Mobile eID application for the German identity card

Technical Report

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Department of Computer Science
Theoretical Computer Science
Cryptography and Computer Algebra

Moritz Horsch[1] Johannes Braun[1] Alex Wiesmaier[2]

[1]Technische Universität Darmstadt
Hochschulstraße 10, 64283 Darmstadt, Germany
{jbraun,horsch}@cdc.informatik.tu-darmstadt.de

[2]AGT Group (R&D) GmbH
Hilpertstraße 35, 64295 Darmstadt, Germany
awiesmaier@agtinternational.com

Abstract

We present MONA, the first and currently the only mobile eCard application enabling the usage of the German ID card with NFC-enabled smartphones. We show how the technical obstacles resulting from poorly fitting specifications and standards are overcome. We also present MONA's design. It is optimized for a maximum of modularity and platform independence and allows MONA to run on a variety of platforms including standard computers. We further compare respective benchmarks on low price phones and computers and provide first timings on high end smartphones. Finally, we explain what is left to be done for the designated open source MONA project to become an alternative to the established AusweisApp eCard application.

1 Introduction

1.1 Motivation

Many countries equip their citizens with electronic identity (eID) cards providing the machine readable travel document functionality specified by the *International Civil Aviation Organization* (ICAO) [1]. Most ID cards are based on the *European Citizen Card* (ECC) [2] specification and often feature a contactless interface according to ISO/IEC 14443 [3]. The German identity card provides common sovereign applications like ePassport [1] for border control as well as authentication in the private sector. German citizens are able to use their ID card in eBusiness and eGovernment applications for login and registration processes. In order to use the eID and eSign functionality of the German ID card, a smart card reader and an eCard middleware is necessary.

Mobile phones are the permanent attendants in the modern society, which makes them an interesting candidate for ubiquitous security elements. The *Near Field Communication* (NFC) [4, 5] technology, more and more phones are equiped with, is a short-range communication technology. NFC allows communicating with RFID chips and contactless smart cards according to ISO/IEC 14443 [3]. Hence, integrated with a mobile eID application, NFC-enabled mobile phones allow the fully mobile utilization of e.g. the German ID card.

In this paper we are concerned with an lightweight and efficient implementation of such an eID application for the German ID card with regard to the restricted capabilities of mobile phones compared to standard computers. The paper at hand is structured as follows. Section 1.2 provides the necessary background on the German ID card and the corresponding infrastructure. Section 1.3 gives an overview over related work. Section 2 presents details on the implementation of the mobile eID application MONA. Section 3 provides an outlook on future development and Section 4 concludes the paper.

1.2 German identity card

The new German ID card is a polycarbonate smart card in the size of a credit card and is equipped with a contactless chip according to ISO/IEC 14443 [3]. The card holder's personal data is printed on the card surface and stored on the integrated chip. Among other things, the stored data contains name, date and place of birth, nationality and a biometric passport photograph of the card holder. The card is compliant to the ECC [2] and ICAO Doc 9303 [1] specifications. Common biometric features like the ePassport [1] application and in particular functions for electronic authentication and the generation of qualified electronic signatures are supported.

1.2.1 Access Control

The access control of the ID card is realized by the *Password Authenticated Connection Establishment* (PACE) and *Extended Access Control* (EAC) as specified in TR-03110 [6] by the *German Federal Office for Information Security*. The PACE protocol performs a user authentication using a *Personal Identification Number* (PIN) and establishes a secure connection between the ID card and the card reader to protect the communication over the contactless interface. The card holder gives his consent for card access by

entering the PIN. The EAC protocol consists of the sub-protocols *Chip Authentication* (CA) and *Terminal Authentication* (TA) and performs a mutual authentication between the card terminal[1] and the ID card. EAC guarantees the authenticity of the smart card chip and the adherence to the access restrictions of the terminal. Furthermore, EAC establishes a secure channel between ID card and terminal. The access rights of the terminal are proven by certificates issued by the *German Federal Administration Office*. A proof of security of PACE and EAC is given in [7, 8].

1.2.2 Electronic identification

The electronic identification (eID) functionality permits a proof of identity in eBusiness and eGovernment applications and can additionally act as a replacement for the user name and password based authentication. Thus, the eID functionality can be used for the registration and login process on websites. However, the electronic identification process is not performed by the web service providers themselves. Special eID servers perform it in the name of the services. The eID server manages the certificate issued for the service, performs the security protocols, and reads the personal data stored on the card. The service only receives the data and performs a local authentication process based on its environment. The participants of the scenario are depicted in Figure 1.

