification on the actual phone. When plugged into a computer, the phone is recognised as a standard smart card reader with its own PIN pad and display. We also describe a sample use case in which the mobile reader is used for authentication with the German electronic identity card.

1.1 Our contributions

The main contributions of this paper are:

- to present and analyse the concept of a mobile chip card reader for two-factor user authentication;
- to present the use of the mobile reader as a trusted intermediary for defence techniques against relay attacks;

– to show partial implementation of the concept, including secure PIN entry functions based on PACE;
– to describe a use case of the mobile reader, namely with an identity card.

1.2 Structure of the paper

In section 2, we give a brief introduction to the German electronic identity card (nPA). Here, we concentrate on those aspects and problems which also play a role for other smart cards. In section 3, we discuss the concept and implementation of the mobile smart card reader at greater depth. In section 4, we go on to present an informal security analysis of the concept and briefly discuss related work in section 5. Section 6 contains a summary of the paper and offers an outlook for possible future research.

2 German electronic identity card

The German electronic identity card (nPA) is a contactless smart card conforming to ISO 14443. The chip has three applications, ePassport, eID and eSign. Although the chip's different applications are accessed in a similar manner, here we only need to discuss the eID application in more detail.[1]

The nPA's eID application can be used for electronic identification by e-Government or e-Business service providers (SPs). Since the eID application contains sensitive data about the card holder (e.g. name, address, date of birth), information is only revealed to authorised parties. Apart from this, the card holder can individually select or deselect what data are to be read, and finally authorise the transaction by entering his or her secret PIN.

The card holder can, for example, use his nPA to log into a website or state the delivery address for goods bought online. Accessing the eID application typically involves the following steps:

1. The card holder's web browser finds an object embedded in the SP's website, and this object starts a client application on the card holder's computer.
2. The client application displays the permissions requested by the SP and the purpose of the transaction (e.g. "login to the website").
3. The card holder checks the information and restricts the SP's permissions further if he wishes to do so.
4. The card holder authorises the transaction to the nPA by entering his PIN using Password Authenticated Connection Establishment (PACE).[2]
5. The SP and the nPA perform mutual authentication using terminal authentication (TA) and chip authentication (CA).[2]
6. The SP reads the data from the nPA using Secure Messaging (SM).

[1] A complete overview is given in BSI TR-03127 [3].
[2] Cryptographic protocols to access the nPA (e.g. PACE, TA and CA) are dealt with in BSI TR-03110 [4].

BSI TR-03119 [5] specifies three types of card readers for the nPA: A simple reader which only provides the ISO 14443 interface (Cat-B), a reader which must also feature a PIN pad (Cat-S) and finally the Cat-K reader which must additionally incorporate a display.[3]

With simple chip card readers which do not have a PIN pad, the user has to input his secret authenticator via the client application, where there is a danger that it might be stolen using malware. Furthermore, access to the smart card is not protected, so all applications on the card holder's computer can communicate with the card. This means that a Trojan can gain full control over the nPA whenever this is inserted in the reader and can then modify legitimate transactions and/or surreptitiously start new transactions.

Readers with a dedicated PIN pad protect the card holder's secret information from malware on the host computer, but the card holder may still be tricked into revealing the PIN by "social engineering". Also, a MITM[4] between the client application and the smart card reader might be able to modify a transaction on-the-fly, because communications with the reader are not protected. The user only sees the original transaction information on his computer screen and enters his PIN on the reader as requested—thus confirming the modified transaction.

To counter this kind of attack, the reader's display, which is assumed to be trustworthy, shows the transaction details. BSI TR-03119 requires the display to have at least two lines with 16 characters each which is typical for common smart card readers. This, however, does not give the card holder a complete and functional overview of the transaction. For example, the nPA offers a total of 15 independent permissions, which are shown in separate screens on the reader's display. Consequently, it is likely that the user will ignore the display and simply press "OK".

