Transactional Mobile Government

Leveraging Qualified Electronic Signatures
on Mobile Devices

Thomas Zefferer

Dipl.-Ing. Thomas Zefferer Bakk.techn.

Transactional Mobile Government

Leveraging Qualified Electronic Signatures on Mobile Devices

DOCTORAL THESIS

to achieve the university degree of

Doktor der technischen Wissenschaften

submitted to

Graz University of Technology

Tutor

O.Univ.-Prof. Dipl.-Ing. Dr.techn. Reinhard Posch

Institute for Applied Information Processing and Communications
Faculty of Computer Science and Biomedical Engineering

Graz, July 2015

Dipl.-Ing. Thomas Zefferer Bakk.techn.

Transaktionales Mobile Government

Qualifizierte elektronische Signaturen auf mobilen Geräten

DISSERTATION

zur Erlangung des akademischen Grades

Doktor der technischen Wissenschaften

eingereicht an der

Technischen Universität Graz

Betreuer

O.Univ.-Prof. Dipl.-Ing. Dr.techn. Reinhard Posch

Institut für Angewandte Informationsverarbeitung und Kommunikationstechnologie
Fakultät für Informatik und Biomedizinische Technik

Graz, Juli 2015

Abstract

Driven by the aim to reduce costs and to improve efficiency, governments and public administrations provide electronic services to spare citizens cumbersome paper-based procedures. These services have become known under the term electronic government or e-government. During the past few years, providers of e-government services have been faced with a new mobile computing paradigm. This new paradigm has yielded the typical always-on society, in which people expect to have access to information and services everywhere and at any time. As smartphones and tablet computers are enablers of always-on societies, provided e-government services need to be adapted, in order to be applicable on these devices. In the e-government domain, the provision of services for mobile devices has become known under the term mobile government or m-government. The transition from e-government to m-government is crucial for governments and public administrations, in order to keep pace with technological progress and to meet requirements of the predominating mobile computing paradigm.

The provision of m-government services for mobile end-user devices turns out to be problematic, if these services require the user to create legally binding electronic signatures. This is the case for so-called transactional services that require the user to authenticate and to provide written consent in electronic form. During the past years, various signature solutions have been developed that enable users to create legally binding electronic signatures. However, these solutions have been tailored to classical end-user devices and raise several issues when being applied on mobile devices. Limited hardware capabilities of mobile devices and incompatible security concepts of signature solutions are the main reasons that prevent an application of these solutions on smartphones and tablet computers. The current lack of solutions for the creation of legally binding electronic signatures on mobile end-user devices currently represents the main obstacle to transactional m-government.

To overcome this obstacle, a mobile signature solution for mobile end-user devices is proposed in this thesis. The proposed solution is developed following an elaborate methodology. Taking into account identified success factors of m-government, the most suitable general architecture is determined first. Then, this architecture is successively refined and further developed towards the proposed solution. The proposed solution is intentionally kept on an abstract and technology-agnostic level to assure its sustainability and capability to respond to future technological changes. Finally, the proposed abstract solution is evaluated by further developing it towards a concrete solution and by implementing it using three different mobile state-of-the-art technologies. This shows that despite its abstract nature, the proposed solution is a suitable basis for concrete realizations.

Both the proposed abstract solution and the provided implementation represent an important step towards transactional m-government. They show that creating legally binding electronic signatures is feasible on mobile end-user devices and that respective solutions can already be realized with available technologies. This way, this thesis facilitates the transition from e-government to m-government and assures that provided services comply with the emerging mobile computing paradigm and are able to meet the requirements and expectations of the current always-on society.

Kurzfassung

Zur Erhöhung der Effizienz und zur Reduktion von Kosten setzen Regierungen und öffentliche Verwaltungseinheiten verstärkt auf elektronische Dienste, auch um Bürgerinnen und Bürgern mühsame Behördenwege zu ersparen. Als Sammelbegriff für derartige Dienste hat sich der Begriff E-Government durchgesetzt. Während der letzten Jahre waren Anbieter von E-Government-Diensten mit einer neuen Art und Weise der mobilen Nutzung von Computern konfrontiert, welche übliche Methoden des Zugriffs auf Informationen und Dienste nachhaltig geändert hat. Dies resultierte schließlich in der heute üblichen Always-On-Gesellschaft, in der davon ausgegangen wird, dass Informationen und Dienste kontextunabhängig jederzeit verfügbar sind. Da Smartphones und Tablet-Computer dafür Wegbereiter sind, müssen auch bestehende E-Government-Dienste entsprechend adaptiert werden, um auf diesen Geräten verwendet werden zu können. Für die Bereitstellung von E-Government-Diensten auf mobilen Geräten hat sich der Begriff M-Government etabliert. Ein Übergang von E-Government hin zu M-Government ist für öffentliche Verwaltungen wichtig und notwendig, um mit technischen Weiterentwicklungen Schritt zu halten und um dem geänderten Nutzungsverhalten gerecht werden zu können.

Die Bereitstellung von E-Government-Diensten für mobile Endnutzergeräte stellt sich als schwierig heraus, wenn diese die Erbringung rechtsgültiger elektronischer Unterschriften von Benutzerinnen und Benutzern erfordern. Dies betrifft im Wesentlichen transaktionale Dienste, die eine Benutzerauthentifizierung und die Erbringung einer bindenden schriftlichen Zustimmung verlangen. In letzter Zeit wurden zahlreiche Lösungen entwickelt, die Benutzerinnen und Benutzern die Erstellung rechtsgültiger elektronischer Signaturen erlauben. Diese Lösungen wurden jedoch für klassische Endnutzergeräte entworfen, wodurch sich bei deren Verwendung auf mobilen Endnutzergeräten zahlreiche Probleme ergeben. Diese Probleme werden in den meisten Fällen durch beschränkte Hardware mobiler Geräte und durch für diese Geräte ungeeignete Sicherheitskonzepte verursacht. Die aktuelle Nichtverfügbarkeit von geeigneten Lösungen zur Erstellung rechtsgültiger elektronischer Signaturen auf mobilen Endgeräten kann als größte Hürde für transaktionale M-Government-Dienste identifiziert werden.

Als Lösung für dieses Problem wird in dieser Arbeit eine Signaturlösung für mobile Endnutzergeräte vorgeschlagen. Diese wird entsprechend einer wohlüberlegten Methodik entwickelt. Unter Beachtung identifizierter Erfolgsfaktoren wird zunächst die bestgeeignete Architektur der vorgeschlagenen Lösung bestimmt. In weiterer Folge wird diese Architektur schrittweise verfeinert, woraus sich schließlich die vorgeschlagene Signaturlösung ergibt. Diese wird bewusst abstrakt und technologieunabhängig gehalten, um ihre Nachhaltigkeit und Flexibilität in Bezug auf sich ändernde technologische Rahmenbedingungen zu gewährleisten. Die vorgeschlagene Lösung wird schließlich evaluiert, indem diese zunächst in eine konkrete Lösung übergeführt und schließlich unter Verwendung verschiedener aktuell verfügbarer mobiler Technologien realisiert wird. Auf diese Weise wird gezeigt, dass die vorgeschlagene Lösung trotz ihrer abstrakten Natur eine geeignete Basis für die Entwicklung konkreter Umsetzungen ist.

Sowohl die vorgeschlagene abstrakte Lösung als auch die konkrete Umsetzung stellen einen wichtigen Schritt hin zu transaktionalem M-Government dar. Sie zeigen, dass die Erstellung rechtsgültiger elektronsicher Unterschriften auf mobilen Endnutzergeräte möglich ist und dass entsprechende Lösungen unter Verwendung verfügbarer Technologien umgesetzt werden können. Dadurch unterstützt diese Arbeit den Übergang von E-Government zu M-Government und gewährleistet, dass bereitgestellte Dienste den Anforderungen heutiger Always-On-Gesellschaften gewachsen sind.

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present doctoral thesis.

SIGNATURE INFORMATION	**Signatory**	DI Thomas Zefferer
	Date/Time-UTC	2015-07-23T14:43:25+02:00
	Verification	Information about the verification of the electronic signature can be found at: https://www.signaturpruefung.gv.at
Note	This document is signed with a qualified electronic signature. According to § 4 art. 1 of the Signature Act it in principle is legally equivalent to a handwritten signature.	

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Dissertation identisch.

SIGNATURE INFORMATION	**Signatory**	DI Thomas Zefferer
	Date/Time-UTC	2015-07-23T14:58:46+02:00
	Verification	Information about the verification of the electronic signature can be found at: https://www.signaturpruefung.gv.at
Note	This document is signed with a qualified electronic signature. According to § 4 art. 1 of the Signature Act it in principle is legally equivalent to a handwritten signature.	

Contents

List of Figures

Acknowledgements

First and foremost, I would like to thank Prof. Reinhard Posch for giving me the opportunity to be part of the Institute for Applied Information Processing and Communications at Graz University of Technology and for making this thesis possible. Furthermore, I want to thank Herbert Leitold for his guidance during several years of work on this thesis and for his valuable feedback.

My work on this thesis has considerably benefited from my colleagues, who have always been a source of inspiration. In particular, I would like to thank Arne Tauber, Peter Teufl, Klaus Stranacher, and Bernd Zwattendorfer for many hours of fruitful discussions and their various valuable inputs. Special thanks also go to my colleagues Christian Maierhofer, Christof Rath, Simon Roth, and Manuel Schallar for being a great help during implementation tasks related to this thesis.

Last but not least, I would like to thank my whole family for their support during the past years. My very special thanks go to Ulli for her patience and understanding. Without her support, this would not have been possible.

Thomas Zefferer
Graz, Austria, July 2015

List of Acronyms

3G	Third Generation
4G	Fourth Generation
A-SIT	Secure Information Technologies Center – Austria
AD	Authentication Data
AES	Advanced Encryption Standard
AIDS	Acquired Immune Deficiency Syndrome
AOP	Aspect-Oriented Programming
API	Application Programming Interface
ATF	Bureau of Alcohol, Tobacco, Firearms and Explosives
B2G	Business-to-Government
BYOD	Bring Your Own Device
C2G	Citizen-to-Government
CA	Certification Authority
CADA	Chinese Aged Diabetic Assistant
CAdES	CMS Advanced Electronic Signatures
CELAC	Collecting and Exchange of Local Agricultural Content
CEN	European Committee for Standardization
COPE	Corporate Owned Personally Enabled
CPU	Central Processing Unit
CWA	CEN Workshop Agreement
DI	Dependency Injection
DPA	Differential Power Analysis
DTBS	Data To Be Signed
e-banking	Electronic Banking
e-business	Electronic Business

e-commerce	Electronic Commerce
e-finance	Electronic Finance
e-governance	Electronic Governance
e-government	Electronic Government
e-payment	Electronic Payment
EAL	Evaluation Assurance Level
ECDSA	Elliptic Curve Digital Signature Algorithm
eID	Electronic Identity
eIDAS	Electronic Identification and Trust Services
ENISA	European Union Agency for Network and Information Security
ETSI	European Telecommunication Standards Institute
EU	European Union
FIDO	Fast Identity Online
G2B	Government-to-Business
G2C	Government-to-Citizen
G2E	Government-to-Employee
G2G	Government-to-Government
G2N	Government-to-Nonprofit
GMM	Gaussian Mixture Model
GPS	Global Positioning System
GSM	Global System for Mobile Communications
HIV	Human Immunodeficiency Virus
HOTP	HMAC-Based One-Time Password Algorithm
HSDPA	High Speed Downlink Packet Access
HSM	Hardware Security Module
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
ICT	Information and Communication Technology
IFRAME	Inline Frame
IMSI	International Mobile Subscriber Identity
IPC	Inter-Process Communication

ISO	International Organization for Standardization
IT	Information Technology
JCE	Java Cryptography Extension
JSF	Java Server Faces
LBS	Location-Based Service
LTE	Long Term Evolution
m-administration	Mobile Administration
m-education	Mobile Education
m-government	Mobile Government
m-health	Mobile Health
MDM	Mobile Device Management
MFCC	Mel Frequency Cepstral Coefficient
MNO	Mobile Network Operator
MOA-SS	Modules for Online Applications - Signature Creation
N2G	Nonprofit-to-Government
NFC	Near Field Communication
NGO	Non-Governmental Organization
OATH	Initiative for Open Authentication
OTP	One-Time Password
PAdES	PDF Advanced Electronic Signatures
PC	Personal Computer
PC/SC	Personal Computer/Smart Card
PDF	Portable Document Format
PIN	Personal Identification Number
PKI	Public Key Infrastructure
QR	Quick Response
QSCD	Qualified Signature Creation Device
RFC	Request For Comments
RFID	Radio-Frequency Identification
RP	Relying Party
RSA	Rivest Shamir Adleman

RTR	Austrian Regulatory Authority for Broadcasting and Telecommunications
SCA	Signature Creation Application
SCD	Signature-Creation Data
SD	Signed Data
SE	Secure Element
SIM	Subscriber Identity Module
SMS	Short Message Service
SMTP	Simple Mail Transfer Protocol
SQRL	Secure Quick Reliable Login
SSCD	Secure Signature Creation Device
SSL	Secure Sockets Layer
STORK	Secure Identity Across Borders Linked
T-LTT	Trusted Location-Time Ticket
TAN	Transaction Number
TLS	Transport Layer Security
TOTP	Time-Based One-Time Password Algorithm
TPM	Trusted Platform Module
TSP	Trusted Service Provider
TTC	Text to Change
TTP	Trusted Third Party
TV	Television
U2F	Universal Second Factor
UEFI	Unified Extensible Firmware Interface
UK	United Kingdom
UMTS	Universal Mobile Telecommunications System
UNICEF	United Nations Children's Fund
URL	Uniform Resource Locator
US	United States
USB	Universal Serial Bus
XAdES	XML Advanced Electronic Signatures
XML	Extensible Markup Language

Part I

Problem Definition

Chapter 1

Introduction

" It is not the strongest of the species that survive, nor the most intelligent, but the one most responsive to change."

[Charles Darwin, English Naturalist and Geologist.]

With his pioneering work on the evolutionary theory, Charles Darwin has contributed significantly to the common understanding of the descending of species. Being published in the 19th century, Darwin's theory on the survival of the fittest is nowadays a matter of common knowledge. Interestingly, the basic idea behind this theory is not restricted to biological species, but applies to a certain extent also to solutions from the Information and Communication Technology (ICT) domain. In the current era of dynamic markets and vast technological advances, available technologies and the current state of the art change frequently. In a similar way, user habits to employ available technologies change as well over time. The recent history has shown that only those technical solutions that are sufficiently responsive to technological changes are able to survive, i.e. to remain successful, in the long term.

A recent example underpinning this observation can be found in the mobile-computing domain. For many years, Symbian OS has been one of the predominating operating systems for mobile phones. With the introduction of smartphones and more powerful mobile operating systems such as Apple iOS[1] or Google Android[2], user demands and habits regarding the use of mobile phones suddenly changed. As Symbian OS was unable to adequately respond to the changed circumstances, it quickly lost market shares [Ziegler, 2011]. This finally resulted in a complete stop of support for the operating system in 2012. Symbian OS can hence be regarded as a prime example showing that the incapability to respond to changing circumstances and requirements can finally drive a successful technical solution to ruin.

The need to adequately respond to changing circumstances applies to virtually all technical solutions. In particular, also Electronic Government (e-government) solutions, which enable the electronic accomplishment of administrative procedures and the electronic interaction with public-sector agencies, need to continuously adapt to new technological advances and to current trends in the ICT domain. During the past years, especially mobile computing has risen as new computing paradigm. This thesis contributes to the sustainable success of e-government solutions by leveraging their use together with emerging mobile-computing technologies. In particular, this thesis proposes, discusses, and evaluates solutions that enable established e-government solutions to respond to changing computing habits caused by the growing popularity of smartphones and related mobile end-user devices. The motivation behind this thesis, its methodology, its relevant contributions, and its structure are presented in the following subsections.

[1] https://www.apple.com/ios/
[2] http://www.android.com/

1.1 Motivation

The past years have yielded various technological innovations, which have had a strong influence on daily life. This especially applies to the Information Technology (IT) sector, where continuous improvements of available end-user devices and communication networks can be observed. Frequent technological innovations in the IT sector have recently led to two trends. First, the restriction to classical end-user devices such as Personal Computers (PCs) or laptops has been eased by the introduction of alternative devices. Especially mobile end-user devices such as smartphones and tablet computers have significantly gained popularity and relevance during the past years and have gradually replaced desktop PCs and laptops. Second, ongoing technological advances continuously facilitate new use cases and application scenarios. In this regard, e-government can be mentioned as a representative example. Enabled by the availability of powerful technologies, governments and public administrations successively transfer the provision of services to the digital domain. This especially applies to member states of the European Union (EU), which increasingly employ ICT to improve provided services and to reduce costs [European Commission, 2014b].

The first trend, i.e. the renunciation of classical end-user devices, has mainly been caused by vast enhancements in mobile computing, which have led to the development of both powerful mobile end-user devices and efficient mobile communication networks. According to EMarketer [2014], there was a worldwide 25% growth in smartphone usage in 2014. Behind the Asia-Pacific region, Western Europe represents the area with the second largest number of smartphone users. In total, Western Europe had 196.6 million smartphone users in 2014 [EMarketer, 2014]. Relevant reports underpin the growing relevance of mobile computing all over the world and show that mobile computing is actually a global trend [MobiForge, 2014].