1.2.3 eID authentication process

The authentication sequence using the eID function for login into a web service is described in the following:

1. The user opens the website of a service provider, which is hosted on a web server. For example, the service offers a web mail service or an online shop. The user clicks on a link to perform a login process and the eID application is starts.

2. The eID application establishes a TLS connection to the eID server and receives the terminal certificate and a list of requested personal information (data groups on the ID card) which is to be read from the ID card.

3. The certificate description (e.g. issuer, URL, terms of usage) is displayed and the user is able to restrict the data groups which will be submitted to the eID server. The user gives his consent by entering the PIN.

4. The PACE protocol is performed and establishes a secure channel between the card reader and the ID card.

5. The eID server and the ID card perform a mutual authentication using the EAC protocol. The terminal certificate received in Step 2 is used during the Terminal Authentication to prove the access rights of the eID server.

6. The selected data groups are read by the eID server and passed to the web server.

[1] Note that the terminal can be a physical terminal as well as a remote terminal e.g. in form of an eID server.

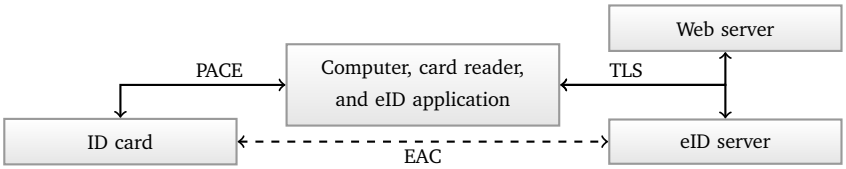

Figure 1: eID infrastructure

7. The web server checks the received data and grants access to the user. The user refreshes the website and can use the service.

1.2.4 eID application

The eID application acts as a middleware between the ID card and the eID server and implements the required communication and cryptographic protocols. The application establishes the communication between card and server and provides the user interaction. Thus, it implements PACE and handles EAC messages between eID server and ID card. The communication between the eID application and both servers is secured by *Transport Layer Security* (TLS).

A governmentally funded implementation of such an eID application, called *AusweisApp* [9], is freely available for computers running Windows, Mac OS and popular Linux distributions. We developed a mobile eID application called MONA (Mobile usage of the new German identity card) [10], which permits to use the German ID card in mobile environments.

1.3 Related Work

Much work on implementing the required protocols and a complete eID application has been done. The official AusweisApp [9] with a size of more than 50 MB is far too big to be used on mobile phones. Alternative proprietary client implementations exist from Ageto [11] and bremen online services [12]. However, these are also clients for stationary computers.

Concerning the mobile realization, [13] describes the first Java implementation of the PACE protocol on a mobile phone. [14] describes several optimizations and significantly improves the performance of the PACE protocol execution on resource restricted devices. [15] presents an efficient implementation of the PACE and the EAC protocol on mobile devices, yet without adhering to the eCard-API (see Section 2.1) . Parallel to these developments, a C implementation of the PACE protocol was realized [16]. In the Androsmex project [17] PACE was implemented for the Android platform. A more common and abstract technical description of the design and implementation of an European eID client is given in [18].

[19] gives an overview over mobile scenarios and describes a shared method of using the eSign function of the German ID card. Furthermore [20] describes a shared authentication of the PACE protocol to avoiding security threats on mobile devices.

2 Mobile eID application

Existing eID applications are designed to use the German ID card with a computer and a card reader. The web browser is used to access web services which support an authentication using the ID card. However, we more often browse the Internet on our smartphone, tablet computer, and notebook. NFC-enabled smartphones can combine a computer and a card reader into a single device. MONA represents the missing mobile eID application and acts as an equivalent to existing applications for computers such as the AusweisApp. Therefore, an NFC-enabled device together with MONA enables a mobile authentication using the German ID card. Thus, username and password can be replaced by using the ID card for fast, secure and comfortable login and registration processes in a mobile environment.

The development of a mobile eID application is a big challenge due to the restricted resources of mobile devices and the fixed infrastructure of the eID environment. The German ID card was rolled out in November 2010, hence we are not able to change or optimize any protocols or communication schemes for the mobile environment. Additionally, the eID server carriers are not willing to support different protocols which are respectively designed and optimized for a mobile and the computer platform. In particular, different protocols are contrary to the objective of having common and standardized interfaces.