As far as the use of public terminals is concerned, the need for secure user input and output is an even more urgent issue. If, for instance, the card holder uses eID to prove his age in a video shop, he must allow the shop access to his card by entering his PIN. How can he be sure that the terminal does only what he is allowing it to do—and that it does not read more personal data from the card than has been permitted? In the past, it has been shown that skimming of credit card terminals can be performed at a low cost [2].

3 Mobile smart card reader

The concept behind the mobile smart card reader is as follows: The smartphone is connected to a computer or public terminal and can then be used it as a secure input and output device for the smart card (see fig. 1). The mobile reader counteracts malware on the host computer and skimming at a public terminal.

[3] The classes also differ in security requirements and other functions which are not explained in detail here (see [5]).

[4] The MITM can use malware either on the host computer or on a hardware module installed between the reader and the user's computer.

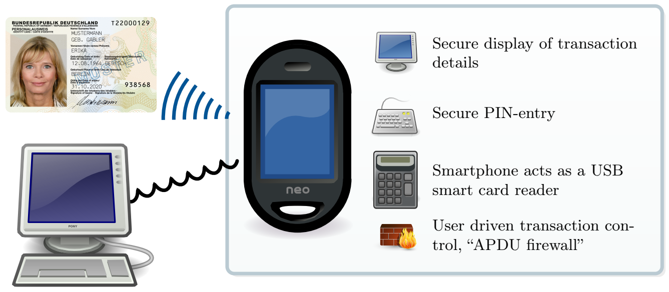
Secure display of transaction details
Secure PIN-entry
Smartphone acts as a USB smart card reader
User driven transaction control, "APDU firewall"

Fig. 1. Using the mobile smart card reader as secure input and output device for a computer that may not be trustworthy

Although this concept is fairly straight forward, there are a number of stumbling blocks to overcome before it can actually be implemented. In this chapter, we shall address these problems. A more detailed discussion can be found in [6].

3.1 Standardised drivers for mobile usage

Mobile connectivity is one of the great advantages of smartphones. Most devices offer multiple interfaces such as Wi-Fi, Bluetooth, USB or GSM/UMTS. Approaches for connecting the smartphone being used as smart card reader to a host computer could be just as diverse.

Even though wired connectors pose drawbacks—especially in mobile use cases—we advocate the use of USB for connecting the mobile card reader to the host computer, the reason being that USB is the only standardised method of communicating with smart card readers. USB CCID readers can be accessed on standard operating systems (including Windows, Mac OS and Linux) without any special configuration—currently-used operating systems usually come with the drivers installed.

Although USB CCID [7] specifications define some security related operations (e.g. PIN verification), the most recent Windows driver (2003 version) implements only simple commands for transmitting APDUs to the card.[5] This means that the PIN pad of an USB CCID compliant reader cannot be used without installing additional software (simple transmissions to the card are possible, however). On the other hand, the Unix driver libccid[6] is updated regularly and includes PIN pad support.

In order to use the mobile phone as a smart card reader, we have implemented an application called CCID emulator.[7] The emulator accesses the USB hardware through the Linux kernel module GadgetFS[8] (see fig. 2). Most USB chipsets on

[5] `http://msdn.microsoft.com/en-us/windows/hardware/gg487509`

[6] `http://pcsclite.alioth.debian.org/ccid.html`

[7] All software developed for this paper can be found at: `http://vsmartcard.sourceforge.net`

[8] `http://www.linux-usb.org/gadget/`

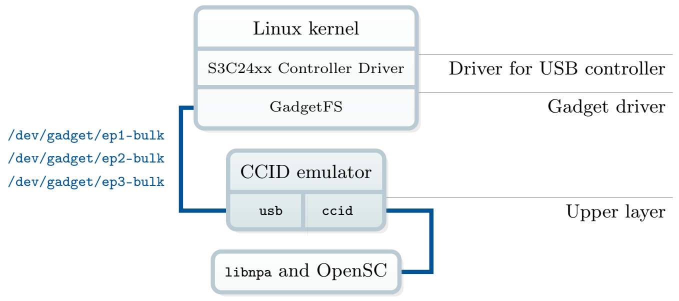

Fig. 2. Software layers of the CCID emulator running on the Openmoko Neo FreeRunner

smartphones already support USB gadget mode to provide USB functionality to external computers (e.g. USB mass storage or USB networking). For smart card access on the mobile phone, we use OpenSC[9] which supports multiple interfaces—most prominently PC/SC—to communicate to the smart card.