The increasing popularity of mobile computing and the continuing trend to replace classical end-user devices raises several challenges. For instance, mobile end-user devices differ from classical ones in various aspects. This requires existing solutions and services to be adapted accordingly, in order to meet the special requirements of mobile end-user devices. For example, web-based services must be redesigned such that they are compliant with mobile web browsers and remain usable on mobile devices with reduced input and output capabilities. Further challenges are raised by the heterogeneity of current mobile platforms. At present, three mobile platforms hold significant market shares [IDC, 2014]. These are Google Android[3], Apple iOS[4], and Microsoft Windows Phone 8[5]. All of these platforms must be regarded as closed ecosystems and are hardly compatible to each other. This means that mobile solutions must be developed, deployed, and maintained for all platforms separately. In particular, this applies to the provision of so-called mobile apps, i.e. applications that are designed for mobile platforms and that are distributed over a centrally maintained platform-specific app repository. Even though all major smartphone platforms implement the concept of mobile apps, they are specific to the mobile operating system and hence cannot be reused on different platforms.

Despite these challenges, more and more services are nowadays offered for mobile end-user devices. There are several reasons for that. First and foremost, smartphones and related end-user devices enable so-called always-on societies. This means that users are online 24/7 and have access to information and services everywhere and at any time. This, in turn, enables service providers to offer services that are accessible and usable irrespective of the user's current context. Second, modern mobile end-user devices integrate various technologies that are typically not available on classical end-user devices. This includes for instance positioning systems, short-range communication technologies, or various sensors. These technologies enable new and innovative applications such as Location-Based Services (LBSs), which are typically infeasible on classical end-user devices. Due to the availability of innovative technologies and the capability to satisfy the demands of always-on societies, mobile computing has emerged to one of

[3] http://www.android.com/
[4] https://www.apple.com/ios/
[5] http://www.windowsphone.com

the most relevant ICT trends during the past years.

Besides the trend towards mobile computing, technological enhancements have facilitated various new use cases and application scenarios for information and communication technologies. For instance, the accomplishment of banking transactions and the buying of goods have been gradually moved to the digital domain during the past years, and have leveraged the fields of Electronic Banking (e-banking) and Electronic Commerce (e-commerce). Inspired by the success of these fields of application, also governments and public-sector administrations have started to provide services and procedures electronically. The use of ICT in the public sector has soon become known under the term e-government. Nowadays, e-government covers a broad spectrum of electronic public-sector services ranging from the simple provision of static information on governmental websites to the provision of fully transactional electronic services that enable citizens to complete entire procedures over the Internet. Hence, e-government can be regarded as a prime example for a new use case and application scenario that has been enabled by enhancements of ICTs.

Especially in the EU, the immense potential of e-government has been recognized early [European Commission, 2014c]. To achieve the goals defined by the European Commission's Digital Agenda [European Commission, 2014a], various member states of the EU have developed e-government solutions according to national legal and organizational requirements. Currently, Europe faces a heterogeneous landscape of mainly national e-government concepts and solutions. Although most national solutions rely on the same concepts such as electronic identities or electronic signatures, solutions from different member states are usually not interoperable. Achieving interoperability between national solutions has been the goal of several pan-European pilot projects[6,7,8,9,10], which have aimed to provide e-government services across national borders. National initiatives to further develop member state specific e-government solutions and pan-European activities to achieve interoperability between national solutions emphasize the relevance of e-government in Europe.

Similar to other fields of application, the global trend towards mobile computing has recently also affected developments in the e-government domain. The integration of mobile communication technologies into e-government services and applications has become commonly known under the term Mobile Government (m-government). Mobile technologies offer various opportunities to improve existing e-government services and enable new and innovative solutions. For instance, m-government apps can provide location-based e-government services by incorporating the user's current context. At the same time, provision of e-government services through mobile end-user devices also bears several risks and raises various challenges. This especially applies to transactional services, which integrate security-critical concepts such as electronic identities and electronic signatures. Solutions implementing these concepts are available and have been in productive operation in many countries for years. However, they have usually been designed for classical end-user devices. This raises issues regarding feasibility, security, and usability, when these solutions are ported to and applied on mobile end-user devices. Due to these issues, transactional m-government services that rely on electronic identities and electronic signatures are still rare. To address this issue, this thesis proposes, discusses, and evaluates solutions to leverage the transition from pure informational m-government to transactional m-government. In particular, this thesis focuses on the development of electronic-signature solutions that are feasible on mobile end-user devices and can be applied on mobile devices in a secure and usable way.

[6] https://www.eid-stork.eu/
[7] https://www.eid-stork2.eu/
[8] http://www.peppol.eu/
[9] http://www.e-codex.eu
[10] http://www.eu-spocs.eu/

1.2 Methodology

Responsiveness to changing circumstances is key for the sustainable success of ICT solutions. This especially holds true for solutions incorporating mobile technologies, as the mobile sector is subject to frequent technological innovations, short release cycles, and dynamic markets. Mobile solutions hence need to provide a sufficient degree of flexibility, in order to be able to respond to fast changing circumstances and varying requirements.

To maintain a sufficient level of responsiveness and flexibility, solutions proposed in this thesis are provided on different levels of abstraction. In terms of sustainability, a high degree of abstraction is advantageous, as it enables the modeling of solutions independent of concrete technologies. This way, proposed solutions remain valid, even if available technologies change. When it comes to concrete realizations, a high degree of abstraction is often disadvantageous, as the mapping of abstract concepts to currently available technologies can be difficult. It is hence reasonable to propose and provide technical solutions on different levels of abstraction, in order to achieve sustainability and to assure feasibility of concrete realizations at the same time.

The approach to provide solutions on different levels of abstraction is also followed in this thesis. This becomes apparent from the thesis's methodology, which is illustrated in Figure 1.1. Figure 1.1 lists activities, which have been completed during the work on this thesis, and achievements, which have been yielded from these activities. Together, activities and achievements listed illustrate the way followed from the definition of the thesis's basic goal, over the proposal of an abstract solution, to the realization and evaluation of a concrete implementation. Three achievements represent the three basic milestones of this thesis and are marked in Figure 1.1 accordingly.

As illustrated in Figure 1.1, this thesis's methodology comprises the following activities. First, the e-government state of the art is analyzed and relevant current trends are identified. This yields the general goal of this thesis, i.e. the improvement of m-government. Taking into account this general goal, the methodology followed defines the survey of current m-government solutions as subsequent activity. From the results of this survey, the concrete goal of the thesis, i.e. the leveraging of transactional m-government, is derived. From this concrete goal, the basic problem tackled by this thesis is derived: the provision of a signature solution for mobile end-user devices. According to the methodology followed, the problem derivation is accompanied by an assessment of current mobile technologies. The definition of the concrete problem addressed by this thesis represents the first milestone of this thesis.

To tackle the problem defined, legal requirements that are relevant for mobile signature solutions are identified first. From these requirements, a first abstract model is derived. From this model, implementation options are systematically derived. This yields a set of possible implementation variants for mobile signature solutions. The preferred implementation variant is subsequently determined by conducting assessments regarding feasibility, security, and usability. From the obtained assessment results, server-based approaches are identified as the preferred implementation variant for mobile signature solutions and an abstract model of a server-based signature solution for mobile end-user devices is derived. This is achieved by integrating a user-authentication method that satisfies the needs of signature solutions on mobile end-user devices. The derived model of an abstract signature solution for mobile end-user devices represents the second milestone of this thesis.

To assess and evaluate its feasibility and applicability, the proposed abstract model is further developed towards a concrete solution. For this purpose, a functional model is derived first. This is achieved by defining and applying a set of design principles, which further detail the proposed abstract model. From the functional model, a concrete solution named Smartphone Signature is subsequently derived by selecting available mobile technologies. Joining on an existing solution, the Smartphone Signature is finally implemented. This proves that the proposed abstract signature solution for mobile end-user devices can be mapped to concrete technologies and can be further developed towards a concrete implementation. The resulting implementation thus represents the third and final milestone of this thesis.

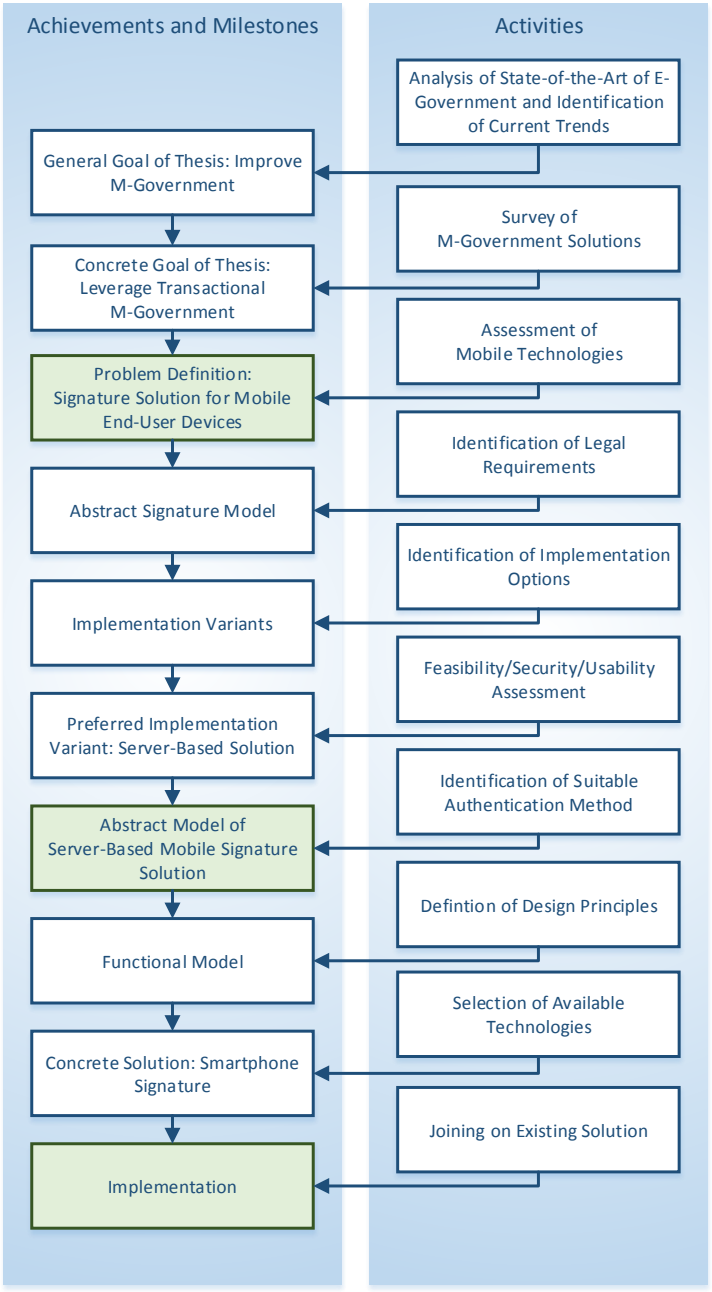

Figure 1.1: The followed methodology yields mobile signature solutions on different levels of abstraction.

The list of activities and achievements emphasize the basic aim of this thesis, i.e. the provision of solutions on different levels of abstraction. By providing both an abstract model and a concrete implementation, the signature solution for mobile end-user devices proposed in this thesis is responsive to technological changes and is proven to be feasible and compliant to state-of-the-art technologies.

1.3 Contribution

Contributions of this thesis become apparent from the list of achievements illustrated in Figure 1.1. Main achievements have also been introduced and discussed in published conference papers and journal articles. They can be regarded as scientific backbone of this thesis. To provide a comprehensive overview, own papers and articles related to the topics covered by this thesis are listed in the following.

Listed publications have been classified into three categories. Publications of Category A are directly related to the concrete problem tackled by this thesis, i.e. to mobile government and mobile signature solutions. Papers and articles assigned to Category A can hence be regarded as core publications of this thesis. Publications assigned to Category B and Category C are related to two topics that are relevant for mobile government, i.e. mobile security and e-government. Even though not all publications from these categories are directly related to the problem tackled by this thesis, they still yield valuable results and findings that enable the achievement of this thesis's goals.

Following common scientific practice, work on this thesis has benefited from discussions and cooperations with colleagues and partners, who have partially contributed to the achievement of this thesis's goals. This is also reflected by the fact that several publications listed below contain one or more coauthors. In most cases, the provided order of authors corresponds to the share in contribution, whereas the main contributor is the first author of the publication. Accordingly, the author of this thesis has been the main contributor of all core publications assigned to Category A and of numerous related publications assigned to Category B and Category C.

1.3.1 Category A: Mobile Government and Mobile Signature Solutions

Thomas Zefferer and Peter Teufl. *Opportunities and Forthcoming Challenges of Smartphone-based m-Government Services. European Journal of ePractice, (Megatrends in eGovernment)*, volume 13, 2011.

[Zefferer and Teufl, 2011]

Thomas Zefferer, Peter Teufl, and Herbert Leitold. *Mobile qualifizierte Signaturen in Europa. Datenschutz und Datensicherheit – DuD*, volume 35, pages 768–773. Springer. 2011.

[Zefferer et al., 2011c]

Thomas Zefferer, Arne Tauber, Bernd Zwattendorfer, and Klaus Stranacher. *Qualified PDF Signatures on Mobile Phones. In Electronic Government and Electronic Participation - Joint Proceedings of Ongoing Research and Projects of IFIP EGOV and IFIP ePart 2012*, pages 115–123. Trauner. 2012.

[Zefferer et al., 2012b]

Thomas Zefferer, Sandra Kreuzhuber, and Peter Teufl. *Assessing the Suitability of Current Smartphone Platforms for Mobile Government. In Technology-Enabled Innovation for Democracy, Government and Governance*, pages 125–139. Springer. 2013.

[Zefferer et al., 2013b]

Thomas Zefferer, Fabian Golser, and Thomas Lenz. *Towards Mobile Government: Verification of Electronic Signatures on Smartphones. In Technology-Enabled Innovation for Democracy, Government and Governance*, pages 140–151. Springer. 2013.

[Zefferer et al., 2013a]

Thomas Zefferer and Bernd Zwattendorfer. *An Implementation-Independent Evaluation Model for Server-Based Signature Solutions. In 10th International Conference on Web Information Systems and Technologies*, pages 302–309. SciTePress. 2014.

[Zefferer and Zwattendorfer, 2014]

Thomas Zefferer. *A Server-Based Signature Solution for Mobile Devices. In The 12th International Conference on Advances in Mobile Computing and Multimedia*, pages 175–184. ACM. 2014.

[Zefferer, 2014]

Thomas Zefferer. *Towards a New Generation of Mobile Government. In 13th International Conference on e-Society (ES 2015)*, pages 191–198. IADIS. 2015.

[Zefferer, 2015a]

Thomas Zefferer. *Towards Transactional Electronic Services on Mobile End-User Devices – A Sustainable Architecture for Mobile Signature Solutions. In 11th International Conference on Web Information Systems and Technologies*, pages 586–597. SciTePress. 2015.

[Zefferer, 2015b]

Thomas Zefferer and Peter Teufl. *Leveraging the Adoption of Mobile eID and e-Signature Solutions in Europe. In 4th International Conference on Electronic Government and the Information Systems Perspective*. Springer. In press.

[Zefferer and Teufl, in press]

1.3.2 Category B: Mobile Security

Thomas Zefferer, Arne Tauber, and Bernd Zwattendorfer. *Improving the Security of SMS-based Services using Electronic Signatures - Towards SMS-based Processing of Transactional m-Government Services. In 8th International Conference on Web Information Systems and Technologies*, pages 743–752. SciTePress. 2012.

[Zefferer et al., 2012a]

Peter Teufl, Thomas Zefferer, Sandra Kreuzhuber, and Christian Lesjak. *Trusted Location Based Services. In International Conference for Internet Technology and Secured Transactions 2012* , pages 185–192. IEEE. 2012.

[Teufl et al., 2012]

Thomas Zefferer, Arne Tauber, and Bernd Zwattendorfer. *Harnessing Electronic Signatures to Improve the Security of SMS-based Services. In Web Information Systems and Technologies – Lecture Notes in Business Information Processing*, pages 331–346. Springer. 2013.

[Zefferer et al., 2013c]

Thomas Zefferer and Peter Teufl. *Policy-based Security Assessment of Mobile End-User Devices. In Proceedings of the 10th International Conference on Security and Cryptography*, pages 347–354. SciTePress. 2013.

[Zefferer and Teufl, 2013]

Thomas Zefferer. *A Survey and Analysis of NFC-Based Payment Solutions for Smartphones. In International Conference e-Society 2013*, pages 275–282. IADIS. 2013.

[Zefferer, 2013]

Thomas Zefferer, Peter Teufl, David Derler, Klaus Potzmader, Alexander Oprisnik, Hubert Gasparitz, and Andrea Höller. *Power Consumption-Based Application Classification and Malware Detection on Android Using Machine-Learning Techniques. In FUTURE COMPUTING 2013, The Fifth International Conference on Future Computational Technologies and Applications*, pages 26–31. IARIA. 2013

[Zefferer et al., 2013d]

Peter Teufl, Thomas Zefferer, and Christof Stromberger. *Mobile Device Encryption Systems. In 28th IFIP TC-11 SEC 2013 International Information Security and Privacy Conference*, pages 203–216. Springer. 2013.

[Teufl et al., 2013a]

Peter Teufl, Thomas Zefferer, Christof Stromberger, and Christoph Hechenblaikner. *iOS Encryption Systems – Deploying iOS Devices in Security-Critical Environments. In 10th International Conference on Security and Cryptography*, pages 170–182. SciTePress. 2013.

[Teufl et al., 2013b]

Thomas Zefferer, Peter Teufl, David Derler, Klaus Potzmader, Alexander Oprisnik, Hubert Gasparitz, and Andrea Höller. *Towards Secure Mobile Computing: Employing Power-Consumption Information to Detect Malware on Mobile Devices. In International Journal On Advances in Software*, 6, pages 150–160. IARIA. 2014.

[Zefferer et al., 2014b]

Peter Teufl, Thomas Zefferer, Christoph Wörgötter, Alexander Oprisnik, and Daniel Hein. *Android – On-Device Detection of SMS Catchers and Sniffers. In International Conference on Privacy & Security in Mobile Systems*, pages 1–8. IEEE. 2014.