Modifications to cope with the mobile environment would be reasonable, but are difficult to achieve. The communication with the ID card is unchangeable, whereas the communication with the eID server could be adjusted in a minor degree. However, that is only possible by changing the specifications considering the interests of the stakeholders like government authorities and eID server carriers. On the contrary, the web service can be adjusted to the capabilities of mobile devices. For instance, this concerns the design, user interface, and distribution of the eID application (MONA) as well as its execution. The communication between the web service and the eID server is precisely specified.

In the following, we describe requirements on the implementation of a mobile eID application and how they are realized in MONA.

2.1 Requirements

The eID infrastructure, including interfaces, communication and security protocols, is defined by the German Federal Office for Information Security in several specifications [21]. A mobile eID application must implement these specifications in order to work with the existing eID infrastructure. The major specifications that have to be taken into account are described in the following:

The Technical Guideline TR-03110 [6] defines the cryptographic protocols PACE and EAC that provide communication security over the contactless interface, user authentication, and mutual authentication between the ID card and the eID server. TR-03112 [22] describes the so called *eCard-API-Framework*. The eCard-API is a common interface for a standardized usage of different smart cards in various applications. The eCard-API-Framework is the technical realization of the German eCard-Strategy [23]. Concerning the German ID card the framework applies to the communication between the eID application and the eID server. The TR-03119 [24] describes requirements for smart card readers. The specification must be taken into account, because a NFC-enabled device acts as card reader in the mobile scenario. The

user interaction with the card holder is specified by TR-03127 [25]. It describes the required password prompt, notifications, and information which must be displayed to the user.

In summary, an eID application must implement the following protocols: PACE, EAC, Secure Messaging, SOAP [26], PAOS [27], and TLS with pre-shared keys [28, 29]. Additionally, the application must implement the required set of eCard-API messages and the specified user interaction.

2.2 Architecture of the mobile eID application MONA

Acting as a middleware between the ID card and the eID server the eID application is responsible for data transformation and communication establishment and relaying. The ID card expects *Application Protocol Data Units* (APDU) according to ISO/IEC 7816 [30], whereas the eID server expects XML encoded messages according to the eCard-API-Framework. The eID application must transform and encode the data into the expected formats of the participants.

The mobile eID application communicates with the eID server over a wireless network like General Packet Radio Service (GPRS) as depicted in Figure 2. The communication to the ID card is established over the NFC interface using the JSR 257 [31]. Messages from the server arrive in a TLS channel and must be decrypted by the application. The messages contain PAOS and SOAP protocol headers, which are only necessary for the transport and can be removed. In a next step the messages are analyzed and processed regarding their data. The data is transformed to the desired structure and forwarded to the destination, for instance, a user interface, a protocol implementation or just directly to the ID card. Processing messages from the ID card works on the same principle.

The steps of the communication progress are separated into different layers. Each layer implements one step in the process and passes the data to the overlying or underlying layer. Every layer uses the same interfaces and does not know anything about the sequence of the layers and its own position. That allows reusing, expanding or interchanging the various layers in an easy way. For example, to use WebSockets [32] instead of SOAP and PAOS for transporting eCard-API messages we can just exchange the SOAP and POAS layer with a respective WebSocket layer. The layered architecture also allows to hide platform specific technologies in a single layer. The ID card communication can be implemented in such a layer for example. On a mobile platform that layer accesses the NFC interfaces, whereas common card reader interfaces are addressed on a computer platform. In case of running the application on a computer or a mobile device we just exchange that layer to fit to the actual platform. As we have the same interfaces between each layer there are no dependencies and thus no changes are needed on the other layers. That makes it easy to achieve platform specific functionalities while major parts of the source can be reused on both platforms.

The architecture of MONA and their layers are depicted in Figure 2. The individual layers are described in the following:

2.2.1 IFD

The IFD (Interface Device) layer provides an interface for contactless communication. It abstracts from the actually employed communication technology. In the mobile environment the NFC interface is used, whereas the Java Smart Card I/O [33] is commonly used on personal computers.

