Mobile Devices as Secure eID Reader using Trusted Execution Environments

Maximilian Stein

secunet Security Networks AG
Alt-Moabit 91c
10559 Berlin
Maximilian.Stein@secunet.com

Abstract: This work presents a prototype implementation of a smartphone as secure eID reader using NFC technology. The presented approach aims to reach a security level close to standalone smart card readers. This security level will be allowed by the means of a trusted execution environment (TEE) which allows strong isolation and separation for critical applications and provides trusted, not interceptable user input and output. The prototype supports the German eID (nPA) and follows the relevant guidelines.

1 Introduction

Mobile internet devices (smartphones, tablets) have become the omnipresent companion in the modern society. The capabilities and processing power of today's devices is enormous. Especially high-end devices featuring quad-core CPUs and high definition graphics can compete easily with mid-range PC systems while being smaller, more energy-efficient and portable. They can satisfy nearly all needs of an ordinary PC user like access to the internet, e-mail and social networks, music and video playback or other entertainment. In the long run mobile internet devices may replace the PC for such users and use cases completely.

National electronic ID cards are emerging slowly but surely. In Germany there are yet few citizens using their eID in online processes and there are still not many applications available. Nevertheless electronic IDs are believed to become more important and will be essential in future citizenship. To gain more acceptance from citizens though, it is important to provide low-threshold access to technology and knowledge for the usage of electronic IDs. The necessity of an additional, expensive reader device to make use of eID cards is not likely going to raise acceptance.

Since a few years mobile internet devices feature near-filed-communication (NFC) technology. Among others the NFC specification is based on the standard for contactless smart cards [ISO11]. Therefore NFC-devices are technically enabled to communicate with proximity cards like national eIDs. By this they can be used as card reader for eIDs and other smart cards. This has already been implemented for the German eID in [Hor11]

and [Mor12]. Both showed a proof of concept that the eID can be accessed using the PACE protocol through a NFC-enabled mobile phone (Nokia 6212 & Openmoko Neo FreeRunner customized smartphone). The security established through PACE depends on the secure input of the PIN on the device. The security in both approaches relies only on the assumption that the used mobile devices are trustworthy and no user input can be intercepted by a software of an attacker. However, since mobile devices gained popularity, more sophisticated attacks and malware for such devices emerged. For this reason smartphones and tablets have to be regarded as potentially untrustworthy and malicious.

The present work presents an approach to use a NFC-enabled mobile internet device as secure embedded reader for the German eID card by using a trusted execution environment (TEE). The remainder of this work is structured as follows. Section 2 briefly presents related work. In section 3 the basic principle of a trusted execution environment is described. Section 4 presents the current embedded smart phone reader implementation. Finally section 5 concludes this paper.

2 Related Work

Horsch [Hor11] implemented the eID application MONA as Java MIDlet on a Nokia 6212. It is capable of performing an online authentication with the German eID card. In [Mor12] Morgner implemented an embedded eID reader with PACE support on an Openmoko Neo FreeRunner customized smartphone with SHR Linux operating system. The implementation relies on OpenPACE [MO12], an open source implementation of PACE based on OpenSSL. An efficient implementation of the PACE protocol for mobile devices has been proposed in [WHB+11]. Alternative solutions for the security concerns regarding the mobile use of eID and eSignature were proposed in [BHW12] and [BHWH11], respectively. An open source eID application for the German eID for Android devices is available through the Open eCard project [Ope12]. This app supports multiple methods to access the eID card. It is possible to use an external reader or the internal NFC interface, if available. The Governikus Autent PINApp [bre12] provides PIN management functionalities on Android devices for the German eID. The NFC Tag Info app [Hag13] is capable of reading electronic passports (eMRTDs) via basic access control (BAC) but does not provide PACE capabilities.

3 Trusted Execution Environment

A trusted execution environment (TEE) is a separate execution environment that runs alongside the Rich OS (i.e. regular mobile device OS). The TEE provides security services for the rich environment and isolates access to its hardware and software security resources from the Rich OS and its applications [Glo11].

Figure 1 depicts the TEE architecture as envisioned by the GlobalPlatform industry forum. It was designed for mobile and embedded devices but could be used for PCs as well if

all requirements are met. The three depicted TEE APIs in figure 1 were specified by GlobalPlatform in 2010 and 2011, respectively.