3.2 Interface to contactless smart cards

NFC is an umbrella technology encompassing such well-established standards as ISO/IEC 18092, the ISO/IEC 14443 series and JIS X6319-4. NFC supports three different modes of operation: peer-to-peer, card emulation and reader/writer. Interoperability of different NFC devices is one of the main focuses of the NFC Forum[10]. Mobile phones have limitations, e.g. with regard to chip complexity and power consumption. Consequently, the function range provided by NFC phones does not meet the needs of a dedicated smart card reader (e.g. regarding field strength or APDU size).

We expect the contactless chips of future phone's to provide the same functions as dedicated smart card readers. In the absence of fully compatible NFC-enabled phones, we chose to use an Openmoko mobile phone with an attached PN512 based board (extracted from Reiner SCT cyberJack RFID basis, see fig. 3). In this way we were able to avoid any problems which might arise from the differences between NFC and ISO 14443 and concentrate fully on the mobile reader concept.

3.3 Secure PIN entry

In order to enter the PIN securely, the user must have a trusted interface to the intended application. This requires a trusted channel from the input devices to the application. To ensure that the user knows which application is requesting his authenticator and for what purpose, the application's output, too, must presented to the user in a secure form. User awareness is also greatly enhanced

[9] `http://www.opensc-project.org/`
[10] `http://www.nfc-forum.org`

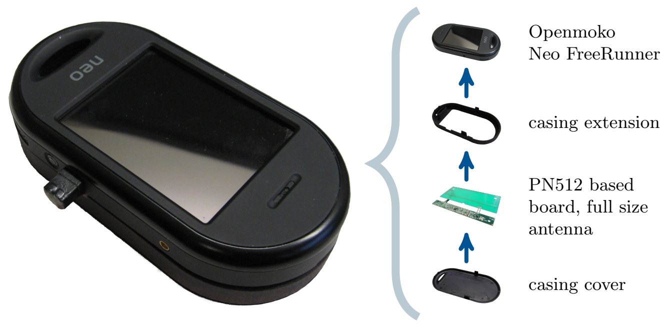

Fig. 3. Openmoko mobile phone with casing extension, PN512 board and antenna.

by the mobile smart card reader's consistent user interface—which remains the same even when it is used at different vendor's public terminals or points of sale.

The requirements of secure input and output will be discussed later (see section 4.1). On account of the mobile nature of the smart card reader presented here, it is desirable to use a PIN input method that reduces the risk of shoulder surfing (see section 4.2). Here, we shall focus on the technical aspects of communication between the host computer and the mobile smart card reader.

Applications on the host computer communicate with smart card readers using PC/SC middleware. Smart card reader drivers map the PC/SC commands to the reader's hardware interface. For ease of implementation, USB CCID commands are very similar to their PC/SC counterpart. For example, the PIN verification data structure according to PC/SC pt. 10 [8] is almost identical to the `PIN_VERIFY` data structure in USB CCID so that the hardware driver only has to change the byte order where necessary.

The CCID emulator allows simple PIN verification and modification defined by USB CCID or PC/SC pt. 10, respectively. No modifications to existing drivers are required. These simple commands, however, are not suitable for the nPA, because the reader would send the PIN to the card via the contactless interface without any kind of protection. The CCID emulator therefore adds a new extension to USB CCID to apply the PACE key agreement protocol. This extension can be seamlessly integrated into the USB standard as it is equivalent to its PC/SC counterpart, in which PACE has recently been standardised [9].

Our solution for PACE via USB CCID requires modifications to existing drivers.[11] However, the traditional USB CCID and PC/SC commands do not have this restriction and can be used without leaving a footprint on the host computer.