[Teufl et al., 2014b]

Peter Teufl, Daniel Hein, Alexander Marsalek, Thomas Zefferer, and Alexander Oprisnik. *Android Encryption Systems. In International Conference on Privacy & Security in Mobile Systems*, pages 1–8. IEEE. 2014.

[Teufl et al., 2014a]

Andreas Reiter and Thomas Zefferer. *Paving the Way for Security in Cloud-Based Mobile Augmentation Systems. In 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (IEEE Mobile Cloud 2015)*, pages 89–98. ACM. 2015.

[Reiter and Zefferer, 2015]

Florian Reimair, Peter Teufl, and Thomas Zefferer. *WebCrySIL – Web Cryptographic Service Interoperability Layer. In In 11th International Conference on Web Information Systems and Technologies*, pages 35–44. SciTePress. 2015.

[Reimair et al., 2015]

1.3.3 Category C: Electronic Government

Thomas Zefferer and Thomas Knall. *An Electronic-Signature Based Circular Resolution Database System. In Proceedings of the 25th Annual ACM Symposium on Applied Computing 2010*, pages 1840–1845. ACM. 2010.

[Zefferer and Knall, 2010]

Arne Tauber, Bernd Zwattendorfer, Thomas Zefferer, Yasmin Mazhari, and Eleftherios Chamakiotis. *Towards Interoperability: An Architecture for Pan-European eID-based Authentication Services. In Proceedings of the International Conference on Electronic Government and the Information Systems Perspective (EGOVIS)*, volume 6267/2010, pages 120–133. Springer. 2010.

[Tauber et al., 2010]

Thomas Zefferer, Arne Tauber, Bernd Zwattendorfer, and Thomas Knall. *Secure and Reliable Online-Verification of Electronic Signatures in the Digital Age. In Proceedings of the International Conference WWW/INTERNET 2011*, pages 269–276. IADIS. 2011.

[Zefferer et al., 2011b]

Thomas Zefferer, Vesna Krnjic, and Bernd Zwattendorfer. *Ein virtuelles Testframework für EGovernment Komponenten. In D-A-CH Security 2011*, pages 492–503. syssec. 2011.

[Zefferer et al., 2011a]

Clemens Orthacker and Thomas Zefferer. *Accessibility Challenges in e-Government: an Austrian Experience. In Proceedings of the Forth International Conference on Internet Technologies and Applications (ITA 11)*, pages 221–228. Glyndwr University. 2011.

[Orthacker and Zefferer, 2011]

Bernd Zwattendorfer, Thomas Zefferer, and Arne Tauber. E-ID Meets E-Health on a Pan-European Level. In Proceedings of the IADIS International Conference e-Health 2011, pages 97–104. IADIS. 2011.

[Zwattendorfer et al., 2011b]

Bernd Zwattendorfer, Thomas Zefferer, and Arne Tauber. *A Privacy-Preserving eID Based Single Sign-On Solution. In Proceedings of 5th International Conference on Network and System Security*, pages 295–299. Springer. 2011.

[Zwattendorfer et al., 2011a]

Thomas Knall, Arne Tauber, Thomas Zefferer, Bernd Zwattendorfer, Arnaldur Axfjord, and Haraldur Bjarnason. *Secure and Privacy-Preserving Cross-Border Authentication: The STORK Pilot SaferChat. In Proceedings of the Conference on Electronic Government and the Information Systems Perspective*, pages 94–106. Springer. 2011.

[Knall et al., 2011]

Arne Tauber, Bernd Zwattendorfer, and Thomas Zefferer. *STORK: Pilot 4 - Towards Crossborder Electronic Delivery. In Joint Proceedings of Ongoing Research and Projects of IFIP EGOV and ePart*, pages 295–301. Trauner. 2011.

[Tauber et al., 2011c]

Karl-Christian Posch, Reinhard Posch, Arne Tauber, Thomas Zefferer, and Bernd Zwattendorfer. *Secure and Privacy-Preserving eGovernment - Best Practice Austria. In Rainbow of Computer Science*, pages 259–269. Springer. 2011.

[Posch et al., 2011]

Arne Tauber, Bernd Zwattendorfer, and Thomas Zefferer. *A Shared Certified Mail System for the Austrian Public and Private Sectors. In Electronic Government and the Information Systems Perspective*, pages 356–369. Springer. 2011.

[Tauber et al., 2011b]

Arne Tauber, Thomas Zefferer, and Bernd Zwattendorfer. *Elektronisches Einschreiben im D-A-CH Raum. In D-A-CH Security 2011*, pages 510–521. syssec. 2011.

[Tauber et al., 2011a]

Arne Tauber, Bernd Zwattendorfer, Thomas Zefferer, and Klaus Stranacher. *Grenzüberschreitendes E-Government in Europa. In eGovernment Review*, volume 8, pages 8–9. 2011.

[Tauber et al., 2011d]

Thomas Zefferer and Vesna Krnjic. *Usability-Evaluierung der österreichischen Handy-Signatur. In D-A-CH Security 2012*, pages 365–376. syssec. 2012.

[Zefferer and Krnjic, 2012c]

Thomas Zefferer and Vesna Krnjic. *Towards User-friendly E-Government Solutions: Usability Evaluation of Austrian Smart-Card Integration Techniques. In Advancing Democracy, Government and Governance*, pages 88–102. Springer. 2012.

[Zefferer and Krnjic, 2012a]

Thomas Zefferer and Vesna Krnjic. *Usability Evaluation of Electronic Signature Based EGovernment Solutions. In Proceedings of the International Conference WWW/INTERNET 2012*, pages 227–234. IADIS. 2012.

[Zefferer and Krnjic, 2012b]

Reinhard Posch, Clemens Orthacker, Klaus Stranacher, Arne Tauber, Thomas Zefferer, and Bernd Zwattendorfer. *Open Source Bausteine als Kooperationsgrundlage. In E-Government - Zwischen Partizipation und Kooperation*, pages 185–209. Springer. 2012.

[Posch et al., 2012]

Arne Tauber, Thomas Zefferer, and Bernd Zwattendorfer. *Approaching the Challenge of eID Interoperability: An Austrian Perspective. In European Journal of ePractice*, volume 14, pages 22–39. 2012.

[Tauber et al., 2012]

Bernd Zwattendorfer, Thomas Zefferer, and Arne Tauber. *The Prevalence of SAML within the European Union. In 8th International Conference on Web Information Systems and Technologies*, pages 571–576. SciTePress. 2012.

[Zwattendorfer et al., 2012]

Klaus Stranacher, Vesna Krnjic, and Thomas Zefferer. *Vertrauenswürdiges Open Government Data. In 1.OGD D-A-CH-LI Konferenz*, pages 27–39. ADV. 2012.

[Stranacher et al., 2012]

Thomas Lenz, Klaus Stranacher, and Thomas Zefferer. *Towards a Modular Architecture for Adaptable Signature-verification Tools. In 9th International Conference on Web Information Systems and Technologies*, pages 325–334. SciTePress. 2013.

[Lenz et al., 2013b]

Thomas Lenz, Klaus Stranacher, and Thomas Zefferer. *Enhancing the Modularity and Applicability of Web-Based Signature-Verification Tools. In WEBIST 2013 Selected and Revised Papers*, pages 173–188. Springer. 2013.

[Lenz et al., 2013a]

Vesna Krnjic, Philip Weber, Thomas Zefferer, and Bernd Zwattendorfer. *Effizientes Testen von E-Government Komponenten in der Cloud. In D-A-CH Security 2013*, pages 225–236. syssec. 2013.

[Krnjic et al., 2013]

Klaus Stranacher, Vesna Krnjic, Bernd Zwattendorfer, and Thomas Zefferer. *Assessment of Redactable Signature Schemes for Trusted and Reliable Public Sector Data. In Proceedings of the 13th European Conference on e-Government*, pages 508–516. ACPI. 2013.

[Stranacher et al., 2013c]

Klaus Stranacher, Vesna Krnjic, and Thomas Zefferer. *Trust and Reliability for Public Sector Data. In Proceedings of International Conference on e-Business and e-Government*, volume 73, pages 124–132. ACPI. 2013.

[Stranacher et al., 2013b]

Klaus Stranacher, Vesna Krnjic, Bernd Zwattendorfer, and Thomas Zefferer. *Evaluation and Assessment of Editable Signatures for Trusted and Reliable Public Sector Data. In Electronic Journal of e-Government*, volume 11, pages 360–372. 2013.

[Stranacher et al., 2013d]

Klaus Stranacher, Vesna Krnjic, and Thomas Zefferer. *Authentische und integritätsgesicherte Verwaltungsdaten. In eGovernment Review*, volume 11, pages 30–31. 2013.

[Stranacher et al., 2013a]

Klaus Stranacher, Arne Tauber, Thomas Zefferer, and Bernd Zwattendorfer. *The Austrian Identity Ecosystem - An e-Government Experience. In Architectures and Protocols for Secure Information Technology*, pages 288-309. IGI Global. 2013.

[Stranacher et al., 2013e]

Thomas Zefferer, Vesna Krnjic, Klaus Stranacher, and Bernd Zwattendorfer. *Measuring Usability to Improve the Efficiency of Electronic Signature-based E-Government Solutions. In Measuring E-government Efficiency. The Opinions of Public Administrators and other Stakeholders*, pages 45–74. Springer. 2014.

[Zefferer et al., 2014a]

Bernd Zwattendorfer, Thomas Zefferer, and Klaus Stranacher. *An Overview of Cloud Identity Management-Models. In Proceedings of the 10th International Conference on Web Information Systems and Technologies*, pages 82–92. SciTePress. 2014.

[Zwattendorfer et al., 2014]

Christof Rath, Simon Roth, Manuel Schallar, and Thomas Zefferer. *A Secure and Flexible Server-Based Mobile eID and e-Signature Solution. In The Eighth International Conference on Digital Society*, pages 7–12. IARIA. 2014.

[Rath et al., 2014a]

Bernd Zwattendorfer, Thomas Zefferer, and Klaus Stranacher. *A Comparative Survey of Cloud Identity Management-Models. In WEBIST 2014 - Selected and Revised Papers*. Springer. In press.

[Zwattendorfer et al., in press]

Christof Rath, Simon Roth, Manuel Schallar, and Thomas Zefferer. *Design and Application of a Secure and Flexible Server-Based Mobile eID and e-Signature Solution. International Journal on Advances in Security*, pages 50–61. IARIA. 2014.

[Rath et al., 2014b]

Christof Rath, Simon Roth, Harald Bratko, and Thomas Zefferer. *Encryption-based Second Authentication Factor Solutions for Qualified Server-side Signature Creation. In 4th International Conference on Electronic Government and the Information Systems Perspective*. Springer. In press.

[Rath et al., in press]

1.4 Outline

The structure of this thesis approximately corresponds to the defined methodology illustrated in Figure 1.1 on page 7. Achievements of this thesis according to the defined methodology are introduced and discussed in a total of eight chapters. The eight chapters are grouped into three parts, which are based on each other. Each part is dedicated to one of the three milestones of this thesis.

Accordingly, Part I focuses on the identification of the concrete problem tackled. For this purpose, Part I is subdivided into three chapters. After a brief general introduction to this thesis provided in Chapter 1, the evolution of mobile government is discussed in Chapter 2. This includes the introduction of relevant e-government concepts, the discussion of the transition from e-government to m-government, and a survey of the current state of the art of m-government. Subsequently, Chapter 3 focuses on opportunities and limitations of current mobile technologies in terms of security, and investigates these technologies' capabilities to satisfy requirements of m-government solutions. Based on the findings obtained from Chapter 2 and Chapter 3, the concrete problem tackled by this thesis, i.e. development of a signature solution for mobile end-user devices, is finally defined.

Part II of this thesis systematically analyzes the defined problem in order to develop a suitable solution and to reach the thesis's second milestone. The problem analysis and solution development is covered by two chapters. In Chapter 4, the most suitable general architecture for mobile signature solutions is developed. The developed architecture is subsequently refined in Chapter 5 by integrating a suitable user-authentication method. This finally yields a complete abstract model of a server-based mobile signature solution, which represents the second milestone of this thesis.

Based on the developed abstract model, a concrete implementation is introduced and discussed in Part III. This implementation evaluates the proposed model's feasibility and applicability in practice. In total, Part III comprises three chapters. In Chapter 6, the proposed model is further developed towards a concrete solution called Smartphone Signature. Subsequently, an implementation of this solution that relies on state-of-the-art technology is presented in Chapter 7. This implementation integrates three different mobile cutting-edge technologies to demonstrate the proposed solution's flexibility. Final conclusions are subsequently drawn in Chapter 8.

Chapter 2

The Evolution of Mobile Government

" The single most important top-level trend is the shift to mobile."

[Max Rafael Levchin, American Computer Scientist.]

With his concise statement, Max Rafael Levchin has summarized a phenomenon that has been omnipresent during the past years: the trend towards mobile computing. Emerging trends, available technologies, and the current state of the art have always influenced workaday life. This also applies to the interaction between governments, public administrations, and citizens. Before PCs became mass-produced goods, paper-based administrative procedures had been state of the art. Citizen had to show up personally in public offices to file paper-based applications. These applications were then processed by officials in charge according to defined back-office processes. Results of these processes such as official notifications or administrative rulings were finally delivered to citizens by paper-based mail. In the 1980's, PCs finally became mass-produced goods. With their increasing spread, these devices were also increasingly used by public administrations to speed up back-office processes, while the filing of applications as well as the delivery of official notifications and administrative rulings was still based on paper. This has finally changed with the emergence of the Internet and the feasibility of web-based services. Since the early noughties, transactional e-government services have enabled citizens to interact with public administrations and to carry out complete electronic procedures over the Internet.

During the past years, the predominating computing paradigm has changed, as mobile communication technologies have significantly gained importance. Powered by the success story of smartphones and tablet computers, and enabled by the increasing availability of mobile broadband networks, mobile end-user devices are nowadays gradually replacing classical devices such as desktop PCs and laptops. Governments and public administrations are already reacting to this new computing paradigm and are currently preparing their services for access by and use with mobile end-user devices. The use of mobile technologies in e-government solutions has become commonly known under the term mobile government or m-government. First experiences with m-government show that the incorporation of mobile cutting-edge technologies into e-government services opens up new use cases and application possibilities. At the same time, the integration of these technologies also raises new threats and challenges that need to be addressed.

In this chapter, the evolution from paper-based administrative procedures over e-government towards m-government is discussed in more detail. This way, general requirements of administrative procedures are identified, basic concepts of e-government are introduced, and an overview of the current state of the art of m-government is provided. From this overview, limitations and drawbacks of current m-government solutions are identified and the basic goal of this thesis is motivated.

2.1 E-Government Concepts

For many years, interaction with public administrations has been cumbersome for citizens, who have in many cases been treated rather as supplicants than as customers. In these times, citizens had to queue at counters, had to cope with limited office hours, and often had to deal with complex paper-based procedures. During the past few decades, public administrations have experienced a growing pressure to increase efficiency in order to save costs. At the same time, public administrations have been positively influenced by the private sector so that customer satisfaction has become an increasingly important aspect also for public services.

In order to improve customer experience and to increase efficiency, public administrations have started to make use of ICTs. Nowadays, ICTs are used by the public sector to speed-up internal back-office processes and to improve the interaction with both citizens and businesses from the private sector. Irrespective of the actual use case, the use of ICTs in the public sector has become commonly known under the term e-government. The World Bank defines e-government as *'the use by government agencies of information technologies (such as Wide Area Networks, the Internet, and mobile computing) that have the ability to transform relations with citizens, businesses, and other arms of government.'* [World Bank, 2014]. Nowadays, e-government concepts are adopted by public bodies and administrations of nearly all developed countries. This especially applies to the EU, where the Digital Agenda for Europe [European Commission, 2014a] leverages the use of ICTs in the public sector.

In many member states of the EU, e-government initiatives have already been started in the 1990's and have yielded e-government services and solutions relying on technologies of this era. Powered by the emergence of new and innovative technologies, concrete implementations of e-government services have changed during the past decades. Nevertheless, the fundamentals and basic underlying concepts have basically remained the same. In this section, relevant concepts of e-government are briefly sketched and their application in practice is exemplified by means of concrete solutions.

2.1.1 E-Government Fundamentals

Definitions of the term e-government such as the one provided by the World Bank [2014] are typically rather generic. Thus, the term e-government covers a broad spectrum of services and applications ranging from simple informational websites provided by public-sector agencies to full electronic procedures. It is hence unsurprising that several classification schemes have been introduced to group and divide e-government services and applications.

Depending on the used classification criterion, e-government services can be classified according to different aspects. A commonly used classification scheme groups e-government services according to the set of involved participants. This approach has for instance been followed by Fang [2002]. Potential participants of e-government services and applications are governmental agencies, citizens, private-sector businesses, non-profit organizations, and employees of governmental agencies. Accordingly, Fang [2002] identifies the following types of e-government services: Government-to-Citizen (G2C), Citizen-to-Government (C2G), Government-to-Business (G2B), Business-to-Government (B2G), Government-to-Government (G2G), Government-to-Nonprofit (G2N), Nonprofit-to-Government (N2G), and finally Government-to-Employee (G2E).

As an alternative to the set of involved participants, also the nature of communication can be used to classify e-government services and applications. This is a common approach followed in several scientific publications. A prime example is a paper by Nariman and Yamamoto [2008], in which the authors rely on this classification scheme and also provide definitions of the different categories of e-government services. Concretely, they distinguish between the following types of e-government services:

- **Informational e-government:** This kind of e-government services refers to *'a passive presentation of general information'* [Nariman and Yamamoto, 2008]. Common implementations of

informational e-government services are web sites provided by public agencies, which contain useful information such as office hours, relevant phone numbers, or mailing addresses.