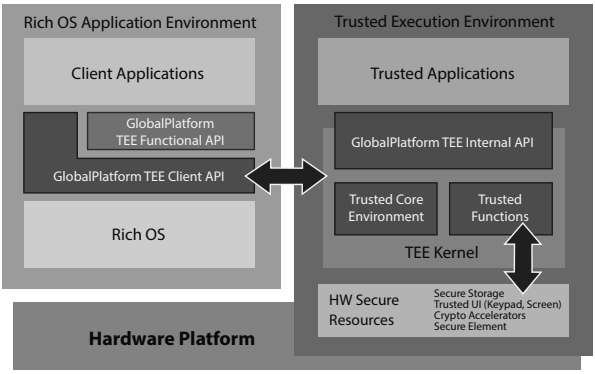

Figure 1: Architecture of the TEE as specified by GlobalPlatform

The security of the TEE relies on hardware extensions that help isolating the two environments. The hardware architecture to enable TEEs on ARM based mobile devices is the ARM TrustZone technology [ARM02][1]. The TrustZone extensions are integrated into the SoC and comprise an address space controller, memory adaptor, interrupt controller, reserved secure cache, and hardware keys. These features are available for ARM Cortex-A processors if and only if they were included by the SoC manufacturer. Secure operating systems can be implemented on top of the TrustZone hardware[2]. The physical CPU is presented as two virtual CPUs to the secure OS via TrustZone. One CPU dedicated to the rich environment and the other one to the trusted environment. In TrustZone terminology the rich OS application environment is referred to as Normal World (NWd) and the TEE as Secure World (SWd). The secure operating system controls the virtualisation, the security extensions, and provides the TEE.

A TEE can host multiple trusted applications. These applications are executed in the trusted environment where the TEE guarantees isolated execution between different trusted applications, protection, confidentiality and integrity. Trusted application binaries are stored in the file system as cryptogram and they are verified by the TEE each time before their execution. The root of trust for the TEE is established at boot time through a chain of trust: a hardware anchor verifies the boot loader which in turn verifies the TEE loader which verifies the TEE system image.

Current TEEs based on ARM TrustZone were MobiCore by Giesecke & Devrient and Trusted Foundations by Trusted Logic Mobile. However, both decided to merge their products in a joint venture named Trustonic[3] together with ARM. The TEE developed

[1]Other hardware architectures with similar features are for example Aegis, XOM, and SecureCore.

[2]TEEs can be implemented through pure software virtualisation as well (XenARM, SIVARM), but lack the additional security through hardware support.

[3]http://www.trustonic.com/about-us/who-we-are

by Trustonic is called <t-base. Sierraware implemented the SierraTEE and SierraVisor TEE solution which is freely available under the GNU GPL v2 license for the Samsung Exynos 4412 and nVIDIA Tegra 3 SoCs. So far Giesecke & Devrient's MobiCore was integrated in the Samsung Galaxy S3 and the Galaxy Note II. Since Samsung is hardware integrator and device maker partner of Trustonic it can be expected that Samsung will integrate <t-base in upcoming high-end devices too.

4 Implementation

The prototype device used for the implementation is a Samsung Galaxy S3 (GT-i9300) NFC-enabled smartphone running Android 4.1.1. The device combines all necessary components for the use of an eID card in one entity. Table 1 shows the analogy of components in the mobile eID reader system.

Original Component	Counterpart in Mobile Scenario
Host Computer	GT-i9300 NWd with Android
eID Application	Android App (e.g. Open eCard)
eID Reader Hardware	GT-i9300 SWd virtual CPU
eID Reader Firmware	Trusted Application (Trustlet)
eID Reader Driver	Trustlet Connector

Table 1: Analogy of components in the embedded eID reader system

A regular smartphone has the same capabilities of using an eID securely as a regular PC. It requires a smart card reader with a PIN pad, that is connected via the USB interface, and it needs to run an eID application. The already mentioned Open eCard project provides such an open source eID application for the Android OS. It is capable of using external smart card readers via USB and the internal NFC interface.

The here described eID reader implementation is an embedded smart card reader, which consists of a firmware part and a driver part. So far, this is identical to regular standalone eID readers. The difference is, that the firmware of a standalone reader resides inside the reader hardware. As shown in table 1, the reader hardware in this approach is physically the same as the host computers hardware, therefore it is called an embedded eID reader.