3.4 Transaction control and visualisation

We want to empower the user to have greater control over what is exactly done with his smart card. Usually the user has to trust the transaction details shown

[11] The CCID emulator is provided with a patch for libccid.

on his computer screen, at a public terminal or on the point of sale device. With the mobile smart card reader, the user has full and precise control over all commands sent to the smart card.

The user can impose limits to the transactions he is willing to make with his card before the actual transaction even begins. During smart card communication, the reader passes on those commands to the card the user has allowed. Additionally the details are displayed to the user for confirmation. This might be called a "Man-in-the-Middle defence" [10]. The mobile smart card reader can also log all details of the transaction to obtain proof of what actions the user carried out and what actions he did not.

When the nPA is used for electronic identification, the first APDU from the SP commits itself to the permissions allowed in the ensuing transaction. In addition, descriptive data concerning details of the SP and the purpose of the transaction are transmitted to the reader. The CCID emulator displays all details *before* the user approves the transaction by entering his PIN.

When the SP sends APDUs to the nPA, the CCID emulator checks. One distinct advantage of the CCID emulator over existing smart card readers when handling an nPA is that it applies further sanity checks when the SP proves its authenticity using TA. For example, the CCID emulator checks the validity period of the SP's certificate against the current date. The CCID emulator performs all checks independent of the nPA. This additional mode of verification adds robustness to counteract potential smart card implementation errors.

It is important to design the user interface for the transaction inspection system so that it can be operated easily by most users. User interface design for security relevant programs running on an NFC-enabled phone is challenging due to a variety of factors such as space-constraints, environmental factors and the need to handle the card and the phone simultaneously. These problems are discussed in detail in [11].

3.5 Independent smart card access

As a smartphone always comes with a battery, the mobile smart card reader is independent of a host computer. This allows the user to manage or inspect his smart card without having to rely on a computer or terminal. Depending on the actual type of smart card, the user can detect fraudulent transactions by reading data stored on the card. The user can also update any obsolete data stored on the card and change or unblock his PIN.

Independent PIN management can be used to create a *temporary PIN*. The temporary PIN does not reveal any details about the permanent secret information memorised by the user. This can be useful if the card has to be used on a terminal that does not support an external (mobile smart card) reader. When all transactions with the terminal are completed, the smartphone changes the PIN back to the user's permanent secret PIN.

The temporary PIN helps to reduce theft and tracking of the permanent secret PIN. However, one should keep in mind that the terminal has unhindered access to the card when the user inserts his card into the terminal and enters his temporary PIN. Usefulness of the temporary PIN and inspection or validation of data stored on the card depends on the specific type of card and should be considered individually for each case.

We developed a program for the nPA called `npa-tool`, which is based on OpenSC. It can change and unblock the PIN using PACE, thus enabling the user to enter temporary PINs for transactions at public terminals.

The `npa-tool` is also able to update the nPA's approximated current date, allowing the actual card to recognize when an SP is using an expired certificate. The card only updates its state if specially-crafted certificates are presented. A *date update service* as suggested in BSI TR-03127 [3] might be able to provide these certificates for the `npa-tool`. We have discussed other applications of the nPA combined with a mobile phone in an other paper [12].

4 Security considerations

4.1 Secure execution environment

Many of the attacks that can be made on desktop systems can also be made on smartphones. Vulnerabilities of the software supplied with the smartphone, as well as third-party apps can impose a threat, even when installed from a trusted distribution platform.

User expectations and infrastructure of mobile phones and desktop computers differ considerably. Smartphones are expected to be reliable, whereas users of desktop computers are accustomed to errors. More importantly, most smartphone platforms are closed systems preventing access to hardware or core software. Most users knowingly accept these restrictions. This makes it possible to apply extensive security measures to the mobile platform.

Regardless of how a secure execution environment is specifically implemented on a smartphone, it should fulfil the following requirements [13]:

- protection of the software against external interference;
- observation of the computations and data of a program running within an isolated environment via controlled inter-process communication only;
- secure communication between programs running in independent execution environments; and
- provision of a trusted channel between an input/output device and a program running in an isolated environment.