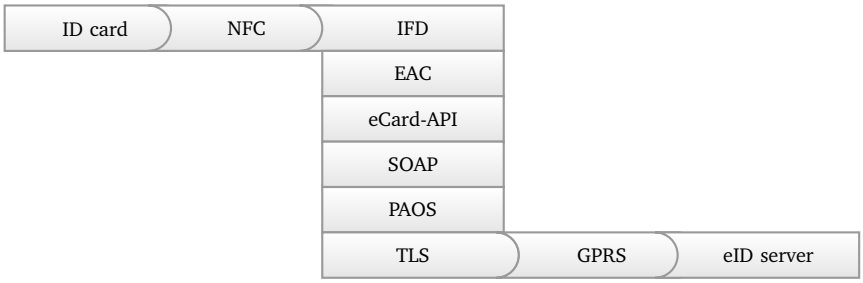

Figure 2: Architecture

2.2.2 EAC

The EAC layer implements the cryptographic protocols PACE and EAC. The transformation between the eCard-API messages and the APDUs are performed in this layer.

2.2.3 eCard-API

The eCard-API layer is the core component of the application. This layer manages the application flow and handles the communication between the application and the eID server. Messages containing information addressed to the ID card are forwarded to the EAC layer, whereas messages for the user are passed to a *user interface controller*, which displays the content. Messages addressed to the eID server are forwarded to the SOAP layer.

2.2.4 SOAP

The SOAP layer implements the required SOAP protocol [26] for the transport of the eCard-API messages between the eID application and eID server. The layer generates the required SOAP format for outgoing messages. Messages from the overlying eCard-API layer are packed in the SOAP body and the corresponding SOAP header is generated. For incoming messages the SOAP structure is removed and the content of the SOAP body is passed to the eCard-API layer.

2.2.5 PAOS

The Technical Guideline TR-03112 defines the *Reverse HTTP binding for SOAP* (PAOS) binding [27] as the transport method for SOAP messages. The regular SOAP over HTTP binding adds a HTTP header to a SOAP message. It uses HTTP requests to send SOAP requests and the respective HTTP reply for SOAP responses. However, to enable the client to handle HTTP or rather SOAP requests a web server would be required. Yet, that is not applicable due to security reasons and restricted resources. PAOS solves this problem by sending SOAP requests with HTTP responses and SOAP responses with HTTP requests. Thus, the PAOS binding allows the server to send requests to the client. The PAOS layer performs the required modifications of the HTTP header.

2.2.6 TLS

The TLS layer implements a TLS variant using pre-shared keys (PSK) according to [28, 29] based on MicroTLS [34]. The layer establishes the TLS connection to the eID server and handles the message flow.

2.3 Implementation

The mobile eID application MONA is implemented as a *MIDlet* [35, 36] using the Java Micro Edition and the NFC-enabled Nokia 6212 [37] device. Additionally, we use the FlexiProvider [38] as a Cryptographic Service Provider (CSP), which offers cryptographic methods and primitives. The required protocols are implemented in different layers as described in Section 2.2. MONA has approximately 20,000 lines of source code and a size of 422 KB using obfuscation.

To have a reusable and flexible implementation we separate the source into different modules. The *MONACore* module includes the implementation of the protocols (e.g. PACE, EAC, PAOS, SOAP, and TLS), the business logic for application flow, and error handling. The source is compliant to Java 1.3 and platform independent regarding a mobile or computer environment. However, there are platform specific interfaces like the communication technology for data exchange with the ID card. JSR 257 [31] offers a Java API for NFC on mobile devices, whereas the Java Smart Card I/O allows to access a card reader on a computer. Additionally, the graphical user interface depends on platform specific APIs, for instance, the Swing and LWUIT [39] framework. Hence, we create two specialized modules *MONAClient* and *MONALocalClient*. These include the platform dependent resources and implementations. Both modules extend the MONACore module and build a fully mobile or computer application.

The separation into different modules allows the usage of the MONACore module without any changes on a mobile device as well as on a computer. Thus, the MONACore module runs without changes on a computer heading towards a lightweight alternative to the above mentioned AusweisApp. Only the platform specific modules MONAClient and MONALocalClient depend on interfaces which are only available on the actual device.