The smartphone is split up into two virtual devices by the TEE. The eID application resides in the so called normal world with the Android OS. The embedded reader firmware resides in the secure world. By this, the embedded reader can be treated as if it had its own separated hardware. The implemented prototype can be categorised as seen in table 2. The depicted categorisation for the reader categories Cat-S and Cat-C is taken from [BSI13] and shows the properties an eID reader has to have to be categorized as *standard reader* (Cat-S) or *comfort reader* (Cat-C). The prototype currently implements a *standard reader* with an additional display. So it can be categorized as Cat-S with additional functionality. Regardless of certification issues, the prototype can be enhanced in future to implement

	Cat-S	Cat-S$^+$	Cat-C
Interface to the host computer	✓	✓	✓
Contactless interface according to ISO/IEC 14443	✓	✓	✓
Contact interface according to ISO/IEC 7816			✓
PIN pad (secure PIN entry) with PACE support	✓	✓	✓
Display (2x16 alpha-numeric characters)		✓	✓
Qualified signature with contact cards			✓
Qualified signature with contactless cards (e.g. identity card)		✓	✓
Firmware update	✓	✓	✓

Table 2: Overview of Smart Card Reader Categories (source: [BSI13])

the properties of a signature terminal. In this way it implements the same properties as a *comfort reader* only without a contact interface. As it is unlikely that smartphones will be equipped with contact interfaces for smart cards, embedded readers like the presented prototype will only be capable to implement the properties that are presented here as Cat-S$^+$.[4]

The system architecture of the embedded reader and the associated components is depicted in figure 2. The shown eID reader **Trustlet** represents the reader firmware. The

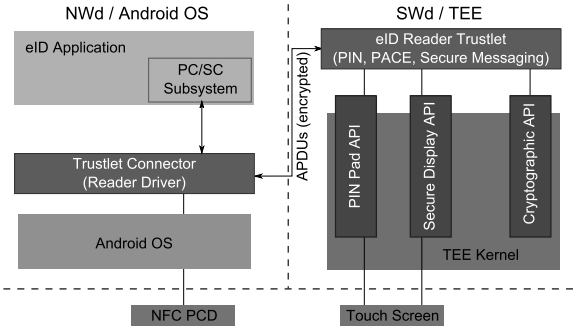

Figure 2: Architecture for a device using the eID Reader Trustlet

reader driver is implemented through the so called **Trustlet Connector**. The Trustlet Connector implements the PC/SC IFD handler interface through which it can be accessed by any application, that is PC/SC aware. Generally the Trustlet Connector provides the security services of the Trustlet to regular applications in the NWd. Any information that is proccessed inside the Trustlet will not be accessible for NWd applications unless provided through the Trustlet Connector. The PIN Pad API, the Secure Display API, and the Cryptographic API are provided by the TEE itself. The PIN Pad and Secure Display API together provide a trusted user interface which is immune to interception and manipulation by other software.

[4]Please note that no new reader category is proposed here. The name Cat-S$^+$ is only chosen to symbolize the enhanced functionality compared to a Cat-S reader.

4.1 Trustlet

The embedded eID reader firmware is implemented as trusted application – a so called **Trustlet** as depicted in figure 2. The TEE is capable to isolate the execution of Trustlets and grants access to a secure user interface. In this way it is guaranteed that: (i) all processing results and the execution of the Trustlet itself is safe to interception and manipulation by malicious software (neither Android Apps nor other Trustlets), and (ii) a PIN can be entered directly inside the protected environment.[5]

The Trustlet implements the PACE protocol by using the internal cryptographic API that is provided by the TEE. The NFC interface is not (yet) available for Trustlets due to lack of driver support by the TEE. However, the GlobalPlatform TEE roadmap shows that additional peripheral devices like the NFC interface will be included in future versions of the specification. Currently the APDUs are transmitted from the inside of the TEE to the eID card through the Trustlet Connector via the Android NFC API. However, all secret information and processes of the PACE protocol – the PIN, key material and key generation – are isolated inside the TEE. The APDUs are transmitted encrypted through secure messaging between the endpoints Trustlet and eID. The security of this solution relies on the security of the PACE protocol. This is because the interface between the Trustlet and the eID can be assumed to be as insecure as the air interface in a regular PACE establishment process. The encrypted APDUs are interceptable from the NWd, because they are forwarded by the Trustlet Connector and the Android NFC API. Since the PACE protocol for key-agreement has been proven to be secure [BFK09], the implemented transmission of APDUs via the normal world can be considered to be secure as well.

It is intended to implement extended access control (EAC v2) in the future to use the smartphone as signature terminal for the German eID as well[6]. The mobile eID reader can reach a security level comparable to a physically separated standalone card reader device through the hardware backed detachment of NWd and SWd. A security and conformity certification according to the technical guidelines of the German Federal Office for Information Security [BSI13, BSI11] seems possible at the moment. However, this highly relies on the certifications for TrustZone hardware implementations and TEE systems, which is a future challenge.