A number of different isolation and verification techniques are suitable for mobile phones: Static and dynamic code verification [14], policy based isolation [15, 16, 17] or various types of virtualisation [18, 19] possibly in combination with a hardware security module [20].

Correct implementation of software isolation is difficult. Again, the nature of the software distribution systems for mobile platforms can help to mitigate known problems. Critical software updates can be pushed to the smartphone by trusted parties—no user interaction is required. Also, a combination of security measures can be applied, for example, software verification *and* software isolation.

4.2 Protection against shoulder surfing

When a user enters a PIN on the mobile smart card reader in a public space, there is a danger that a potential attacker might see this. With the mobile smart card reader, the user can block the attacker's view by turning around in whichever direction he wishes, or by covering up the display.

His freedom of movement is limited only by the length of the USB cable. However, if a headset [21] or a pre-shared secret [22] is available, the user could also enter a modified version of the PIN which the attacker cannot use with the smart card alone. In addition, zero-knowledge proofs can be applied if the mobile smart card reader saves the PIN [23] securely.

4.3 Attacks via USB

Experience has shown that it is possible to compromise mobile phones and install malicious software via USB [24]. Therefore the USB connection to the host computer has to be regarded as part of the mobile reader's potential attack surface.

Mobile phones typically support several different USB profiles and their USB stack is generally more complex than that used with dedicated smart card readers, resulting in a comparatively larger the attack surface. A solution to this problem might be to introduce a dedicated smart card reader mode on the mobile phone. In this mode, the phone only supports CCID and all other USB drivers are unloaded from the kernel. This would reduce the attack surface of the mobile reader. Further research is required in this area.

5 Related work

Use of an NFC-enabled mobile phone as an intermediary between the smart card and the terminal has been suggested in several publications. Two main use cases are under discussion in respective literature [25]: On the one hand a mobile phone may be used to carry out relay attacks [26, 27]. On the other it may be used to inspect transactions and provide users with a trusted user interface [10].

Use of the nPA for two-factor-authentication on a mobile phone has also been suggested by Hühnlein et. al [28]. The authors suggest a security aware design for an Open eCard App to also provide defence against unconventional attack vectors. If possible, the Open eCard App should make use of platform-specific security features such as a Trusted Execution Environment. The Open eCard App should feature all the middleware aspects required by BSI TR-03112 [29]—including online communication to the SP. Vulnerabilities of the official client software for the nPA (AusweisApp) have shown, that this opens additional paths for conventional attacks. In contrast, our solution aims to reduce complexity by providing the reader interface only.

Mannan and Oorschot [30] propose a protocol called MP-Auth, which uses a personal handheld device (e.g. a mobile phone) to derive a one time password for authentication and session key agreement for use with an untrusted computer. The user has to enter, on his phone, both his long-term password and a nonce generated by the server. Furthermore, the server's public key must be stored on the phone, too. A one-time-password is then generated and the user has to enter this on the (untrusted) computer to authenticate himself to the server and establish a shared secret which is then used for the remaining session. The authors propose protocol steps for protecting the integrity of the nonce and transactions issued after the session has been established.

Hart et. al [31] propose using the SIM card in a mobile phone as a secure storage medium for web credentials. Their system uses either a browser extension on the client computer or a dedicated authentication server to request the (encrypted) credentials from the user's phone (via an SMS gateway). The browser extension approach is vulnerable to malware installed on the client computer, which can steal the user's credentials while they are being entered into the extension. Furthermore, the paper does not discuss the secure display of transaction information on the phone, so that in both approaches the system is vulnerable to phishing attacks.

Several publications have suggested to use the actual phone as an authentication token [32, 33, 34]. The main difference between these publications and this paper is that we do not propose to integrate the smart card into the mobile phone but use the phone as a trusted reader. This allows a solution which serves as a drop-in replacement for an existing infrastructure. Furthermore, keeping the smart card separate from the mobile reader makes our solution at least partially more robust against mobile malware because a compromised mobile phone does not have permanent access to the user's smart card.