2.4 NFC-enabled devices

At the beginning of our research the Nokia 6212 [37] was the only available NFC-enabled device. Today NFC becomes increasingly popular and even more devices are equipped with NFC. However, the specification of NFC does not fulfil the requirements of the ISO/IEC 14443. The data between the ID card and the eID application is exchanged using APDUs according to ISO/IEC 7816 [30]. The security protocols of the ID card require the transmission of large data packets and therefore the support of *extended length* APDUs, which is not supported by NFC. Hence, current NFC-enabled devices like the *Google Nexus* are not applicable as a card reader for the German ID card. The Nokia 6212, which was released in 2008, surprisingly supports extended length APDUs and is currently the only applicable device for our requirements and a fully-fledged prototype. The difficulty of the extended APDUs is recognized by the manufacturers and we are confident to get next generation devices in the beginning of 2012.

	INIT	TLS	PACE	TA	CA	RI, DATA
Nokia 6212	839	2516	7807	3958	1320	1648
Computer	110	562	2078	2125	407	860

Table 1: Benchmarks (in msec)

2.5 Results

MONA is the first mobile eID application for the German ID card and allows using the eID function for authentication on NFC-enabled devices[2]. We show that a complex eID application can be implemented in a lightweight and efficient manner for mobile devices and NFC-enabled devices provide a basis for a mobile usage of the German ID card.

2.5.1 Benchmarks

The Nokia 6212 was shipped in 2008 as a phone from the lower price segment. Thus, it cannot keep up with modern smartphones. The device has very limited resources and only supports the Enhanced Data Rates for GSM Evolution (EDGE) network for data transmission and internet access. This leads to long (cf. Table 1) but promising (cf. Table 2) runtimes. Table 1 shows the measured runtimes of the protocols. The low wireless speed of the EDGE network appears in the measurements of TLS, CA, TA, and RI, Data (cf. Table 1) where communication with the eID server is involved. In detail, the wireless communication takes 984 msec on the computer and 6204 msec on the mobile device. The NFC interface has less field strength in comparison to a card reader. The card communication (sending and receiving messages to/from the ID card) takes 5474 msec on the mobile device and 4515 msec using a computer and card reader. The implementation of the PACE protocol contains sophisticated optimizations according to [14]. Yet, it still has a considerable runtime resulting from the computationally intensive elliptic curve operations. The total runtime on the Nokia is 26.7 sec, while it takes 7.3 sec on a computer.

2.5.2 Optimization

The eCard-API-Framework is described in the *Web Service Definition Language* (WSDL). JAXWS [40] for example includes a tool which automatically generates corresponding Java classes. However, the generated Java classes are not compatible with the Java 1.3 version used for the Java ME platform. Hence, we use the *XML Pull API* [41] to implement the eCard-API messages manually. That results in a lightweight implementation of the eCard-API. The eCard-API implementation in MONA does not add any unnecessary XML namespace definitions like common XML serializers do. That reduces the XML messages up to a third. The optimization should be implemented on the eID server too. Such an optimization is not relevant for computers with high speed Internet, but reduces the data amount and therefore the runtime in a mobile environment.

[2] A video of MONA is available at `https://www.cdc.informatik.tu-darmstadt.de/mona`.

	Nokia 6212	Nokia 6220c	Samsung S8000	HTC Desire
Point multiplication	1051.3	624.6	763.7	305.2
Point doubling	2.3	2.0	1.8	0.7
Point addition	3.8	1.9	3.0	1.5

Table 2: Benchmarks for elliptic curve operations (in msec)

3 Future Work

We have seen a considerable runtime concerning the eID functionality (cf. Table 1). A major part of the runtime results from the resource restrictions of the Nokia 6212. First runtime measurements depicted in Table 2 show, that more powerful devices enable a significantly lower runtime. We expect to reach an overall runtime comparable to the runtime on computers, when using the most up to date mobile devices. As the Android platform is promising and many upcoming Android phones support the NFC functionality, we are about to port our eID client to this platform. This is to be done in a planned open source project. In parallel, we adapt our implementation to the newest protocol specifications, such that it is applicable with the currently released eID cards.

Further work will be done on the implementation of the PIN management, thus providing a full alternative to the AusweisApp concerning the eID functionality.

The support of the signature functionality is also an interesting task, in order to be able to support the complete functionality of the German ID card. To achieve the security requirements for the signature functionality, the approach of a distributed reader realization described in [19] seams promising. The modular design of our application allows for an easy reuse and adoption to such a distributed environment. Another approach to enable the signature functionality, will be to consider security modules for secure key storage and execution of the application.