- **Responsive e-government:** The general goal of responsive e-government services is to *'make commonly requested information and forms available around the clock'* [Nariman and Yamamoto, 2008], in order to avoid necessary personal interactions with the agency, e.g. during office hours. Responsive e-government services represent a step towards full electronic procedures but still comprise several manual processing steps.
- **Transactional e-government:** Transactional services are the most complex but also the most powerful kind of e-government solutions. They *'enable clients/users to complete entire tasks electronically at any time of the day or night'* [Nariman and Yamamoto, 2008]. These services typically require a reliable authentication of the user at the e-government service and the electronic provision of written consent by the user.

From a technical and implementation perspective, transactional services are most challenging, as they implement full electronic procedures. Considering the participants-based classification scheme, transactional services are especially relevant for interactions between governments and citizens, i.e. for G2C and C2G transactions. In general, transactional e-government services resemble classical administrative procedures and therefore need to satisfy the same requirements. We have discussed relevant aspects of classical administrative procedures [Posch et al., 2011] and have identified their three basic stages, which are illustrated in Figure 2.1.

Figure 2.1: Classical administrative procedures typically comprise the three stages Application, Back-Office Processing, and Delivery.

As shown in Figure 2.1, classical administrative procedures typically consist of the three stages Application, Back-Office Processing, and Delivery. In the first stage, the citizen shows up in a public office to file an application. Therefore, the citizen hands over a filled paper-based form to the official in charge. Depending on the concrete procedure, the citizen might be required to authenticate at the official in charge. This is usually achieved by presenting an official identity card or a passport. In most cases, the filled paper-based form needs to be signed by the citizen. The official in charge acknowledges receipt of the filled paper-based form and hence of the filed application. Filed applications are forwarded internally to the public administration's back office. There, they are processed according to the defined process flow of the respective procedure. In most cases, this involves a manual processing by responsible officials. Back-office processing defines the second stage of classical administrative procedures. If the respective procedure demands it, relevant documents are finally delivered to the citizen, who has filed the application. This might include official notifications, administrative rulings, or newly issued documents. Delivery represents the third and last stage of classical administrative procedures.

Transactional e-government services resemble classical administrative procedures. Hence, mandatory concepts for transactional e-government services can be derived from the three stages of classical administrative procedures. This is elaborated in more detail in the following.

2.1.2 Basic Concepts of Transactional E-Government Services

From the three stages of classical administrative procedures, a set of relevant concepts can be derived. Transactional e-government services need to implement appropriate means in order to map these concepts to the digital world. Concretely, the following concepts can be identified as crucial for classical administrative procedures and hence also for transactional e-government services:

- **Identification:** For most procedures, citizens need to identify themselves before being able to file an application. In classical procedures, this is usually achieved by showing a valid identity card or passport. Transactional e-government services need to make use of respective electronic identities, in order to remotely identify citizens in online processes.
- **Authentication:** During authentication, citizens confirm an indicated identity. For classical administrative procedures, authentication is implicitly covered during identification by showing a valid identity document. In the digital world, the concepts of identification and authentication often need to be considered separately. This is best explained by means of the concrete example of user name and password based access-control systems. Provision of the user name covers identification: the user claims a certain electronic identity. Authentication is on the contrary covered by proving knowledge of a secret password, which is assigned to the electronic identity.
- **Filing of applications:** In the analog world, filing an application is typically accomplished by filling a paper-based form. This concept can be easily mapped to the digital world and implemented by transactional e-government services e.g. using approved web-based technologies.
- **Provision of written consent:** In classical administrative procedures, citizens usually need to sign filled forms in order to file an application. This way, citizens provide written consent on the contents of the filled form. Provision of written consent is hence a key concept that also needs to be mapped to the digital world by transactional e-government services. There, electronic signatures are usually the technology of choice to provide data-origin authentication in online processes.
- **Back-office processing:** Required back-office processes depend heavily on the respective procedure that needs to be run through. Irrespective of the concrete procedure, back-office processes can benefit from the use of ICTs. This applies to classical procedures as well as to procedures implemented by transactional e-government services. The latter additionally leverage the use of ICTs in back-office processes, as they avoid media breaks caused by filed applications in paper-based form.
- **Delivery:** Delivery of relevant documents to the citizen is optional and not required by every procedure. If required, classical mail services are typically used in the analog world for this purpose. In the digital world, e-mail is the de-facto standard for the electronic delivery of messages and documents. As e-mail does not assure any service quality, this technology is hardly used by e-government services for delivery purposes. Instead, special e-delivery solutions are employed for this purpose. We have discussed challenges, approaches, and concrete solutions of e-delivery systems in Tauber et al. [2011b] and Tauber et al. [2011c].

From the basic concepts of classical administrative procedures, two relevant concepts for transactional e-government services can be extracted. These are the concepts of electronic identities and electronic signatures. Electronic identities are crucial for all transactional e-government services, as they enable a reliable identification of remote users. Electronic signatures are required for multiple purposes. First, they represent an ideal mechanism to provide written consent in online processes. Second, they can also be used to authenticate users, as provision of authenticity is one key feature of electronic signatures. Together, the related concepts of electronic identities and electronic signatures enable users to identify and authenticate at transactional e-government services, and to provide written consent on electronic

data such as filed applications. Thus, these concepts are crucial for transactional e-government services, especially regarding the interaction between central services and remote users. Relevant details of the concepts of electronic identities and electronic signatures are discussed in the following.

2.1.2.1 Electronic Identities

The Oxford Dictionaries define the term identity as *'the fact of being who or what a person or thing is'* [Oxford University Press, 2014]. The definition of an electronic identity is slightly different. According to Windley [2005], the term Electronic Identity (eID) or digital identity means *'data that uniquely describes a person or a thing and contains information about the subject's relationships.'* [Windley, 2005]. According to this definition, the eID of a person does hence not only define its identity but does potentially also include a set of related attributes.

From a technical or implementation point of view, all data that uniquely identifies a person in electronic processes can be regarded as eID. This can for instance be a unique number, a unique name, or an e-mail address. Irrespective of the chosen implementation, the eID needs to be unique in the respective eID system. Hence, the same eID must not be assigned to two different subjects. However, each subject can theoretically have multiple unique eIDs. This is probably one of the most important differences between the classical identity concept and the concept of electronic identities.

With the emergence of Internet-based services and applications, eIDs have become an integral part of daily life. Nowadays, eIDs are used whenever users need to be identified and authenticated over the Internet. In most cases, eIDs are implemented by means of simple user names and associated passwords. In the private sector, eIDs are for instance used by e-mail services, social networks, or e-commerce solutions. Also the public sector has identified the need for appropriate eID concepts and solutions in order to reliably identify citizens during e-government processes. Especially in Europe, national eID systems, which enable citizens to identify and authenticate at e-government services, have been deployed by public administrations of various countries.

The increasing use of eIDs in the public sector raises the need for solid frameworks that regulate their legal effects. In the European Union, the Regulation on Electronic Identification and Trust Services (eIDAS) [The European Parliament and the Council of the European Union, 2014] will represent the relevant legal basis for eIDs. The eIDAS Regulation defines the use of eIDs and other trust services such as electronic signatures in the European Union. Special focus is put on the interoperability of different national eID systems on European level.

Even though the eIDAS Regulation constitutes a common legal basis for all EU member states, Europe is currently still facing a heterogeneous ecosystem of different national eID solutions. This is mainly due to historical reasons, as member states have started to deploy national eID system long before a common legal basis has been established. This has led to today's situation, in which each EU member state runs its own eID system, which is typically aligned with national laws. For instance, Italy makes use of unique fiscal numbers to identify citizens during e-government processes. In contrast, Austria follows a more complex approach. For data-protection reasons, the Austrian national eID system relies on sector-specific identifiers. This means that each governmental sector such as health, finance, or agriculture uses different identifiers for one and the same citizen. All sector-specific identifiers are derived from the citizen's eID using cryptographic one-way functions. Details of the Austrian national eID system have been discussed by Leitold et al. [2002]. The Italian and the Austrian example show that deployed eID systems can differ fundamentally between EU member states.

The heterogeneity of current national eID systems in Europe raises several issues especially in the context of cross-border services. Usually, eID systems of different countries are not interoperable. This means that e.g. an Italian citizen cannot identify and authenticate at an Austrian e-government service. This compromises the European Commission's idea of a digital single market [European Commission, 2014a] and prevents the broad use of eID-based cross-border services in Europe. To overcome this

issue, the EU large-scale pilot Secure Identity Across Borders Linked (STORK)[1] has been started in 2008 with the goal to achieve interoperability between national eID systems. We have discussed basic concepts of the interoperability framework developed in this project by means of two pilot applications that demonstrate the framework's capabilities [Knall et al., 2011] [Tauber et al., 2011c]. In general, STORK has shown that interoperability between different national eIDs is feasible. Currently, results of the STORK project are consolidated and further improved in its successor project STORK 2.0[2].

The broad use of national eIDs in EU member states and the various research activities in this area emphasize the relevance of electronic identities. Enabling the reliable remote identification of citizens in online processes, electronic identities represent one of the main pillars of transactional e-government services.

2.1.2.2 Electronic Signatures

Electronic signatures represent the second basic concept frequently used in transactional e-government services. There, electronic signatures mainly serve two purposes. First, they enable citizens to provide written consent on electronic data such as electronically filed applications. Second, they can be used to authenticate citizens.

From a non-technical perspective, electronic signatures can be regarded as digital pendant to hand-written signatures as they provide authenticity, integrity, and non-repudiation of signed content. However, compared to their analog pendant, electronic signatures provide several additional features. For instance, they are unambiguously verifiable by technical means and they enable the reliable detection of subsequent modifications of signed content. These features are usually not available for handwritten signatures. Thus, electronic signatures are a powerful tool and are especially useful in transactional e-government services.

From a technical perspective, electronic signatures typically rely on digital signatures, which are a special kind of cryptographic operation. Digital signatures rely on asymmetric cryptography and hence on a cryptographic key pair consisting of a private and a public key. The private key is required to create a digital signature and has to be kept confidential by the signatory. The created signature can only be verified with the corresponding public key. However, it is infeasible to create a valid signature with the public key or to derive the private key from the corresponding public key. Thus, the public key can be made public and does not need to be kept confidential. There are several cryptographic algorithms available that follow the asymmetric cryptographic approach. Examples are Rivest Shamir Adleman (RSA) [Rivest et al., 1978], or the Elliptic Curve Digital Signature Algorithm (ECDSA) [ANSI, 2005]. They all rely on complex mathematical problems and are currently regarded as secure as long as keys with sufficient length are used. Today, these digital-signature algorithms represent the technical foundation of the concept of electronic signatures.

In practice, the concept of electronic signatures is often closely related to the concept of electronic identities. This is comprehensible, as electronic signatures provide authenticity of signed data. Hence, also the identity of the signatory is of relevance. This raises several issues in practice, as there is no obvious link between the signatory's key pair and his or her identity. In other words, being aware of the correct public key, the receiver of an electronically signed document can cryptographically verify the validity of the electronic signature. However, the plain public key does not contain any information about the signatory's identity. To overcome this issue, Public Key Infrastructures (PKIs) are used. PKIs rely on a so-called Certification Authority (CA). A CA is a trusted third party, i.e. it is trusted by both the signatory and the verifier. The principle task of the CA is to establish a verifiable link between public keys and identities. For this purpose, the CA issues electronic certificates. An issued certificate contains the signatory's public key, identity-related information, and optionally additional attributes. The certificate

[1] https://www.eid-stork.eu/
[2] https://www.eid-stork2.eu/

is electronically signed by the CA. This way, the CA confirms the correctness of the contents of the certificate and hence establishes a verifiable binding between the signatory's public key and identity.

During the past few decades, the concept of electronic signatures and its underlying cryptographic concept of digital signatures have significantly gained importance. Digital signatures have for instance been used for many years by the Secure Sockets Layer (SSL)/Transport Layer Security (TLS) protocol [Network Working Group, 2008] to e.g. secure connections between web browsers and servers, by software developers and providers to assure authenticity and integrity of software modules, or by end users to sign e-mails. Recently, also the concept of electronic signatures has gained popularity. Due to their conceptual similarity to handwritten signatures, electronic signatures are nowadays used in various e-banking and e-government solutions to obtain written consent from remote users. To assure equivalence to handwritten signatures also on legal level, legal frameworks have been enacted to define requirements and legal effects of electronic signatures. In the EU, the EU Signature Directive [The European Parliament and the Council of the European Union, 1999] has been the relevant legal foundation for several years. This directive will soon be replaced by the eIDAS Regulation [The European Parliament and the Council of the European Union, 2014], which differentiates various types of electronic signatures and defines so-called qualified electronic signatures to be legally equivalent to handwritten signatures. Additionally, the eIDAS Regulation defines various technical and non-technical requirements for this type of signatures. Essentially, qualified electronic signatures need to be created by so-called Qualified Signature Creation Device (QSCD) and have to rely on qualified certificates. Requirements for QSCDs and qualified certificates are also specified by the eIDAS Regulation.

Due to their various useful features and the availability of respective legal frameworks, electronic signatures have evolved to a central and relevant concept of European e-government solutions. Together with the related concept of electronic identities, electronic signatures represent the backbone of today's e-government solutions. The concrete application and implementation of these concepts in practice is exemplified in the following section.

2.1.3 Electronic Identities and Electronic Signatures in Practice

In practice, the concepts of electronic identities and electronic signatures are often used together in order to implement full transactional e-government services. For this purpose, EU member states usually issue their citizens personalized hardware tokens, which store personal eID data, cryptographic signing keys, and electronic signing certificates. Furthermore, issued tokens support the secure and reliable creation of legally binding electronic signatures. This way, these tokens can be used in online processes to implement required eID and electronic-signature functionality. In most cases, there is a well-defined link between eID-related information and electronic certificates stored on the issued token. For instance, tokens issued to Italian citizens simply store the citizen's unique eID as additional attribute in the signing certificate. In contrast, Austrian tokens store eID information in a separate data structure, which also contains the public keys of the token's signing certificates. Irrespective of the concrete implementation, eID and electronic-signature functionality are usually closely related.

Due to the close relation between eID and electronic-signature functionality, it is unsurprising that in the EU, the legal frameworks for both concepts are defined by one and the same regulation, i.e. the eIDAS Regulation [The European Parliament and the Council of the European Union, 2014]. From a technical point of view, the eIDAS Regulation defines only few requirements and formulates relevant specifications on a rather abstract level. More concrete specifications are expected to be developed in the course of several implementing acts, which are currently under progress. Keeping legal frameworks on a rather abstract level is a double-edged sword. On the one hand, it enables member states to maintain legacy solutions, which have been rolled out long before the respective legal framework has come into force. On the other hand, the abstract nature of defined requirements has led to a heterogeneous ecosystem of different national eID and electronic-signature solutions throughout Europe.

Even though existing solutions of different member states are rather heterogeneous and usually not

compliant with each other, they typically rely on similar concepts and technologies. Hence, a general abstract model can be derived that can be seen as basis for all existing implementations of transactional e-government services. This model is shown in Figure 2.2. According to this model, citizens access a transactional service provided by a Service Provider with the help of a suitable User Client. The Service Provider relies on an Identity/Signature Provider in order to integrate required eID and electronic-signature functionality. Thus, the Identity/Signature Provider carries out necessary tasks such as authenticating citizens or obtaining written consent from citizen. For this purpose, it accesses the citizen's personal eID/Signature Token, which stores relevant eID data and supports the creation of electronic signatures. The citizen usually has to authorize access to the eID/Signature Token, e.g. by providing a secret password. Note that for the sake of simplicity, the shown model assumes that eID and electronic-signature functionality are provided by one and the same component. This is not necessarily the case in practice.

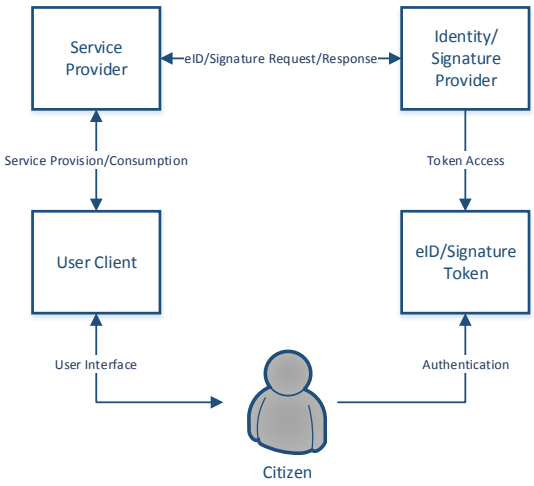

Figure 2.2: Transactional e-government services can be described by means of an abstract model.

Most transactional e-government solutions that have been deployed in Europe during the past years rely on the general architecture shown in Figure 2.2. Depending on the concrete implementation of the eID/Signature Token, currently deployed solutions can be roughly classified into different categories. In practice, implementation alternatives for eID/Signature Tokens are limited. This is mainly due to the strict requirements that must be met by these tokens if they are used for the creation of legally binding signatures. Existing solutions can be classified into two categories. These categories are introduced and discussed in more detail in the following subsections.

2.1.3.1 Smart Card Based Solutions

Smart card based solutions rely on smart-card technology to implement eID/Signature Tokens. In order to provide citizens with eID and electronic-signature functionality, they are supplied with personalized smart cards. These smart cards contain identity-related information such as unique identifiers that can be used to unambiguously identify citizens. Furthermore, they contain one or more cryptographic key pairs and related electronic certificates for the creation of electronic signatures. Access to stored eID data and to the private components of the key pairs are typically protected by a Personal Identification Number (PIN). The smart card must be provided with the correct PIN in order to gain access to these data.