A lot of research has been carried out regarding the security of various two-factor authentication schemes. For example, Drimer et al. [35] examined the CAP protocol used for online banking in Europe. Even though the protocol makes use of smart cards and trusted handheld readers, the authors found vulnerabilities in the protocols, which enabled them to perform a relay attack. This shows that even secure infrastructure cannot protect the user against failure at the protocol level. In contrast to proprietary protocols which rely on obscurity the protocols used for the nPA are freely available and have been proven to be cryptographically secure [36].

6 Conclusion and future work

In this paper we have discussed the idea of using an NFC-enabled mobile phone as a chip card reader for contactless smart cards. We have explained issues—implementation details as well as security considerations—that have to be taken into account when implementing a mobile smart card reader. We have discussed how the mobile reader can help to obtain secure user authentication in untrustworthy environments and how the mobile reader can be used as a trusted intermediary at a public terminal, which is susceptible to skimming attacks. Furthermore, we have described a use case for the German electronic identity card.

As discussed in section 4.1 a secure execution environment is of central importance for the work presented in this paper. A lot of research is currently being done on techniques for making mobile phones more secure, as well as on new attack vectors for these phones.

Our proposal centres around the use of NFC radio technology. This relatively new technology should be subjected to rigorous security evaluation, especially as it is being used in financial transaction systems such as Google Wallet.

Acknowledgements

Dominik Oepen is being sponsored by the German Research Foundation (DFG) Graduate School METRIK.

References

1. Bruce Schneier. Two-factor authentication: too little, too late. *Commun. ACM*, 48(4):136, April 2005.
2. Ben Adida, Mike Bond, Jolyon Clulow, Amerson Lin, Steven Murdoch, Ross Anderson, and Ron Rivest. Phish and Chips. In Bruce Christianson, Bruno Crispo, James Malcolm, and Michael Roe, editors, *Security Protocols*, volume 5087 of *Lecture Notes in Computer Science*, pages 40–48. Springer Berlin / Heidelberg, 2009.
3. Bundesamt für Sicherheit in der Informationstechnik. *Technical Guideline TR-03127: Architecture electronic Identity Card and electronic Resident Permit*, 1.13 edition, March 2011.
4. Bundesamt für Sicherheit in der Informationstechnik. *Technical Guideline TR-03110: Advanced Security Mechanisms for Machine Readable Travel Documents*, 2.05 edition, October 2010.
5. Bundesamt für Sicherheit in der Informationstechnik. *Technische Richtlinie TR-03119: Anforderungen an Chipkartenleser mit nPA Unterstützung*, 1.2 edition, May 2011.
6. Frank Morgner. Mobiler Chipkartenleser für den neuen Personalausweis: Sicherheitsanalyse und Erweiterung des "Systems nPA". Master's thesis, Humboldt-Universität zu Berlin, 2012.
7. USB Implementers Forum. *Universal Serial Bus. Device Class: Smart Card CCID*, April 2005.
8. PC/SC Workgroup. *Interoperability Specification for ICCs and Personal Computer Systems: Part 10 IFDs with Secure PIN Entry Capabilities*, 2.02.08 edition, April 2010.
9. PC/SC Workgroup. *Interoperability Specification for ICCs and Personal Computer Systems: Part 10 IFDs with Secure PIN Entry Capabilities – Amendment 1: PIN-Verification with Contactless Smart Cards based on PACE*, 2.02.08 edition, 2011.