4 Conclusion

We saw that the combination of contactless (national) ID cards, here the German ID card, with NFC-enabled mobile phones is a promising approach. The fact that the involved specifications and standards were not made with this use case in mind, results in many technical obstacles. These lead experts and authorities to believe that the mobile use of the German ID card is not possible at the moment. However, the presented prototype called MONA proves that these obstacles can be mastered in a way that even allows for a maximum of modularity and platform independence. Although the applied mobile phone is an outdated low price model, the overall runtime is tolerable. First tests with modern high end phones indicate that MONA executes with a comfortable runtime on this class of devices. As MONA also runs on computers, and is about to become an open source project, it has the potential to become a full-fledged alternative to the official eID application AusweisApp.

Acknowledgements

We would like to thank to *Telekom Innovation Laboratories*, *T-Systems* and *Media Transfer AG* for the cooperation and support to this research. We are grateful to J. Schaaf for his commitment.

References

[1] International Civil Aviation Organization (ICAO). Machine Readable Travel Documents. ICAO Doc 9303, Part 1-3, 2006 - 2008. http://www2.icao.int/en/MRTD/Pages/Doc9393.aspx.

[2] Comité Européen de Normalisation (CEN). Identification card systems - European Citizen Card - Part 1-4, CEN/TS 15480. Technical Specification, 2008.

[3] ISO/IEC. Identification cards – Contactless integrated circuit cards – Proximity cards. International Standard, ISO/IEC 14443, Part 1-4, 2008 - 2011.

[4] ISO/IEC. Information technology – Telecommunications and information exchange between systems – Near Field Communication – Interface and Protocol (NFCIP-1). International Standard, ISO/IEC 18092, 2004.

[5] ISO/IEC. Information technology – Telecommunications and information exchange between systems – Near Field Communication Interface and Protocol -2 (NFCIP-2). International Standard, ISO/IEC 21481, 2005.

[6] Federal Office for Information Security. Advanced Security Mechanism for Machine Readable Travel Documents - Extended Access Control (EAC), Password Authenticated Connection Establishment (PACE), and Restricted Identification (RI). Technical Guideline BSI-TR-03110, Version 2.05, 2010. https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TR03110/TR-03110_v205_pdf.pdf.

[7] Jens Bender, Marc Fischlin, and Dennis Kügler. Security Analysis of the PACE Key-Agreement Protocol. In *Information Security Conference (ISC) 2009*, volume 5735 of *Lecture Notes in Computer Science*, pages 33–48. Springer, Sep 2009.

[8] Özgür Dagdelen and Marc Fischlin. Security Analysis of the Extended Access Control Protocol for Machine Readable Travel Documents. In *13th Information Security Conference*, Lecture Notes in Computer Science. Springer, Okt 2010.

[9] Federal Office for Information Security. AusweisApp. https://www.ausweisapp.bund.de.

[10] Moritz Horsch. Mobile Authentisierung mit dem neuen Personalausweis (MONA). Master Thesis, TU Darmstadt, Jul 2011. http://www-old.cdc.informatik.tu-darmstadt.de/reports/reports/Moritz_Horsch_MONA.master.pdf.

[11] AGETO Holding AG. AGETO AusweisApp, 2011. http://www.ageto.de/egovernment/ageto-ausweis-app.

[12] Bremen Online Services. Governikus autent. http://www.bos-bremen.de/de/governikus_autent/1854605/.

[13] Moritz Horsch. MobilePACE - Password Authenticated Connection Establishment implementation on mobile devices. Bachelor Thesis, TU Darmstadt, September 2009. http://www.cdc.informatik.tu-darmstadt.de/reports/reports/Moritz_Horsch.bachelor.pdf.

[14] Alex Wiesmaier, Moritz Horsch, Johannes Braun, Franziskus Kiefer, Detlef Hühnlein, Falko Strenzke, and Johannes Buchmann. An efficient PACE Implementation for mobile Devices. In *ASIA CCS '11: 6th ACM Symposium on Information, Computer and Communications Security*, March 2011.

[15] Franziskus Kiefer. Effiziente Implementierung des PACE- und EAC-Protokolls für mobile Geräte. Bachelor Thesis, TU Darmstadt, July 2010.

[16] Frank Morgner, Dominik Oepen, and Wolf Müller. Crypto library for the PACE protocol, 2011. http://sourceforge.net/projects/openpace.

[17] T. Senger. Androsmex project a mobile smart card explorer for android smartphones with nfc capabilities. http://code.google.com/p/androsmex/.