4.2 Trustlet Connector

To access the eID Reader Trustlet from any application in the NWd, a counterpart is required – the so called **Trustlet Connector**. In the prototype implementation the Trustlet Connector is an Android app that bundles two native C/C++ libraries, the native NFC wrapper and the actual Trustlet Connector library.

The Trustlet Connector is a standalone Android application with access to the Android NFC API. This is necessary because the Android NFC API is only accessible from Android

[5] As of the writing of this paper the secure UI functionality is not yet available, see section 4.3

[6] The certification of such a solution poses a greater challenge than the actual implementation.

applications with the appropriate Android permissions. It can not be accessed directly by native C/C++ libraries or executables. The Android app "catches" the eID when it is placed on the device and provides access to it for the Trustlet Connector library. This is achieved by providing transmission methods to the native NFC wrapper library via JNI. This native wrapper provides a RPC server for the actual Trustlet Connector library. It should be mentioned, that the NFC interface can not be used to actively poll for contactless cards and to initialize connections manually. This functionality is encapsulated by the Android NFC framework. Therefore apps can only wait for the NFC API to notify them, once an eID – or some other NFC tag – is available. This implies, that it is impossible for the embedded reader to power down the NFC interface or reset the connection with a present card. The eID has to be moved from the device and then replaced manually to achieve this.

The Trustlet Connector library implements the PC/SC IFD handler interface. The IFD handler is the driver of the embedded reader for the PC/SC interface. The IFD handler is loaded by the PC/SC daemon, which makes the reader available to any PC/SC aware application. In a regular PC environment this would be enough to provide access to the embedded reader. But there exists no standard implementation of PC/SC for Android. Therefore applications that rely on PC/SC may include the PC/SC library and daemon locally. This is the case for the Open eCard App. In this case, the location of the driver for the embedded reader has to be provided to the PC/SC daemon by a configuration file, to make the reader available for the application. For the prototype, the corresponding class of the Open eCard App was customized and a configuration file for the embedded reader was added. The Trustlet Connector library resides in the data folder of the Trustlet Connector app, the configuration file only points to the location of the driver library.

The Trustlet Connector library further contains an interface for the communication with the Trustlet. This interface uses the systems TEE driver to communicate with the Trustlet. The functional interface between the Trustlet and its Trustlet Connector can be defined freely. Basically both components have access to a shared buffer and are able to notify each other, if the content of this buffer has changed. Through this buffer, a RPC interface was implemented to allow the Trustlet to execute specific functions of the Trustlet Connector and vice versa. It should be noted that the shared buffer is not protected in any special way, nor does it reside inside the TEE. Therefore no unencrypted secret information should be written to it.

In short the Trustlet Connector app works as follows. When a contactless card is placed on the NFC interface, the Trustlet Connector app will be notified by the Android event manager and can be chosen to handle the event by the user. It will check if the present card is a German eID and start a background service that will listen to the native NFC wrapper library. The NFC interface can now be accessed via the native wrapper RPC server. If a PC/SC daemon has already loaded the Trustlet Connector library, the IFD handler will be informed, that there is a card present at the NFC interface. A PC/SC aware application is now able to use the embedded eID reader.

The Trutlet Connector library is the hub of the implementation. It handles the function calls from the PC/SC interface, from and to the Trustlet and to the native NFC wrapper.

4.3 Discussion

It has to be noted that the security of the overall implementation is highly depending on the secure and correct implementation of the TEE and the proper implementation of the Trustlet itself. For example it is crucial to implement the interface between the Trustlet and the Trustlet Connector very carefully. Since the Trustlet Connector resides in the NWd, it could be replaced by a malicious Trustlet Connector which tries to read secret information from the Trustlet by manipulating pointer locations or input data.

The implemented reader was successfully tested to be usable by the Open eCard Android app as external reader via PC/SC. The integration of the secure reader into an eID application requires some effort as there is not yet a default smart card reader interface for the Android OS.[7] The integration via PC/SC allows the usage of the embedded reader on unrooted off-the-shelf devices, because no special access rights like for the USB interface are required for the application.

As of the writing of this paper the author is not aware of any off-the-shelf smartphones with support for extended length APDUs. This is the same for the prototype device. Therefore the prototype only allows PIN management functionalities for the German eID and is not capable to perform an online authentication.