10. Ross Anderson and Mike Bond. The Man-in-the-Middle Defence. In Bruce Christianson, Bruno Crispo, James Malcolm, and Michael Roe, editors, *Security Protocols*, volume 5087 of *Lecture Notes in Computer Science*, pages 153–156. Springer Berlin / Heidelberg, 2009.
11. Dominik Oepen. Authentisierung im mobilen Web: Zur Usability eID basierter Authentisierung auf einem NFC Handy. Master's thesis, Humboldt Universität Berlin, September 2010.
12. Frank Morgner, Dominik Oepen, Wolf Müller, and Jens-Peter Redlich. Mobiler Leser für den neuen Personalausweis. In *Tagungsband zum 12. IT-Sicherheitskongress*, pages 227–240, Gau-Algesheim, May 2011. SecuMedia Verlag.
13. Eimear Gallery and Chris J. Mitchell. Trusted Mobile Platforms. In Alessandro Aldini and Roberto Gorrieri, editors, *FOSAD*, volume 4677 of *Lecture Notes in Computer Science*, pages 282–323. Springer, 2007.
14. Thomas Bläsing, Aubrey-Derrick Schmidt, Leonid Batyuk, Seyit A. Camtepe, and Sahin Albayrak. An Android Application Sandbox System for Suspicious Software Detection. In *5th International Conference on Malicious and Unwanted Software (Malware 2010)*, Nancy, France, France, 2010.
15. Mohammad Nauman, Sohail Khan, Xinwen Zhang, and Jean-Pierre Seifert. Beyond Kernel-level Integrity Measurement: Enabling Remote Attestation for the Android Platform. 2010.
16. Xinwen Zhang, Onur Aciiçmez, and Jean-Pierre Seifert. Building Efficient Integrity Measurement and Attestation for Mobile Phone Platforms. In Andreas U. Schmidt and Shiguo Lian, editors, *MOBISEC*, volume 17 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 71–82. Springer, 2009.
17. Onur Acıicmez, Afshin Latifi, Jean-Pierre Seifert, and Xinwen Zhang. A Trusted Mobile Phone Prototype. In *Proceedings of IEEE Consumer Communications and Networking Conference (CCNC 2008)*, Las Vegas, 2008. Samsung Electron. R&D Center, San Jose.
18. Marcel Selhorst, Christian Stüble, Florian Feldmann, and Utz Gnaida. Towards a Trusted Mobile Desktop. In Alessandro Acquisti, Sean W. Smith, and Ahmad-Reza Sadeghi, editors, *TRUST*, volume 6101 of *Lecture Notes in Computer Science*, pages 78–94. Springer, 2010.
19. Joo-Young Hwang, Sang-Bum Suh, Sung-Kwan Heo, Chan-Ju Park, Jae-Min Ryu, Seong-Yeol Park, and Chul-Ryun Kim. Xen on ARM: System Virtualization Using Xen Hypervisor for ARM-Based Secure Mobile Phones. In *Proceedings of IEEE Consumer Communications and Networking Conference (CCNC 2008)*, Las Vegas, 2008. Samsung Electron. R&D Center, San Jose.
20. Kari Kostiainen, Elena Reshetova, Jan-Erik Ekberg, and N. Asokan. Old, New, Borrowed, Blue – A Perspective on the Evolution of Mobile Platform Security Architectures. In Ravi S. Sandhu and Elisa Bertino, editors, *CODASPY*, pages 13–23. ACM, 2011.
21. Toni Perkovi, Mario Cagalj, and Nitesh Saxena. Shoulder-Surfing Safe Login in a Partially Observable Attacker Model. Technical report, 2011.
22. Nicholas J. Hopper and Manuel Blum. A Secure Human-Computer Authentication Scheme. Technical report, Carnegie Mellon University, Pittsburgh, May 2000.
23. Volker Roth, Kai Richter, and Rene Freidinger. A PIN-Entry Method Resilient Against Shoulder Surfing. Technical report, 2004.
24. Zhaohui Wang and Angelos Stavrou. Exploiting Smart-Phone USB Connectivity for Fun and Profit. In *Proceedings of the 26th Annual Computer Security Applications Conference*, ACSAC '10, pages 357–366, New York, NY, USA, 2010. ACM.