Assuming the use of smart cards, the abstract model shown in Figure 2.2 can be further refined. This yields the model shown in Figure 2.3. Even though this model is specific for smart card based

tokens, the basic processing steps remain the same as for the abstract model. However, reliance on smart cards imposes the need for an additional middleware component. This middleware is necessary to provide the remote Identity/Signature Provider access to locally connected smart cards. In most cases, this middleware resides on the citizen's local system and is implemented by means of locally installed software, browser plug-ins, or Java Applets.

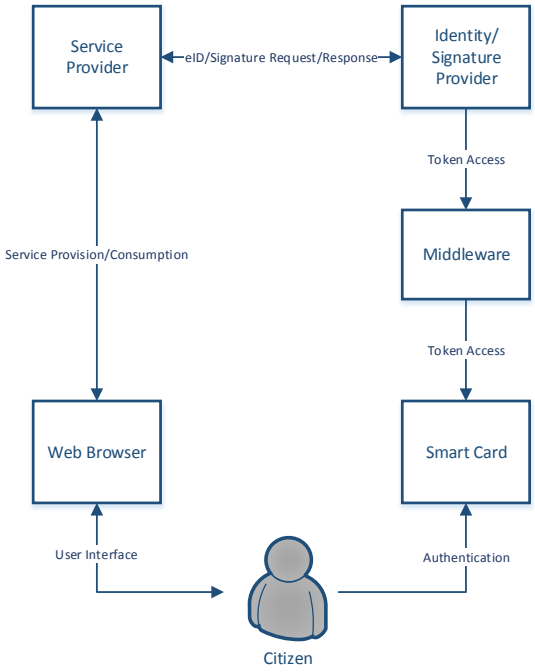

Figure 2.3: Assuming the use of smart cards, the abstract model for transactional e-government services can be refined accordingly.

Reliance on smart card based tokens is a rather old and hence time-tested approach. From a security perspective, smart cards represent an adequate technology, as they enable two-factor authentication. Provision of eID data and the creation of electronic signatures require possession of the smart card and knowledge of the secret PIN that protects data stored on and functionality provided by the card. Furthermore, smart cards can meet the requirements of QSCDs as defined by the eIDAS Regulation. Therefore, they are well-suited for the creation of legally binding electronic signatures according to EU law.

Even though the smart-card approach is sufficient from a functional and security perspective, it raises several problems from a usability point of view. Most of these problems are caused by the fact that citizens need to acquire, maintain, and use a card-reading device together with adequate software, i.e. middleware, in order to provide access to local smart cards. We have analyzed usability issues of smart card based solutions in detail by means of a comprehensive usability test [Zefferer and Krnjic, 2012a]. Results of this usability test show that smart card based solutions suffer from several usability limitations in practice.

Despite known usability drawbacks, smart card based eID and electronic-signature solutions are still frequently used throughout Europe. Countries that rely on this approach include for instance Aus-

tria[3], Belgium[4], Estonia[5], Germany[6], Portugal[7], or Spain[8]. All these countries supply citizens with eID/Signature Tokens in the form of personalized smart cards and provide various national e-government services that can be accessed with these cards.

2.1.3.2 Mobile Solutions

Disadvantages of smart card based solutions have become apparent in practice soon and have in many cases limited their user acceptance. Driven by the objective to replace smart card based approaches, mobile eID and electronic-signature solutions have been developed as an alternative in several countries. All mobile solutions have in common that mobile phones are used instead of smart cards. This sounds reasonable at a first glance, as mobile-phone coverage has already been above 100% in Europe in 2013 [GSMA, 2013] and mobile devices are increasingly gaining popularity. However, replacing smart cards with mobile phones also raises additional issues, as requirements defined by relevant legal frameworks still need to be met. In the EU, mobile eID and electronic-signature solutions need to fulfill the requirements for the creation of legally binding electronic signatures as defined by the eIDAS Regulation [The European Parliament and the Council of the European Union, 2014]. During the past years, two basic approaches for mobile eID and electronic-signature solutions have evolved that are able to meet these requirements.

The first approach makes use of the Subscriber Identity Module (SIM) that is part of virtually each mobile phone. The SIM stores the unique International Mobile Subscriber Identity (IMSI) to identify and authenticate the user at the mobile network. Access to the SIM and its functionality is typically protected by a PIN, which has to be entered by the user. This way, the user authenticates at the SIM and in further consequence at the mobile network. In mobile networks based on the Global System for Mobile Communications (GSM), the SIM is nowadays a mandatory component of every mobile phone. Special cryptography-enabled SIMs, which feature required cryptographic functionality, can also be used to store eID information and to create electronic signatures. This enables these SIMs to act as eID/Signature Token as a substitute to smart cards.

Assuming reliance on a special cryptography-enabled SIM, the abstract model shown in Figure 2.2 can be further refined. This yields the SIM-specific model illustrated in Figure 2.4, which shows that SIM-based solutions require an additional player, i.e. the Mobile Network Operator (MNO). Only the MNO is able to access the citizen's SIM in order to read stored eID data or to trigger the creation of electronic signatures. Hence, the Identity/Signature Provider cannot access the SIM directly but has to route respective requests through the MNO.

Conceptually, SIM-based mobile eID and electronic-signature solutions resemble their smart card based pendants. The SIM can be regarded as special kind of smart card. Under this presumption, the mobile phone assumes the role of the card-reading device. Consequently, the MNO implements the functionality of the middleware component. Despite these similarities, SIM-based mobile eID and electronic-signature solutions have several advantages compared to smart card based approaches. First and foremost, citizens do not need to acquire, install, and maintain card reading devices and local software to provide smart-card access. This way, SIM-based solutions can also be used in scenarios and use cases, where these kinds of hardware and software are not available. Furthermore, SIM-based solutions have been designed and developed for a use on two separate end-user devices. While the User Client, e.g. a web browser, is run on a classical end-user device such as a desktop PC or laptop, the eID/Signature Token is implemented by the physically detached mobile phone. This is advantageous in terms of secu-

[3] http://www.buergerkarte.at/en/index.html
[4] http://eid.belgium.be
[5] http://www.id.ee/?lang=en
[6] http://www.personalausweisportal.de
[7] http://www.cartaodecidadao.pt/
[8] http://www.dnielectronico.es/

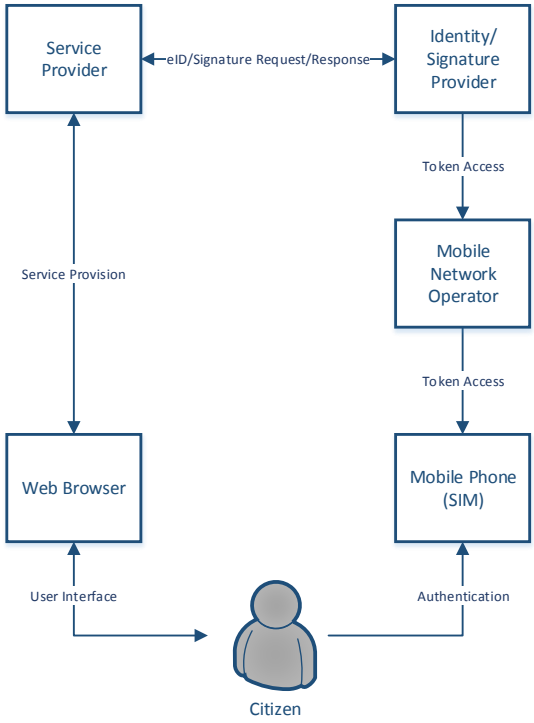

Figure 2.4: Assuming the use of SIMs, the abstract model for transactional e-government services can be refined accordingly.

rity, as attacks need to target two separate devices. Thus, SIM-based approaches can be regarded to be at least as secure as smart card based solutions and to improve usability at the same time.

Despite their advantages, SIM-based solutions also suffer from several drawbacks. First, they require cooperation of the citizen's MNO, as only the MNO is able to access the SIM. This reduces the applicability of this approach to citizens, whose MNOs provide support for SIM-based eID and electronic-signature services. Furthermore, SIM-based solutions require special SIMs that support required cryptographic functionality. Hence, citizens usually need to replace their original SIM, if they wish to use SIM-based eID and electronic-signature functionality. This potentially induces additional effort and causes additional costs.

Even though SIM-based mobile eID and electronic-signature solutions come with several disadvantages, several European countries still rely on this approach. Examples are Estonia[9], Finland[10], Norway[11], or Turkey[12]. In all these countries, SIM-based mobile eID and electronic-signature solutions have been successfully deployed and are in productive operation. In many cases, these solutions complement existing smart card based approaches and provide citizens an alternative method to use the respective national eID and electronic-signature infrastructure. The popularity and relevance of SIM-based eID and electronic-signature solutions is also emphasized by the fact that the European Telecommunication Standards Institute (ETSI) has been maintaining a respective standard [ETSI, 2003] for several years.

As an alternative to SIM-based mobile eID and electronic-signature solutions, server-based mobile approaches have recently gained relevance and popularity. They represent the second basic approach for

[9] http://e-estonia.com/component/mobile-id/
[10] http://www.mobiilivarmenne.fi/fi/bulletin/mobile-verification-launch-in-finland
[11] https://www.bankid.no/
[12] http://www.turkcelltech.com/Product.aspx?Id=5eba6d36-dba7-42be-a499-cea1422d0a3d&PId=15c36401-ddf8-424f-8489-41077a2974ab

mobile eID and electronic-signature solutions that has evolved during the past years.

The server-based approach aims to overcome limitations and drawbacks of SIM-based approaches. In contrast to smart card based solutions and also to the SIM-based approach, server-based eID and electronic-signature solutions store eID data and cryptographic signing keys remotely in a central server. In most cases, a Hardware Security Module (HSM) is used for this purpose. HSMs are secure hardware devices that support the secure storage of data and a set of cryptographic operations such as the creation of electronic signatures. Reliance on a secure central hardware element reduces the requirements for the user's mobile device. In particular, there is no need for local hardware or software installations. Furthermore, users do not need to exchange their SIM, as it does not need to support cryptographic functionality.

The general architecture of server-based mobile eID and electronic-signature solutions is shown in Figure 2.5. This architecture is again a refinement of the abstract model sketched in Figure 2.2 on page 26 and illustrates the main differences between server-based and SIM-based approaches. As shown in Figure 2.5, server-based solutions include an additional component, i.e. the central HSM. Both server-based and SIM-based solutions require the user to possess and use a mobile device. However, the function of the mobile device is different. Following the SIM-based approach, the mobile device, i.e. the SIM, stores required data and creates electronic signatures. Following the server-based approach, the mobile device is solely used to authorize eID and electronic-signature related processes carried out in the central HSM. Many server-based signature solutions rely on an approach based on Short Message Service (SMS) for this purpose. Following this approach, a one-time password—a so-called Transaction Number (TAN)—is sent to the user's mobile phone via SMS. The user has to enter this TAN into his or her User Client in order to prove reception of the TAN and hence possession of the mobile phone. Consequently, also server-based solutions integrate the users' mobile phones. However, the mobile phones are only required to be capable of receiving SMS messages. Support for the secure and reliable storage of data and for cryptographic functionality is not required.

The probably best known and most frequently used server-based mobile eID and electronic-signature solution is in productive use in Austria and is called Austrian Mobile Phone Signature[13]. This solution is based on a concept that has been introduced and discussed by Orthacker et al. [2010] in detail. The Austrian Mobile Phone Signature has been set into productive operation in 2010 and is operated by the Austrian CA A-Trust[14]. It allows Austrian citizens to reliably authenticate at e-government services and at various private-sector services. Furthermore, it enables the creation of qualified electronic signatures according to the eIDAS Regulation. As such, the Austrian Mobile Phone Signature complements the Austrian e-government infrastructure and offers Austrian citizens an alternative to smart card based eID and electronic-signature solutions.

A commercial server-based signature solution is also offered by Intesi Group SpA[15]. The solution is called PkBox[16] and enables the remote creation of qualified electronic signatures in case it is used in combination with qualified electronic certificates. Similar to the Austrian Mobile Phone Signature, PkBox relies on two-factor authentication to protect centrally stored data from unauthorized access. In this regard, PkBox follows a similar concept as the one proposed by Orthacker et al. [2010]. However, it shows a slightly higher degree of flexibility, as it supports different methods to cover the second authentication factor.

[13] http://www.handy-signatur.at/
[14] https://www.a-trust.at/
[15] http://www.intesigroup.com/en/intro.php
[16] http://www.pksuite.it/eng/pr_pkbox.php

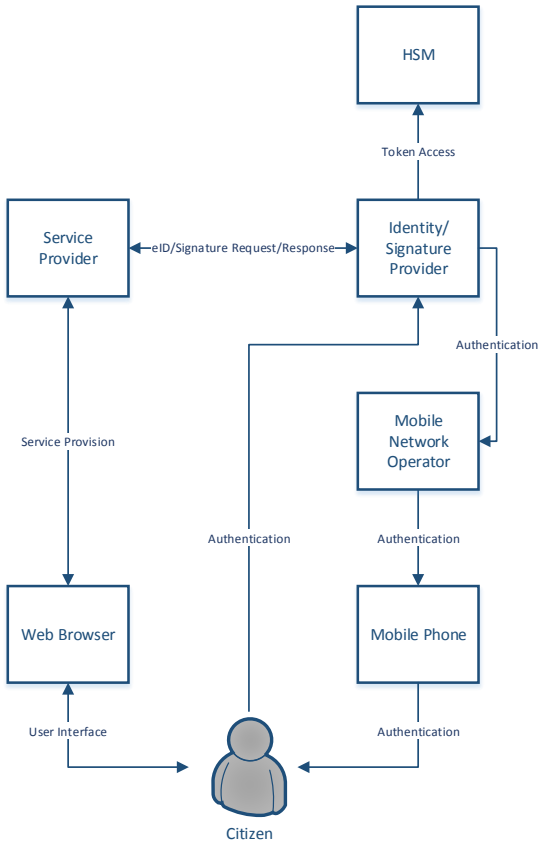

Figure 2.5: Assuming the use of a central HSM, the abstract model for transactional e-government services can be refined accordingly.

2.2 From E-Government to Mobile Government

Having their roots back in the late 1990's, most e-government concepts have been designed with classical end-user devices in mind. During the early noughties, when several European countries started to deploy e-government infrastructures on a large scale, desktop PCs still represented the most commonly used end-user devices. Mobile computing was basically limited to laptops and notebooks. Simple mobile phones were already available but basically restricted to telephony, text messaging, and rudimentary pre-installed applications. Modern mobile-computing approaches based on smartphones and other powerful mobile end-user devices were not envisaged in these times. It is hence unsurprising that basic e-government concepts and early e-government solutions have been mainly tailored to classical end-user devices such as desktop PCs or laptops. A representative example are smart card based eID and electronic-signature solutions, which require an end-user device that can be equipped with a card-reading device. The long lasting focus on classical end-user devices has only changed recently with the introduction of powerful mobile end-user devices and the subsequent emergence of the mobile computing paradigm.

2.2.1 Adopting the Mobile-Computing Paradigm

PCs and laptops have been dominating the end-user device market for many years. In 2007, this classical computing paradigm has suddenly changed with the introduction of the Apple iPhone[17]. The introduction of this device has heralded the success story of smartphones and has paved the way for other powerful

[17] https://www.apple.com/at/iphone/

mobile end-user devices such as tablet computers. At its time of introduction, the Apple iPhone was quite innovative. Compared to simple mobile phones, which were state of the art at that time, the Apple iPhone provided a new touchscreen-based user interface, enhanced hardware capabilities, and the opportunity to dynamically extend pre-installed software by downloading mobile apps from a centrally managed application store.

The immediate success of the Apple iPhone and the subsequent emergence of the mobile-computing paradigm has also been facilitated by the introduction and broad deployment of the Third Generation (3G) of mobile-networking technologies. 3G technologies include the Universal Mobile Telecommunications System (UMTS), High Speed Downlink Packet Access (HSDPA), and CDMA2000. In contrast to their predecessors such as GSM, 3G technologies provide higher data throughput and enable broadband Internet access for mobile end-user devices. The availability of powerful mobile networks and innovative mobile end-user devices has finally leveraged the mobile-computing paradigm on a large scale.

Soon after the introduction of the Apple iPhone, other vendors introduced similar concepts and solutions. Powered by Google, Android[18] emerged as an early rival of Apple's iPhone and its underlying mobile operating system Apple iOS[19]. Android has been available as operating system for smartphones since 2008. It is based on the Linux kernel and is developed following the open-source model. Even though Android has been launched approximately one year after the iPhone, it already had a global market share of more than 80% in the second quarter of 2014 [IDC, 2014]. Today, Apple iOS and Google Android are dominating the market of mobile operating systems for smartphones and tablet computers. Other operating systems such as Microsoft Windows Phone 8[20] or BlackBerry[21] play a minor role only.

Smartphones, tablet computers, and their underlying mobile operating systems are continuously improved. Every generation of mobile end-user devices outperforms its predecessor in terms of available processing power and storage capabilities. Accordingly, every new release of a mobile operating system includes new features and functionality. Similarly, also mobile apps become more and more powerful and in turn require more powerful hardware. With their continuously growing capabilities, smartphones and related mobile end-user devices have become an interesting alternative to classical end-user devices during the past years. Indeed, current surveys show that mobile devices are gradually replacing desktop PCs and laptops [Nielsen, 2014]. This especially applies to use cases, in which the end-user device is mainly used to consume content such as videos or websites, and to applications that require only simple user interactions. Assuming a continuing improvement of mobile devices, it can be expected that the practical relevance of classical end-user devices will further decrease and that the mobile computing paradigm will continue to gain relevance. This trend is also increased by continuous improvements of mobile communication networks all over the world. For instance, in various countries established 3G technologies are currently complemented by the even more powerful Fourth Generation (4G) technologies Long Term Evolution (LTE) and LTE-Advanced. The availability of modern mobile end-user devices with continuously increasing capabilities and the availability of mobile broadband communication networks contribute to the growing popularity and increasing relevance of mobile computing.