[18] Detlef Hühnlein et al. On the design and implementation of a lightweight and open eid-client. 2011. to be published.

[19] Johannes Braun, Moritz Horsch, Alexander Wiesmaier, and Detlef Hühnlein. Mobile authentisierung und signatur. In *D-A-CH Security 2011*, Sep 2011.

[20] Johannes Braun, Moritz Horsch, and Alexander Wiesmaier. ipin and mtan for secure eid applications. 2011. to be published.

[21] Federal Office for Information Security. Technical guidelines and protection profiles regarding electronic ID documents. https://www.bsi.bund.de/EN/Topics/ElectrIDDocuments/TRandSecurProfiles/TRSpec/trspec_node.html.

[22] Federal Office for Information Security. eCard-API-Framework. Technical Guideline BSI-TR-03112, Version 1.1.1, Part 1-7, 2011. https://www.bsi.bund.de/cln_156/ContentBSI/Publikationen/TechnischeRichtlinien/tr03112/index_htm.html.

[23] Moritz Horsch and Martin Stopczynski. The german ecard-strategy. Student research project, TU Darmstadt, Feb 2011. Technical Report: TI-11/01.

[24] Federal Office for Information Security. Anforderungen an Chipkartenleser mit nPA Unterstützung. Technical Guideline BSI-TR-03119, Version 1.2, 2011. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR03119/BSI-TR-03119_V1_pdf.pdf.

[25] Federal Office for Information Security. Architektur elektronischer Personalausweis und elektronischer Aufenthaltstitel. Technical Guideline BSI-TR-03127, Version 1.14,

2011. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR03127/BSI-TR-03127_pdf.pdf.

[26] Don Box, David Ehnebuske, Gopal Kakivaya, Andrew Layman, Noah Mendelsohn, Henrik Frystyk Nielsen, Satish Thatte, and Dave Winer. Simple Object Access Protocol (SOAP) 1.1, 2000. http://www.w3.org/TR/2000/NOTE-SOAP-20000508.

[27] Robert Aarts and John Kemp. Liberty Reverse HTTP Binding for SOAP Specification. Liberty Alliance Specification, Version 2.0, 2006. http://www.projectliberty.org/liberty/content/download/909/6303/file/liberty-paos-v2.0.pdf.

[28] P. Eronen and H. Tschofenig. Pre-Shared Key Ciphersuites for Transport Layer Security (TLS). Request For Comments – RFC 4279, December 2005. http://www.ietf.org/rfc/rfc4279.txt.

[29] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.1. Request For Comments – RFC 4346, April 2006. http://www.ietf.org/rfc/rfc4346.txt.

[30] ISO/IEC. Identification cards – Integrated circuit cards. International Standard, ISO/IEC 7816, Part 1-13,15, 1999 - 2011.

[31] Oracle Corporation. JSR 257: Contactless Communication API. http://jcp.org/en/jsr/detail?id=257, 2009.

[32] P. Lubbers and F. Greco, Kaazing Corporation. HTML5 Web Sockets: A Quantum Leap in Scalability for the Web, 2010. http://www.websockets.org/quantum.html.

[33] Oracle Corporation. JSR 268: Java Smart Card I/O API. http://jcp.org/en/jsr/detail?id=268, 2006.

[34] Erik Tews. Entwicklung von MicroTLS als sichere Ende-zu-Ende-Verbindung über Bluetooth und TCP/IP, 2006.

[35] Oracle Corporation. JSR 118: Mobile Information Device Profile 2.0. http://jcp.org/en/jsr/detail?id=118, 2006.

[36] Oracle Corporation. JSR 139: Connected Limited Device Configuration 1.1. http://jcp.org/en/jsr/detail?id=139, 2007.

[37] Nokia Corporation. Nokia 6212 classic. http://europe.nokia.com/support/product-support/nokia-6212-classic.

[38] Technische Universität Darmstadt. FlexiProvider. http://www.flexiprovider.de.

[39] Oracle Corporation. Light Weight User Interface Toolkit (LWUIT), 2010. http://www.oracle.com/technetwork/java/javame/tech/lwuit-141954.html.

[40] Oracle Corporation. Java API for XML Web Services (JAX-WS). http://jax-ws.java.net, 2011.

[41] Stefan Haustein and Aleksander Slominski. XML Pull API, 2005. http://www.xmlpull.org.