25. Ross Anderson. Position statement in RFID S&P panel: RFID and the middleman. In *Proceedings of the 11th International Conference on Financial cryptography and 1st International conference on Usable Security*, FC'07/USEC'07, pages 46–49, Berlin, Heidelberg, 2007. Springer-Verlag.
26. Lishoy Francis, Gerhard Hancke, Keith Mayes, and Konstantinos Markantonakis. Practical NFC Peer-to-Peer Relay Attack Using Mobile Phones. In Siddika Ors Yalcin, editor, *Radio Frequency Identification: Security and Privacy Issues*, volume 6370 of *Lecture Notes in Computer Science*, pages 35–49. Springer Berlin / Heidelberg, 2010.
27. Lishoy Francis, Gerhard Hancke, Keith Mayes, and Konstantinos Markantonakis. Practical Relay Attack on Contactless Transactions by Using NFC Mobile Phones. Cryptology ePrint Archive, Report 2011/618, 2011. `http://eprint.iacr.org/`.
28. Detlef Hühnlein, Dirk Petrautzki, Johannes Schmölz, Tobias Wich, Moritz Horsch, Thomas Wieland, Jan Eichholz, Alexander Wiesmaier, Johannes Braun, Florian Feldmann, Simon Potzernheim, Jörg Schwenk, Christian Kahlo, Andreas Kühne, and Heiko Veit. On the design and implementation of the Open eCard App. In *GI SICHERHEIT 2012 Sicherheit – Schutz und Zuverlässigkeit*, March 2012.
29. Bundesamt für Sicherheit in der Informationstechnik. *Technical Guideline TR-03112: eCard-API-Framework*, 1.1.1 edition.
30. Mohammad Mannan and P. van Oorschot. Using a Personal Device to Strengthen Password Authentication from an Untrusted Computer. In Sven Dietrich and Rachna Dhamija, editors, *Financial Cryptography and Data Security*, volume 4886 of *Lecture Notes in Computer Science*, pages 88–103. Springer Berlin / Heidelberg, 2007.
31. Jonathan Hart, Konstantinos Markantonakis, and Keith Mayes. Website Credential Storage and Two-Factor Web Authentication with a Java SIM. In Pierangela Samarati, Michael Tunstall, Joachim Posegga, Konstantinos Markantonakis, and Damien Sauveron, editors, *Information Security Theory and Practices. Security and Privacy of Pervasive Systems and Smart Devices*, volume 6033 of *Lecture Notes in Computer Science*, pages 229–236. Springer Berlin / Heidelberg, 2010.
32. Dirk Balfanz and Edward W. Felten. Hand-Held Computers Can Be Better Smart Cards. In *Proceedings of the 8th USENIX Security Symposium*, pages 15–24, Washington, D.C, August 1999.
33. Steffen Hallsteinsen, Ivar Jorstad, and Do Van Thanh. Using the mobile phone as a security token for unified authentication. In *Proceedings of the Second International Conference on Systems and Networks Communications*, ICSNC '07, pages 68–74, Washington, DC, USA, 2007. IEEE Computer Society.
34. Sandeep Tamrakar, Jan-Erik Ekberg, Pekka Laitinen, N. Asokan, and Tuomas Aura. Can Hand-Held Computers Still Be Better Smart Cards? In Liqun Chen and Moti Yung, editors, *Trusted Systems*, volume 6802 of *Lecture Notes in Computer Science*, pages 200–218. Springer Berlin / Heidelberg, 2011.
35. Saar Drimer, Steven Murdoch, and Ross Anderson. Optimised to Fail: Card Readers for Online Banking. In Roger Dingledine and Philippe Golle, editors, *Financial Cryptography and Data Security*, volume 5628 of *Lecture Notes in Computer Science*, pages 184–200. Springer Berlin / Heidelberg, 2009.
36. Jens Bender, Marc Fischlin, and Dennis Kügler. Security Analysis of the PACE Key-Agreement Protocol. In Pierangela Samarati, Moti Yung, Fabio Martinelli, and Claudio Ardagna, editors, *Information Security*, volume 5735 of *Lecture Notes in Computer Science*, pages 33–48. Springer Berlin / Heidelberg, 2009.