The general paradigm change towards mobile computing has had a significant impact on the way users access information and services. Nowadays, users show an always-on behavior. This means that they are typically online and available 24/7 and also expect information and services to be available all the time. As mobile devices are increasingly used to access information and services, provided services need to be tailored to their special characteristics. For instance, websites nowadays often feature a responsive design, in order to assure that presented content is arranged and displayed dynamically depending on the accessing end-user device and its screen size. This also applies to user interfaces of web applications that are expected to be accessed by means of a web browser. In many cases, service providers additionally

[18] http://www.android.com/
[19] https://www.apple.com/at/ios/
[20] http://www.windowsphone.com
[21] http://blackberry.com/

offer a mobile app as alternative to a web-based user interface. This is especially useful for applications that require complex user interactions, which are easier to implement in a user-friendly way by means of mobile apps. A popular example are web-mail services such as Google Gmail[22] or Yahoo Mail[23]. Originally, these services have been designed with a web-based user interface and hence represent classical web applications. To improve user experience for smartphone and tablet users, Google and Yahoo now also provide mobile apps. These apps enable users to efficiently access their e-mails without the need to use a web-based user interface on a potentially small touchscreen. Google's and Yahoo's e-mail services are just two out of many available applications that have been adapted for an efficient use on mobile end-user devices as a reaction to the emerging mobile-computing paradigm. In general, commercial applications have in most cases already been adapted to deal with limitations of mobile end-user devices and to exploit additional opportunities that arise from a use of powerful mobile end-user devices.

2.2.2 Mobile Government

Similar to private-sector companies offering commercial applications, also the public sector is requested to react to the emerging mobile-computing trend. In particular, existing e-government services, which have in many cases originally been implemented as web applications, need to be adapted for a use on smartphones and related mobile end-user devices. Furthermore, the integration of powerful mobile end-user devices offers new opportunities, as these devices feature technologies that are typically not available on PCs or laptops. For instance, e-government applications for smartphones can make use of locally available positioning information in order to provide location-based services. Thus, the mobile-computing paradigm does not only require an adaptation of existing services, it also offers opportunities for new and innovative solutions.

The idea to make use of mobile devices and technologies in e-government solutions was actually born long before the first smartphones were available on the market. Mobile eID and electronic-signature solutions are a representative example for e-government solutions relying on simple mobile technologies. The usage of mobile technologies in e-government solutions has soon become known under the term mobile government or m-government, which is basically a combination of the two terms mobility and e-government. Although this definition seems quite clear and unambiguous, several slightly different definitions of the term m-government can be found in literature. For instance, Kushchu and Kuscu [2004] state that m-government *'may be defined as a strategy and its implementation involving the utilization of all kinds of wireless and mobile technology, services, applications and devices for improving benefits to the parties involved in e-government including citizens, businesses and all government units'*. In contrast, Antovski and Gusev [2005] state that *'m-Government is largely a matter of getting public sector IT systems geared to interoperability with citizen's mobile devices'*. A similar definition is also provided by Carroll [2005], who states that *'m-Government involves the provision of public sector services via mobile technologies'* but amends that *'m-Government involves interaction where the contexts are unknown, where accessing government services might be one of several activities being undertaken and where the physical constraints of interacting with mobile devices limit the amount and type of information that might be located and accessed'*. Finally, Misra [2010] provides a rather compact definition of the term m-government by stating that *'m-Government is public service delivery including transactions on mobile devices like mobile phones, pagers, and PDAs'*.

Most definitions of the term m-government stem from related scientific publications on this topic. The numerousness of publications on this topic shows that m-government has always been a topic of scientific interest. We have provided an extensive overview on scientific publications related to m-government in a survey prepared for the Secure Information Technologies Center – Austria (A-SIT) [Zefferer, 2011]. We have also published relevant findings of this survey in the European Journal of ePractice [Zefferer and Teufl, 2011]. Our surveys show that many publications on m-government focus

[22] http://www.gmail.com
[23] http://mail.yahoo.com/

on the introduction and discussion of particular m-government solutions and applications. Examples are published scientific work by Yoojung et al. [2004], in which they discuss an architecture for implementing m-government services in Korea, or by Scholl et al. [2006], in which Seattle's Mobile City project is introduced. An m-government solution for Dubai has been introduced by Alrazooqi and De Silva [2010]. The situation of m-government in Greece has been discussed by Karadimas et al. [2008]. Recently, the demand for m-government services in Japan has been analyzed by Madden et al. [2013]. These examples show that scientific publications on m-government often focus on particular use cases, solutions, or countries.

In addition to the introduction of concrete m-government solutions, recent scientific work has also focused on the identification of general success factors of and potential barriers to m-government. These works are relevant, as their findings build the basis of successful m-government services and are hence also important for the scope of this thesis. Results of these works are discussed in more detail in the following.

2.2.3 Success Factors of Mobile Government

The consideration of success factors has soon been identified as crucial requirement for the development, deployment, and operation of m-government solutions. Only if relevant success factors are considered, m-government services will achieve an adequate level of user acceptance. The identification of relevant success factors has hence been a topic of scientific interest for many years.

For instance, Karan and Khoo [2008] identify infrastructural investment, regulatory and political environment, awareness and acceptance, security and privacy, and equitable access as success factors for m-government. In contrast to this rather coarse-grained classification, Al-khamayseh et al. [2007] provide a more detailed list of relevant success factors. They list privacy and security, infrastructure, user needs and preferences, quality and user friendly applications, e-government, acceptance, cost, standards and data-exchange protocols, coherent m-government framework, high mobile penetration, infrastructure management, m-government awareness, access, strategy, IT literacy, m-government portals and exclusive gateways, private sector partnerships, and legal issues as relevant factors that influence the success of m-government solutions. Similar success factor as those listed by Karan and Khoo [2008] and Al-khamayseh et al. [2007] have also been identified by El-Kiki and Lawrence [2006]. Sareen et al. [2013] have focused on the Indian use case and have identified user-acceptance factors especially for Indian m-government services. Identified success factors basically resemble those identified by other works, even though several aspects that are special for developing countries have been taken into account.

A comprehensive survey on literature on the identification of success factors for m-government solutions has been provided by Al-Hadidi and Rezgui [2009]. This survey shows that most scientific publications on the identification of success factors for m-government solutions have in common that the factors security, privacy, awareness, and user acceptance are top-ranked and regarded as critical. El-Kiki [2007] and Carroll [2005] have especially focused on the factor user acceptance. For this factor, the aspects availability, effort involved, convenience, input and output mechanisms, privacy and security issues, and lack of public-sector services have been identified to have an impact on the user acceptance of m-government solutions.

In summary, related scientific work on success factors of m-government solutions reveals that despite minor country-specific differences, factors influencing security and usability are in general crucial for the success of m-government solutions. These factors have already been identified in older publications stemming from the pre-smartphone era. In these times, mobile phones have been implicitly assumed to be secure. Modern smartphones are different in this regard, as they support the installation of additional software in the form of mobile apps. This increases functionality but at the same time makes these devices more prone to malware as well. This again increases the relevance of the success factor security when m-government solutions are designed for smartphones and related powerful mobile end-user devices.

2.2.4 Barriers to Mobile Government

Beside the identification of success and acceptance factors, scientific work on m-government has also focused on the identification of potential barriers and challenges that need to be overcome by m-government solutions. Similar to the identification of success factors, investigation of potential barriers has started early. In the following, a brief overview of related scientific work on barriers to m-government is provided.

Kumar and Sinha [2007] have identified two potential barriers of m-government solutions. Concretely, they have mentioned security issues imposed by airwave-based communication channels and accessibility issues imposed by technically limited end-user devices.

A more comprehensive analysis of potential barriers to m-government has been provided by El-Kiki [2007]. He classifies identified barriers into the categories Organizational, Technical, Governance, and Social. According to El-Kiki [2007], organizational barriers include issues such as bureaucratic problems, the lack of cooperation among public organizations or interoperability issues between different departments. Another identified organizational barrier is the lack of user-centric approaches. According to the author, governments often take *'citizens as granted, thinking that they will accept and use a new service as long as it is provided by the government'* and the offered *'service is structured by the goals of the administration, not the goals of the citizen users'* [El-Kiki, 2007]. Further organizational barriers to m-government identified by the author are the absence of combined e-business and e-governance models as well as the lack of sustainable business models. Additionally, the reluctance of authorities to alter traditional ways of dealing with their customers is also identified as a potential barrier. El-Kiki [2007] further states that sometimes also economic and financial barriers hinder the success of m-Government services. In this regard, high development costs, lack of infrastructural investments, and low budget for mobile services are the most frequently listed issues. Finally, El-Kiki [2007] also identifies legal problems as barrier to the success and acceptance of m-government in practice. These problems can arise, if required legal frameworks are missing.

Beside these organizational barriers, also technical barriers can limit the success of m-government according to El-Kiki [2007]. Short release cycles and frequent advances in the mobile computing domain are often difficult to be followed by end users. This may result in a lack of familiarity with mobile technologies in general, but also in a lack of technical knowledge among IT personnel. Other technical barriers identified by El-Kiki [2007] are the lack of interoperability, the competition between access channels, but also the lack of backend-process integration and the absence of ability to bundle information, materials, and service together.

According to El-Kiki [2007], social barriers also reduce the acceptance of m-government services. Hence, these services must be as simple to use as possible, in order to make them accessible to all people. In this context, the author states that beside usability, it is also relevant that people understand why they should use a mobile service. Other identified social barriers to m-government are security and privacy concerns. El-Kiki [2007] states that privacy fears and security concerns are a substantial barrier. The author points out the relevance of security for m-government by concluding that *'if there is no sound solution to security, e-government and m-government will be a dream'* [El-Kiki, 2007].

While El-Kiki [2007] provides a rather generic list of potential barriers, other publications focus on more specific use cases and scenarios. For instance, Moon [2010] focuses on mobile emergency systems in Asia and identifies potential issues and challenges for them. From the identified challenges, Moon [2010] derives four recommendations for facilitating and improving the implementation of m-government initiatives. While Moon [2010] focuses on a certain use case, Mengistu et al. [2009] focus on challenges that need to be overcome when deploying m-government services in developing countries. Interestingly, only few scientific work has been available on potential barriers to m-government assuming a use of powerful mobile end-user devices such as smartphones. To bridge this gap, we have investigated potential challenges of smartphone-based m-government solutions [Zefferer and Teufl, 2011]. In particular, we have identified the heterogeneous ecosystem of different mobile platforms as well as security and

privacy issues on mobile end-user devices as potential barriers to the future success of m-government.

2.2.5 Relevant Aspects for Mobile-Government Solutions

The large number of related scientific publications shows that m-government has always been a topic of scientific interest. Beside related scientific work that introduces, discusses, and assesses concrete m-government solutions, especially the identification of success factors and potential barriers has been a topic of interest during the past years. From these publications and from own work on this topic, a set of relevant aspects, which need to be taken into account during design, development, deployment, and operation of m-government solutions, can be derived. Unfortunately, the derivation of a common set of relevant aspects is complicated by the fact that different authors usually approach the same issue on different levels of abstraction. Thus, also relevant success factors and barriers are identified on a different level of abstraction by different authors. To combine results of related scientific work, a sufficiently high abstraction level hence needs to be chosen. This leads to the following set of relevant aspects that have to be taken into account:

- **Security and privacy:** Security and privacy are mentioned by nearly all authors of related scientific work as crucial success factors. These aspects are for instance mentioned by Karan and Khoo [2008], Al-khamayseh et al. [2007], El-Kiki and Lawrence [2006], Sareen et al. [2013], and Al-Hadidi and Rezgui [2009]. Furthermore, a lack of security or privacy is also frequently mentioned as potential barrier to m-government solutions [Kumar and Sinha, 2007] [El-Kiki, 2007].
- **Usability and accessibility:** Usability and accessibility are also frequently listed as relevant aspects for m-government services and solutions. Examples are works by Karan and Khoo [2008], Carroll [2005], and El-Kiki [2007]. The demand for usability and accessibility includes related aspects such as user acceptance, equitable access, convenience, or the need for user-centric approaches, which are also frequently mentioned in related work.
- **Legal framework:** A suitable legal framework such as an appropriate regulatory and political environment as defined by Karan and Khoo [2008] represents a mandatory basis for m-government services. The relevance of legal issues is also emphasized by Al-khamayseh et al. [2007]. Only if required legislations are in effect, m-government services can be used for productive use cases.
- **Organizational framework:** Appropriate organizational frameworks for m-government solutions include a suitable infrastructure management and private-sector partnerships [Al-khamayseh et al., 2007], appropriately combined Electronic Business (e-business) / Electronic Governance (e-governance) models, and sustainable business models El-Kiki [2007]. Similar to legal frameworks, also an appropriate underlying organizational framework represents a relevant aspect for m-government solutions.
- **Technical framework:** Finally, m-government solutions also require a technical framework, which they can rely on. According to surveyed related publications, a sound technical framework includes a stable e-government infrastructure, a high mobile penetration, m-government portals and exclusive gateways [Al-khamayseh et al., 2007], as well as a general familiarity with mobile technologies [El-Kiki, 2007], and solutions facilitating the handling of different mobile platforms and operating systems [Zefferer and Teufl, 2011].

These five aspects can be regarded as pillars of successful m-government solutions. Depending on the concrete solution and the use case, in which this solution is deployed, the relevance of these five pillars can vary. Nevertheless, all above-mentioned aspects should be taken into account during design, development, deployment, and operation of m-government solutions. In the next section, existing m-government solutions are surveyed in order to illustrate how these five pillars are taken into account in practice by current successful m-government solutions.

2.3 Mobile Government: State of the Art

Common definitions of the term mobile government are rather vague and abstract. Accordingly, the term m-government covers a broad spectrum of applications, services, and solutions. In this section, a brief overview of m-government solutions currently in productive operation is provided. This overview is primarily based on an own study on m-government [Zefferer, 2011] that has been prepared for the Secure Information Technology Center - Austria[24] and on an own survey on m-government [Zefferer, 2015a]. Additionally, publicly available studies of m-government solutions such as those provided by Mobi Solutions Ltd. [2010] or Jotischky and Nye [2011] are considered as well. Furthermore, also recent developments, which are not listed in the mentioned surveys, are taken into account. This way, a representative picture of the current m-government landscape is drawn and the current state of the art is presented.

Due to the rather generic definition of the term m-government, related services and solutions can be subdivided into different categories. Different approaches to classify m-government solutions have been proposed and applied in literature. For instance, Kumar and Sinha [2007] classify m-government solutions according to involved parties. A classification scheme based on employed user interfaces has been applied by Misra [2010]. Zálešák [2003] follows a different approach by classifying m-government services according to their purposes. The type of transaction is also frequently used as classification criterion. This approach has for instance been followed by Norris and Moon [2005], Sheng and Trimi [2008], or Hassan et al. [2009]. Finally, m-government services are sometimes also classified according to the particular public sector, in which they are deployed. Examples are the sectors health, education, or public administration. Accordingly, m-government services can be subdivided into Mobile Health (m-health), Mobile Education (m-education), or Mobile Administration (m-administration). Especially m-health and m-education services in developing countries are often deployed and operated by Non-Governmental Organizations (NGOs). For the sake of completeness, we take these services into account, even if they are not directly offered by public bodies.

To draw a comprehensive and global picture of the current state of the art, concrete m-government services and solutions from all five continents are surveyed in the following. The provided survey makes no claim to be complete, as an exhaustive list of all available solutions would go beyond the scope of this thesis. However, surveyed services have been selected such that a reasonably representative picture is drawn. This way, continent-specific differences and regional trends in m-government become clear.

2.3.1 M-Government in Africa

In many African countries, information and communication infrastructures are often still underdeveloped [International Finance Corporation, 2014]. Interestingly, this mainly applies to wired communication networks. In contrast, mobile communication networks are often already well developed. This makes m-government an interesting and often also the only possible alternative. It is hence unsurprising that African countries have been among the pioneers in deployment and usage of m-government solutions. A sample of these solutions, which the author of this thesis has introduced in more detail in Zefferer [2011], are briefly sketched in the following.

2.3.1.1 BloodBank SMS

The availability of blood transfusions is crucial for hospitals. This applies to developed countries as well as to developing countries. In Kenya, centralized blood banks are responsible for supplying district hospitals with blood. To assure an adequate supply, hospitals have to report the current status of their local blood repository frequently. Because of underdeveloped infrastructures, hospitals in rural areas of

[24] http://www.a-sit.at

Kenya sometimes suffer from a lack of reliable electricity and phone lines, which often renders frequent status updates impossible. To tackle this problem, BloodbankSMS[25] has been developed by Eric Magutu. This service allows health-care workers from remote hospitals to report the current status of their blood repository using SMS. Incoming status updates are collected at the central blood bank and graphically visualized through a web-based interface. SMS-based alerts are triggered automatically, in case blood repositories at district hospitals fall below a predefined threshold.

BloodBank SMS is a prime example for an m-health-related m-government service that exploits relatively well-developed mobile communication infrastructures of developing countries. Even though BloodBank SMS relies on simple technologies only, it significantly improves the availability of blood transfusions in Kenya.

2.3.1.2 Cell-Life

Cell-Life[26] is a South African non-profit company and public benefit organization. It focuses on the provisioning of solutions for the management of health in developing countries. As such, Cell-Life has started several health-related project during the past years. Many of them rely on mobile technologies. A complete list of all projects initiated by Cell-Life is provided on their website[27].

Projects relying on mobile technologies include a mobile app based drug stock management system[28], a mobile monitoring and reporting system for the national Human Immunodeficiency Virus (HIV) counseling and testing campaign[29], and a randomized controlled trial to evaluate the use of SMS technology in order to improve retention and to decrease treatment interruptions amongst patients initiating antiretroviral therapy[30].