As of the writing of this paper the trusted UI functionality is not yet available, but is expected to be ready soon. The prototype therefore contains workarounds. In its current form it can not be regarded as secure or trustworthy because the trusted user interface, especially the secure PIN entry is the main feature that prevents unauthorized access to the eID. As a workaround, for each of the two UI APIs an Android Activity was implemented to simulate a PIN Pad and a "Secure" Display, respectively. When the Trustlet normally would access the secure APIs, it currently calls the Trustlet Connector to start the appropriate Android Activity and to either return the entered PIN or display the given certificate holder information.

5 Conclusion & Future Work

The present work showed the possibilities of using mobile internet devices as trustworthy and secure card reader for eIDs. It further gave a short introduction to trusted execution environments for mobile platforms as specified by the GlobalPlatform industry forum. The general concept and advantages of a TEE were described. It was presented how an embedded eID reader was implemented on an unmodified (but rooted) Samsung Galaxy S3 using a TEE. With the given implementation it is possible to use the eID in a mobile scenario, meaning that it is accessed by applications residing on the mobile device itself. Subject of future work is the usage of the smartphone eID reader as external reader for PC systems. Furthermore the conformity and security certifications of the embedded reader implementation pose the next steps in this working field.

[7] An implementation of a smart card reader API has been achieved by the SEEK for Android project, but is not applicable due to system manufacturer restrictions

References

[ARM02] ARM Ltd. TrustZone® technology. http://www.arm.com/products/processors/technologies/trustzone.php, 2002.

[BFK09] Jens Bender, Marc Fischlin, and Dennis Kügler. Security Analysis of the PACE Key-Agreement Protocol. In Pierangela Samarati, Moti Yung, Fabio Martinelli, and ClaudioA. Ardagna, editors, *Information Security*, volume 5735 of *Lecture Notes in Computer Science*, pages 33–48. Springer Berlin Heidelberg, 2009.

[BHW12] Johannes Braun, Moritz Horsch, and Alexander Wiesmaier. iPIN and mTAN for Secure eID Applications. In MarkD. Ryan, Ben Smyth, and Guilin Wang, editors, *Information Security Practice and Experience*, volume 7232 of *Lecture Notes in Computer Science*, pages 259–276. Springer Berlin Heidelberg, 2012.

[BHWH11] Johannes Braun, Moritz Horsch, Alexander Wiesmaier, and Detlef Hühnlein. Mobile Authentisierung und Signatur. In Peter Schartner and Jrgen Taeger, editors, *D-A-CH Security 2011: Bestandsaufnahme, Konzepte, Anwendungen, Perspektiven*, pages 32–43. syssec Verlag, sep 2011.

[bre12] bremen online services GmbH & Co. KG. Governikus Autent PINApp. https://play.google.com/store/apps/details?id=de.bos_bremen.android.autent.pinapp, 2012.

[BSI11] BSI – Federal Office for Information Security. Technical Guideline BSI TR-03105 Part 5.2: Test plan for eID and eSign compliant eCard reader systems with EAC 2, 2011.

[BSI13] BSI – Federal Office for Information Security. Technical Guideline BSI TR-03119: Requirements for Smart Card Readers Supporting eID and eSign Based on Extended Access Control, 2013.

[Glo11] GlobalPlatform. The Trusted Execution Environment, White Paper. http://www.globalplatform.org/documents/GlobalPlatform_TEE_White_Paper_Feb2011.pdf, 2011.

[Hag13] NFC Research Lab Hagenberg. NFC TagInfo. https://play.google.com/store/apps/details?id=at.mroland.android.apps.nfctaginfo, 2013.

[Hor11] Moritz Horsch. Mobile Authentisierung mit dem neuen Personalausweis (MONA). Master's thesis, Technische Universität Darmstadt, Darmstadt, 2011.

[ISO11] ISO/IEC. ISO 14443: Identification cards – Contactless integrated circuit cards – Proximity cards, 2011.

[MO12] Frank Morgner and Dominik Oepen. OpenPACE: Crypto library for the PACE protocol. http://openpace.sourceforge.net/, 2012.

[Mor12] Frank Morgner. Mobiler Chipkartenleser für den neuen Personalausweis: Sicherheitsanalyse und Erweiterung des „Systems nPA“. Diploma thesis, Humboldt-Universtität zu Berlin, Berlin, 2012.

[Ope12] Open eCard Project. https://www.openecard.org, 2012.

[WHB+11] Alex Wiesmaier, Moritz Horsch, Johannes Braun, Franziskus Kiefer, Detlef Hühnlein, Falko Strenzke, and Johannes Buchmann. An efficient mobile PACE implementation. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, ASIACCS '11, pages 176–185, New York, NY, USA, 2011. ACM.