The variety of projects carried out by Cell-Life underpins the importance of appropriate health services and solutions especially in developing countries. Furthermore, the frequent use of mobile technologies in these projects shows their usefulness and potential for the health sector especially in developing countries.

2.3.1.3 mPedigree

Counterfeit of legal drugs and medicines is a serious issue in developing countries [World Health Organization, 2014]. The mPedigree network[31] aims to tackle this issue by providing solutions to establish a secure supply chain from producers to consumers. Mobile technologies are used for this purpose. Consumers can send the serial number that is printed on the bought medicine to a central service via SMS. Information about the authenticity of the bought drug or medicine is then returned to the consumer via SMS. As an alternative to the SMS-based communication channel, also a web-based user interface is provided.

In general, the concept behind mPedigree can also be applied to other products than drugs or medicine. Nevertheless, mPedigree is another example showing that rather simple mobile technologies can be employed to efficiently achieve an important goal.

[25] http://www.media.mit.edu/ventures/EPROM/research.html#bloodbank
[26] http://www.cell-life.org
[27] http://www.cell-life.org/projects/
[28] http://www.cell-life.org/projects/mobile-app-drug-stock-management-system/
[29] http://www.cell-life.org/projects/health-care-and-testing/
[30] http://www.cell-life.org/projects/idart-and-sms-integration-research-for-cida/
[31] http://mpedigree.net/

2.3.1.4 RapidSMS

RapidSMS[32] was developed by the innovation unit of the United Nations Children's Fund (UNICEF)[33] in 2007. The main goal of RapidSMS was to support data collection and youth-engagement activities in regions with limited communication infrastructures. For such scenarios, RapidSMS provided an efficient way to exchange information using SMS technology. After several years of development, RapidSMS has evolved to an open-source framework for the rapid development of services based on mobile and web technologies. As such, RapidSMS is the basis of several m-government solutions including the Nigeria birth registration[34] and a school monitoring system rolled out in Uganda[35].

The continuing success of RapidSMS shows that SMS is still an important technology in developing countries. As SMS does not require mobile broadband networks and high data-transmission rates, it is an efficient and reliable technology for data exchange in regions with limited mobile communication infrastructures.

2.3.1.5 Text to Change

Text to Change (TTC)[36] started as a health-education initiative relying on mobile telephony. Using SMS-based health-education programs, TTC has aimed to inform people in developing countries about relevant health-related topics such as HIV, Acquired Immune Deficiency Syndrome (AIDS), or malaria.

TTC originally started in Uganda in 2008, but has soon expanded to other African countries and developing regions all over the world. Today, TTC comprises more than 100 projects. In total, more than 50 million text messages have been sent so far. Most of these projects focus on health-related topics. Current projects include a medical helpline in Ghana[37] and several Ebola-prevention campaigns[38].

The success of TTC underpins the fact that SMS is still one of the most important mobile technologies in developing countries. In this aspect, TTC is comparable to RapidSMS, which also relies on SMS technology and is frequently used in several developing countries in Africa.

2.3.1.6 Phones for Health

Phones for Health[39] is a public-private partnership that aims to connect health systems in 10 countries. By improving the health-care infrastructure, Phones for Health mainly aims to address the HIV/AIDS pandemic. Mobile technologies play a central role in this project. Phone for Health employs the wide spread of mobile communication networks in developing countries. Hence, these networks are one of the backbones of the developed health-care infrastructure. This again emphasizes the relevance of mobile technologies for health-care solutions in developing countries.

2.3.1.7 Collecting and Exchange of Local Agricultural Content

The Collecting and Exchange of Local Agricultural Content (CELAC) project[40] has been launched in Uganda and aims to improve the information flow to and between farmers. This is for instance achieved

[32] https://www.rapidsms.org
[33] http://www.unicef.org/innovation/
[34] http://rapidsmsnigeria.org/br
[35] http://edutrac.unicefuganda.org/
[36] http://www.ttcmobile.com/
[37] http://www.ttcmobile.com/newsitem/ttc-launches-medical-helpline-in-ghana/#more-2530
[38] http://www.ttcmobile.com/newsitem/ttc-ebola-prevention-campaigns-keep-growing/#more-2448
[39] http://www.pepfar.gov/c21414.htm
[40] http://www.celac.or.ug/

by broadcasting SMS messages with valuable farming tips and related information about growing lucrative export crops. The exchange of information between farmers in Uganda shall also help them to specialize in new and potentially more lucrative ventures.

CELAC shows that mobile technologies are not only relevant for the health-care sector. In addition, also the agricultural sector can benefit from mobile communication technologies, which provide means to exchange information in an efficient, timely, and cheap way.

2.3.1.8 m-Pesa

The m-Pesa project[41] offers a mobile phone based non-cash money-transfer system. The solution was developed by the Kenyan mobile-network operator Safaricom[42] and by Vodafone[43] in 2007. It is mainly intended for people with low income, who cannot afford an own bank account, or for those, who have no access to financial infrastructures.

For each m-Pesa user, a virtual account is electronically created and maintained. Special m-Pesa agents such as super markets or gas stations act as interface between users and their accounts. By means of these agents, users can deposit or withdraw money from their virtual accounts. In addition, m-Pesa enables money transfers to other users, supports the payment of bills, and allows users to purchase prepaid airtime. Transactions are carried out by exchanging SMS messages between users or between users and agents. Required business logic on mobile devices is implemented by means of special SIM applications.

Having its roots in Kenya, m-Pesa has already been expanded to other African countries. Since 2008, a slightly modified version of m-Pesa has been for instance available in Tanzania. The introduction of m-Pesa in other countries such as India and Egypt is already scheduled.

2.3.2 M-Government in America

Compared to Africa, the situation in America is sort of different. Especially in North America, both wired and mobile communication infrastructures are usually well developed. Especially in urban areas, 3G and 4G networks are available and enable instant Internet access for mobile end-user devices. In contrast to developing countries, where simple mobile communication infrastructures are often the only alternative, mobile communication networks are only one out of many alternatives in developed countries. Accordingly, also the role of m-government and the motivation to use m-government solutions are different. The probably most important reason for m-government in developed countries is convenience. In the typical western always-on society, people are used to access services anytime and everywhere. As mobile devices such as smartphones are usually always carried and always on, these devices are perfectly suitable to act as end-point for services that shall be available and accessible independently of the user's current location and context.

In South and Central America, the situation is slightly different. In these regions, well-developed communication infrastructures are often limited to urban areas. This also influences m-government services and solutions that are provided in these regions. In regions with underdeveloped communication networks, the situation is hence basically comparable to Africa. To provide a comprehensive overview of m-government solutions in America, some of the services offered on this continent are briefly sketched in the following.

[41] http://www.safaricom.co.ke/personal/m-pesa/m-pesa-services-tariffs/relax-you-have-got-m-pesa
[42] http://www.safaricom.co.ke
[43] http://www.vodafone.com

2.3.2.1 United States Federal Mobile Apps Directory

In the United States (US), various mobile applications are provided by different public agencies and administrations. A complete overview of available applications and services is provided at the website of the US Government[44]. Offered services and applications include native mobile apps, hybrid apps, as well as responsive and mobile websites that are optimized for a use on mobile end-user devices.

The US Government currently lists more than 200 mobile applications and websites. Most of them offer pure informational services, i.e. provide relevant information on a particular topic in a structured way. Examples are apps provided by the Bureau of Alcohol, Tobacco, Firearms and Explosives (ATF)[45] or apps offered by the Centers for Disease Control and Prevention[46]. In addition, several mobile apps have been published that provide simple tools. Examples are an app that enables the calculation of the body mass index[47] or the app Tactical Breather[48], which helps users to gain control over physiological and psychological responses to stress. Furthermore, several location-based apps are provided that assist citizens in navigating to public offices and other points of interest. Only few apps enable citizens to directly communicate with public administrations. One of a few exceptions is the app iBurgh[49], which is offered for citizens of the city of Pittsburgh, Pennsylvania. This app can be used to report issues in public space. Using the app, users can take a photo of e.g. a found pothole and send this photo together with a brief description to the responsible municipality.

Even though the US Government offers citizens a broad spectrum of mobile apps and websites, complete transactional services are hardly available for mobile devices. Even partly interactive apps such as the issue-reporting solution iBurgh do not implement complete electronic transactions. Instead, app-based m-government solutions mainly focus on the provision of relevant information and useful tools.

2.3.2.2 Apps of the Government of Canada

Similar to the US Government, also the Canadian Government offers citizens various mobile applications[50]. Provided apps cover different fields and in most cases provide citizens with relevant information on a certain topic. For instance, there is an app for Google Android and Apple iOS guiding applicants through a website that enables application for a grant[51]. Another app called Learn to Camp[52] provides useful information on camping in Canadian national parks.

Similar to apps provided by the US Government, hardly any app provided by Canadian public administrations implements complete transactional services. Instead, focus is mainly put on the provision of information and useful tools.

2.3.2.3 Text4Baby

Text4Baby[53] is an SMS-based service provided in the US for pregnant women. The service supplies registered users with relevant information during pregnancy and provides a simple SMS-based notification and reminder service. Text4Baby is free of charge and is provided by the National Healthy Mothers

[44] http://www.usa.gov/mobileapps.shtml
[45] https://play.google.com/store/apps/details?id=com.nicusa.atf
[46] https://play.google.com/store/apps/details?id=gov.cdc.general&feature=search_result#?t=W10
[47] https://itunes.apple.com/us/app/bmi-calculator/id292796789?mt=8
[48] https://itunes.apple.com/app/tactical-breather/id445893881?mt=8
[49] https://play.google.com/store/apps/details?id=com.yinzcam.iburgh
[50] http://data.gc.ca/apps
[51] http://open.canada.ca/en/apps/how-apply-grant
[52] http://open.canada.ca/en/apps/learn-camp
[53] https://www.text4baby.org/

Healthy Babies Coalition[54].

Text4Baby has originally been launched as a pure SMS-based service. Today, the service can also be accessed by means of a mobile app. In this regard, Text4Baby is a prime example of m-government services in developed countries. Although services provided in these countries often employ cutting-edge technologies such as mobile apps, also established and approved technologies like SMS are still frequently used.

2.3.2.4 Project Patojitos

Patojitos is a pilot project that will be launched in Guatemala by the SHM Foundation[55]. According to the project website[56], Patojitos *'aims to study the impact of ICTs on health service delivery in remote and underserved areas'*. In particular, the project will focus on different interventions that could help to support breastfeeding practices among mothers in Guatemala through the use of mobile technology.

Project Patojitos is a prime example showing that also in developing countries located in America, the health-care sector aims to benefit from mobile technologies and related m-government solutions. In this regard, parallels can be identified between developing countries in America and Africa.

2.3.2.5 Zumbido

The project Zumbido[57] was run in the state of Jalisco, Mexico. It used group mobile-phone communication based on SMS technology with the goal to address challenges faced by citizens suffering from HIV and AIDS. Zumbido is hence another example of an m-government service that has been tailored to the needs of the health sector and aims to improve health care in a developing country.

2.3.2.6 Cell-PREVEN

Another health care related m-government project that has been started and run in Peru during the past years is Cell-PREVEN[58]. Cell-PREVEN is an interactive voice-response system that allows health workers in the field to access a central database using their mobile end-user devices. Data being stored in this database can be accessed and can be remotely supplemented by collected data samples. All data transmissions can be carried out with off-the-shelf mobile phones. The project has been introduced and discussed in detail by Curioso et al. [2005].

2.3.3 M-Government in Asia

In Asia, the situation is roughly comparable to the one in America. On the one hand, there are several developing countries such as Bangladesh or the Philippines. On the other hand, several well-developed countries such as Japan or Korea are also located in Asia. Unsurprisingly, the stage of development has an impact on m-government services provided in the respective country. This also becomes apparent from the sample of Asian m-government services and initiatives that are briefly sketched in the following.

[54] http://www.hmhb.org/
[55] http://shmfoundation.org
[56] http://shmfoundation.org/?page_id=304/
[57] http://shmfoundation.org/?page_id=323
[58] http://www.waltercurioso.com/preven/

2.3.3.1 Chinese Aged Diabetic Assistant

The project Chinese Aged Diabetic Assistant (CADA)[59] investigated possibilities to support elderly diabetics in China. For this purpose, smartphones were used to supply patients with relevant information and guidelines. Furthermore, users could enter data on measured glucose levels using their smartphones. This allowed for a remote surveillance and tracking of patient data. This way, a complex interactive diabetes self-management support system based on mobile end-user devices was set up.

CADA shows that health care related m-government solutions are also required in Asian countries. This especially applies to developed countries, where mobile technologies can be useful to improve the quality of life for patients.

2.3.3.2 Mobile Solutions for Indian Fishermen

At the Indian coast of Kerala, mobile technologies have early been used to better the life of fishermen. Since 2006, mobile communication technologies are used to inform fishermen about current market situations while still being at sea. This has enabled them to divert their boats to those markets that pay best. According to an article of the Washington Post[60], the increased functioning of the market has raised fishermen's profit by 8%, while at the same time consumer prices fell by 4% on average.

2.3.3.3 Text2Teach

Text2Teach can be assigned to the field of m-education. According to the project website[61], *'the mission of Text2Teach is to make a significant contribution to the quality of teaching and learning in underserved schools and communities in the Philippines'*. The Text2Teach project is based on the Bridgeit initiative[62] and supplies teachers in underdeveloped regions with informative video clips that can be shown in class.

The project basically consists of two phases. In the first phase, teachers can order video clips using their mobile phones. The video itself is then transmitted over a satellite link. In the second phase, teachers can load educational material directly on their mobile phones. These phones are then connected to the Television (TV) set in class. New material can instantly be downloaded through the mobile phone using 3G technologies.

Text2Teach is in productive operation in the Philippines. Since its introduction in 2003, more than 1,600 teachers and 57,000 students have benefited from this project. Text2Teach is hence a prime example of an m-education solution that improves the educational system of developing countries.

2.3.3.4 Disaster-Alert Systems

Disaster-alert systems based on mobile communication technologies have become popular especially in Asia, as this region of the world is frequently affected by floods, tsunamis, typhoons, and other natural disasters. Mobile communication technologies such as SMS are frequently used to warn people of imminent disasters and to provide them with urgent information.

There are various examples for productive disaster-alert systems in Asian countries. A cyclone-alert system that relies on mobile technologies has for instance been deployed in Bangladesh[63]. The websites SMS Tsunami Warning[64] or Tsunami Alarm System[65] offer similar services, but mainly focus on tsunami

[59] http://www.cadaproject.com/
[60] http://www.washingtonpost.com/wp-dyn/content/article/2006/10/14/AR2006101400342.html
[61] http://www.text2teach.org.ph
[62] http://www.educationinnovations.org/program/bridgeit
[63] http://phys.org/news165066573.html
[64] http://www.sms-tsunami-warning.com
[65] http://www.tsunami-alarm-system.com/

warnings. Similar solutions focusing on different types of natural disasters have also been introduced in countries outside of Asia. For instance, an SMS-based flood-warning system has been set up in the city of Venice, Italy, which is also frequently affected by floods[66].

All over the world, mobile technologies have turned out to provide efficient means to inform people of imminent threats in time and hence build the backbone of established disaster-alert systems. The key advantage of mobile communication technologies is the capability to reach citizens anytime and everywhere, as long as they carry their mobile end-user devices. This advantage is employed by disaster-alert systems, for which timely communication is key.

2.3.3.5 PayBIR

The Philippine Bureau of Internal Revenue offers a service called PayBIR[67] that allows Philippine citizens to pay their taxes with the help of mobile phones. Technically, PayBIR makes use of G-Cash technology[68], a micro payment service that turns a mobile phone into a virtual wallet. Initially being developed for business registrations, the PayBIR service has later been extended to cover a broader set of taxes.

Today, PayBIR increases the efficiency of tax-related tasks in the Philippines and helps to reduce costs for both citizens and administrations. PayBIR is a representative example for an m-government solution that can be assigned to the field of m-administration.

2.3.4 M-Government in Australia

Australia can be regarded as a well-developed country. It is hence unsurprising that the current state of the art of m-government in Australia is comparable to the situation in North America or Europe. The Australian Government offers various mobile apps for smartphones and tablet computers. Beside these apps, other mobile services exist that complement m-government services provided by public authorities. In the following, a brief overview of the Australian m-government landscape is provided by means of a few sample applications.

2.3.4.1 Australian Government Apps

The Australian Government offers citizens several mobile apps. A complete list of all available apps is provided on the government's official website[69]. Currently, more than 70 apps are listed there.

These apps cover different topics and use cases. For instance, the app ACCC Shopper[70] offers relevant consumer information and provides several useful tools to keep copies of receipts and to set reminders. The Australian Pesticides and Veterinary Medicines Authority[71] offers a smartphone app that provides relevant information about agricultural and veterinary chemical products. An app called DisasterWatch[72] supplies citizens with relevant information about disaster-related events and disaster resilience in Australia.

These examples show that apps of the Australian Government cover a broad range of use cases and applications. Still, similar to mobile apps provided by governments of other developed countries, also apps offered by the Australian Government are in most cases restricted to the provision of information,

[66] http://www.veniceforyou.com/high-water-SMS.html
[67] http://www.globe.com.ph/help/gcash/bir
[68] http://www.globe.com.ph/gcash
[69] http://www.australia.gov.au/news-and-media/apps
[70] http://www.accc.gov.au/about-us/tools-resources/accc-shopper-app
[71] http://apvma.gov.au
[72] http://www.em.gov.au/Resources/Pages/DisasterWatchPhoneApp.aspx

useful tools, or simple location-based services. Hardly any app implements a complete transactional service.

2.3.4.2 mHITs

In addition to mobile apps provided by the Australian Government, several other m-government initiatives and solutions have emerged in Australia during the past years. One example is the mobile payment system mHITs[73]. This system works on a prepaid basis and enables users to charge a virtual account e.g. by means of a bank transfer. Charged money can subsequently be transferred to other mHITs users simply by sending predefined SMS messages. In this regard, mHITs resembles to a certain extent the mobile payment solution m-Pesa, which has been deployed in several African countries.

The mobile payment solution mHITs shows that SMS is frequently used for the realization of productive mobile services also in developed countries. Despite the availability of more powerful technologies in these countries, SMS technology is often still used for several reasons including ease of use, platform independence, and compatibility with legacy devices.

2.3.4.3 School News Channel

School News Channel[74] is another example for a popular SMS-based service offered in Australia. This service aims to improve the communication between schools and parents by facilitating information delivery by SMS. The SMS-based information channel is used to send parents and other registered relatives reminders on school events, attendance confirmations, and emergency messages. School News Channel has been developed and is provided by MGM Wireless[75].

2.3.5 M-Government in Europe

In Europe, most countries can be regarded as well-developed. This especially holds true for member states of the EU. Most of these countries have started to invest in e-government infrastructures and solutions early. With the emergence of mobile technologies, several countries have also invested in the development of respective m-government solutions. With regard to m-government, the situation in most European countries is hence comparable to Australia or to developed countries in America or Asia. Most European countries provide citizens several app-based m-government solutions. In addition, various legacy solutions that rely on older technologies such as SMS still exist. A sample of these solutions will be exemplified later in this section.

In one aspect, European countries differ from other developed countries. In Europe, the concepts of electronic identities and electronic signatures are crucial especially for transactional e-government services. Even though their relevance differs from country to country, EU laws such as the Signature Directive [The European Parliament and the Council of the European Union, 1999] or the eIDAS Regulation [The European Parliament and the Council of the European Union, 2014] define a common legal framework that applies to all EU member states. In many European countries, potentials of mobile technologies have early been employed to implement eID and electronic-signature solutions that meet relevant legal requirements. Some of these solutions will also be exemplified in the following, in order to draw a representative picture of the current state of the art of m-government in Europe.

[73] http://www.mhits.com.au/
[74] http://schoolnewschannel.com.au/snc.php
[75] http://www.mgmwireless.com/

2.3.5.1 Mobile-Government Apps

In Europe, several national governments provide their citizens m-government apps for mobile end-user devices. In Austria, the Federal Chancellery offers a set of apps for popular smartphone and tablet-computer platforms[76]. For instance, the app HELP4BABY[77] supplies parents-to-be with relevant information on rights and duties. The app fem:HELP[78] supports women in emergency situations by providing information on available aid organizations. Finally, the mobile solution RIS:App[79] enables citizens to access federal and regional laws by means of a mobile app.

Austria is just one of several European countries providing m-government apps to their citizens. Other European countries doing so are Estonia[80] or Sweden[81]. In the Netherlands, public administrations run an app-based service called BuitenBeter[82]. BuitenBeter is an app-based issue-reporting solution. It enables citizens to take photos of issues in public space and to send these photos with a brief description to the responsible public administration. This way, citizens can conveniently report e.g. overfull garbage cans, broken street lamps, or potholes. Mobile BuitenBeter apps are available for all major mobile platforms. BuitenBeter has been launched 2010 in the city of Eindhoven and has become popular during the past years. Today, it is available in virtually every city of the Netherlands.

While app-based solutions are frequently used in several European countries, governments and public administrations of other EU member states intentionally do not provide mobile apps to their citizens. An example is the United Kingdom (UK), where the November 2012 Digital Strategy says that *'standalone mobile apps will only be considered once the core web service works well on mobile devices, and if specifically agreed with the Cabinet Office'* [UK Cabinet Office, 2012]. Even though a few countries refrain from providing mobile apps to their citizens, a general trend towards app-based m-government solutions can be observed in Europe. Although the number of available m-government apps is steadily increasing, most of the solutions still provide rather simple and in most cases pure informational services. Complete transactional services implemented by mobile apps can hardly be found in European countries so far.

2.3.5.2 Austrian Mobile Phone Signature

In Austria, mobile technologies have been used early to implement eID and electronic-signature solutions. First attempts to provide a country-wide solution for secure user authentication and electronic signatures based on mobile technologies have been made in 2004, when the so-called A1 Signatur[83] has been introduced. In contrast to other mobile eID and electronic-signature solutions available at that time, the A1 Signatur did not create electronic signatures directly on the mobile device. Instead, signatures were computed centrally.

The service A1 Signatur was operated by the Austrian MNO A1[84]. If users wanted to authenticate at a web portal using the A1 Signatur, they had to enter their user name, password, and mobile-phone number first. The authentication data to be signed was then sent to the central server. The server prepared the data to be signed and sent an SMS with a one-time password to the user's mobile phone. The user had to enter the obtained one-time password at the web portal in order to trigger the central creation of the electronic signature. Finally, the signed authentication data was returned to the web portal. Because of poor user acceptance and changed legal circumstances, the A1 Signatur service was stopped in 2007.

[76] http://www.bundeskanzleramt.at/site/6490/default.aspx
[77] http://www.bundeskanzleramt.at/site/cob__52664/currentpage__0/6490/default.aspx
[78] https://www.bka.gv.at/site/cob__52548/currentpage__0/6490/default.aspx
[79] https://www.bka.gv.at/site/cob__52564/currentpage__0/6490/default.aspx
[80] http://estonia.eu/about-estonia/economy-a-it/e-estonia.html
[81] http://www.skatteverket.se/mobil
[82] http://www.buitenbeter.nl/english
[83] https://www.signatur.rtr.at/de/providers/services/mobilkom-a1signatur.html
[84] http://www.a1.net/

In 2010, another mobile eID and electronic-signature solution called Mobile Phone Signature[85] relying on a concept proposed and introduced by Orthacker et al. [2010] has been introduced in Austria. The Mobile Phone Signature again aims to make use of the various benefits provided by mobile technologies and offers Austrian citizens access to services based on qualified electronic signatures. From the user's point of view, the Mobile Phone Signature is similar to the A1 Signatur. Signatures are computed in a central HSM. Hence, the Austrian Mobile Phone Signature follows the architecture of server-based eID and electronic-signature solutions shown in Figure 2.5 on page 31. The user's mobile phone simply acts as end point for a second communication channel that is used to transmit an SMS message containing a one-time password. This secret password can then be used by the receiver to complete the signature-creation process in the central HSM.

During the past years, the Austrian Mobile Phone Signature has gradually replaced smart card based eID and electronic-signature solutions in Austria. Even though both alternatives are usually supported by e-government applications, users clearly prefer the mobile variant to smart card based approaches. This has been revealed by a conducted usability test. We have presented results of this test in Zefferer and Krnjic [2012b].

2.3.5.3 Estonian Mobiil-ID

Mobiil-ID[86] is a SIM-based mobile eID and e-signature solution that is in productive operation in Estonia. It hence follows the architecture of SIM-based eID and electronic-signature solutions shown in Figure 2.4 on page 29. Accordingly, Mobiil-ID makes use of special SIMs to store user-specific data and to create electronic signatures. Together with Estonia's smart card based eID and electronic-signature solution, Mobiil-ID provides required eID and electronic-signature functionality in Estonian e-government procedures.

To authenticate at e-government services or to create an electronic signature, users have to provide their mobile-phone number and a personal code first. The identification and authentication process itself is processed and controlled by a central service that may be used to access the functionality provided by Mobiil-ID through a uniform interface. After receiving an authentication or signature-creation request, the central service transmits relevant data to the user's SIM. Upon reception of these data, the user is requested to enter one of two PINs, depending on the type of transaction, i.e. authentication or signature creation. After provision of the correct PIN, the SIM carries out required operations to either authenticate the user or to electronically sign the received data. The result of the performed operation is returned to the central service, which forwards relevant data to the calling application. As Mobiil-ID uses appropriately certified SIMs and relies on qualified electronic certificates, electronic signatures created with this solution are legally equivalent to handwritten signatures according to EU law.

SIM-based eID and electronic-signature solutions following similar approaches as Mobiil-ID have also been deployed in several other European countries. Examples are Turkey[87], Finland[88], and Norway[89]. In all these countries, SIM-based mobile eID and electronic-signature solutions enable citizens to authenticate at e-government services and to create legally binding electronic signatures using their mobile phones. We have provided a detailed overview and analysis of mobile eID and electronic-signature solutions in Zefferer and Teufl [in press].

[85] http://www.handy-signatur.at/

[86] http://www.id.ee/index.php?id=36881

[87] http://www.turkcelltech.com/Product.aspx?Id=5eba6d36-dba7-42be-a499-cea1422d0a3d&PId=15c36401-ddf8-424f-8489-41077a2974ab

[88] http://www.mobiilivarmenne.fi/fi/

[89] https://www.bankid.no/

2.3.5.4 EmergencySMS

The EmergencySMS service[90] is provided in the UK and is supported by the government, various mobile network operators, emergency services, Ofcom[91], and Action on Hearing Loss[92]. The service enables deaf, hard of hearing, and speech-impaired people in the UK to communicate with emergency services such as the police, ambulance, fire rescue, and coast guard by means of SMS.

EmergencySMS shows that even though SMS is a rather simple and old technology, it can still be useful to improve the accessibility of certain services. In the concrete case of EmergencySMS, it supports hearing-impaired persons by providing an efficient alternative to speech-based communication.

2.3.5.5 Bustxt

Bustxt[93] is an SMS-based timetable service offered in the city of Dublin, Ireland. It enables passengers to request timetable information for bus routes by SMS. Bustxt is provided by the Irish transport company Dublin Bus[94].

In most developed countries, transport companies provide app-based timetable and journey-planning solutions. Examples are solutions provided by the public transport companies of the cities of Munich[95], Vienna[96], or Graz[97]. The Irish Bustxt service is still an interesting alternative, as it shows that useful services can also be implemented with simple and less powerful mobile technologies. The main advantage of Bustxt compared to app-based solutions is its general applicability on all mobile phones including those without Internet access.

2.3.5.6 Ask Brook

Ask Brook[98] is a British service that offers information and advice for young people. Ask Brook is a service provided by the charity organization Brook[99]. According to their website, Brook has the objective to promote the health of young people and those most vulnerable to sexual ill health through the provision of relevant information, education and counseling, confidential clinical and medical services, professional advice, and training. Ask Brook gives young people the opportunity to make use of counseling services using SMS messages. This mobile communication technology enables those seeking for advice to avoid personal contact and to remain anonymous to a certain extent.

2.3.5.7 Parent-Notification Systems

TextaParent[100] is an Irish service that facilitates the communication between schools and parents using SMS technology. Schools can use the service to inform parents of e.g. last minute timetable changes, emergency school closures, or important reminders. TextaParent is hence basically comparable to the Australian service School News Channel.

Another parent-notification system similar to TextaParent is called CallParents[101]. CallParents is

[90] http://www.emergencysms.org.uk/index.php
[91] http://www.ofcom.org.uk/
[92] http://www.actiononhearingloss.org.uk/
[93] http://www.dublinbus.ie/your-journey1/bustxt/
[94] http://www.dublinbus.ie
[95] http://www.mvg-mobil.de/fahrinfo/
[96] http://www.wienerlinien.at/qando
[97] http://www.verbundlinie.at/fahrplan/bedienung_mobile.php
[98] http://www.askbrook.org.uk
[99] http://www.brook.org.uk/about-brook/category/what-we-do
[100] https://textaparent.ie/
[101] http://www.the-contactgroup.com/products/call-parents/

mainly used in the UK and aims to facilitate efficient communication between teachers, parents, and students. Similar to TextaParent, CallParents relies on SMS technology for message delivery.

Parent-notification systems such as TextaParent or CallParents are further prime examples showing that SMS technology is still frequently used in Europe. Even though SMS is a rather old technology, it is still sufficient for various use cases to implement simple but efficient services.

2.3.6 Trends in Mobile Government

The provided overview of m-government solutions covers just a subset of all solutions currently available. Due to the increasing use of mobile information and communication technologies, the number of available m-government services is continuously growing. A complete list of all available service would hence go beyond the scope of this thesis. Still, sketched solutions have been selected such that a representative picture of m-government in different regions of the world is drawn. For this purpose, focus has not only been put on m-government services provided by public administrations, but also on services from related fields such as m-health or m-education that are often provided by private organizations or NGOs. This way, a comprehensive overview of mobile technologies in mobile government related fields of application has been provided. From this overview, several findings can be derived and current trends in mobile government can be identified. Derived findings and identified trends are discussed in the following subsections.

2.3.6.1 Mobile Government to Fight the Digital Divide

The brief survey on existing m-government solutions shows that mobile government is a global phenomenon. Public-sector solutions that employ mobile technologies can be found on all continents. However, there are differences between m-government solutions provided in developed and developing countries. Underdeveloped countries e.g. in Africa and Asia often suffer from missing wire-based communication infrastructures especially in rural areas. There, mobile technologies are hence often the only opportunity for communication. Accordingly, mobile phones are often the only way for people to communicate with each other over longer distances and the most efficient opportunity for governments to get in contact with their citizens.

The special circumstances in developing countries are also reflected by the nature of provided services from the m-government domain. The survey has shown that in Africa and Asia, mobile technologies are frequently used for health care and educational purposes. As mobile broadband networks are often not available yet, provided services are usually restricted to voice and SMS transmissions. Despite limited transmission capacities, various m-health and m-education services have shown to be an appropriate tool to facilitate the lives of people in developing countries and to fight the digital divide.

2.3.6.2 Mobile Government for Always-On Societies

Missing wire-based communication infrastructures make mobile networks often the one and only alternative for long-distance communication in developing countries. In developed countries, the situation is completely different. There, multiple communication opportunities are usually available at the same time, especially in urban areas. Still, m-government is an important topic in these countries as well. However, the motivation to provide and use m-government services is completely different compared to developing regions. The probably most important reason to rely on m-government in developed countries is convenience. In the typical western always-on society, there is a growing demand for services that are accessible anytime and everywhere. Mobile devices such as smartphones are usually always carried and always on. Hence, they are perfectly suitable to act as enabler for services that shall be available and accessible independently of the current time or the user's current location and context.

The conducted survey has shown that the aim for convenience is also reflected by the nature of provided m-government services, which differ clearly from services provided in developing regions. In developed countries, provided m-government services usually aim to provide citizens nonstop access to services. SMS is still frequently used for this purpose. However, in contrast to developing countries a clear trend towards mobile app based solutions can also be observed. Furthermore, especially European countries rely on mobile technologies to implement eID and electronic-signature solutions, which enable transactional e-government services.

2.3.6.3 SMS as Technology of Choice

The conducted survey has revealed that the rather old SMS technology is still frequently used for m-government services throughout the world. This is unsurprising for developing countries, where mobile broadband networks are often not available and SMS represents one of few technical opportunities to transmit data. Interestingly, SMS is also still frequently used in developed countries. Examples are parent-notification solutions that facilitate communication between schools, parents, and students. Another popular example are mobile eID and electronic-signature solutions such as the Estonian Mobiil-ID or the Austrian Mobile Phone Signature, which make use of SMS technology either to transfer data between the MNO and the user's SIM, or to transmit one-time passwords during user authentication.

Indeed, there are several arguments that justify reliance on the rather old-fashioned SMS technology. First, SMS is an approved and well-established technology. Even technically inexperienced users know how to read and write SMS messages. Second, SMS does not make high demands on the mobile end-user device and is supported by all current devices. Third, there are no compatibility issues with SMS messages. While mobile apps usually need to be tailored to a special mobile platform and operating system, SMS is supported by all current platforms out of the box. All these aspects contribute to the continuing success and popularity of SMS technology and explain the frequent use of this technology in m-government services all over the world.

2.3.6.4 Mobile Apps on the Rise

Despite the continuing success of SMS technology, the recent emergence of smartphones especially in developed countries has brought a new take on m-government solutions. In various countries, governments and public administrations already provide mobile apps for smartphones and tablet computers. The conducted survey has revealed that m-government portals, on which such apps are offered, are for instance available in the US, Canada, Austria, or Estonia. For the future, it can be expected that with an increasing smartphone penetration and with the further development of mobile communication networks also the number of provided m-government apps will further increase. While simple SMS-based services will still be the most appropriate choice for certain use cases, existing mobile apps already show their potential to open up new application scenarios.

2.3.6.5 Lack of Transactional Services

Solutions from the field of e-government can be classified into informational, responsive, and transactional services. Only the latter can be regarded as pendant to classical administrative procedures, i.e. include the three phases application, back-office processing, and delivery. The conducted survey has revealed that there are hardly any m-government solutions implementing or providing full transactional services so far.

Transactional e-government services typically require a reliable user authentication and require the user to provide written consent by means of an electronic signature. Accordingly, eIDs and electronic signatures have been identified as basic concepts of transactional e-government services. Interestingly, the conducted survey shows that several solutions exist that incorporate mobile technologies to implement

eID and electronic-signature functionality. Examples are the Austrian Mobile Phone Signature or the Estonian Mobiil-ID. However, these solutions have been designed and developed in the pre-smartphone era and are hence tailored to a use on classical end-user devices. Concretely, these solutions and their underlying security concepts assume that the mobile device is used as additional device only. This explains, why existing mobile eID and electronic-signature solutions are hardly used by mobile apps to implement transactional services.

In the absence of fully transactional m-government services, most surveyed solutions must be classified as purely informational. Other surveyed solutions provide simple tools or location-based services, but do not implement transactional services either. Hence, they do not integrate eID or electronic-signature functionality. This applies to both simple SMS-based services as well as to m-government solutions based on mobile apps.

2.4 Chapter Conclusions

The provided overview of the current state of the art of m-government has revealed several interesting findings. First of all, it has shown that m-government in developing countries must be dissociated from m-government in developed countries. In both developing and developed regions, mobile public-sector services are referred to as m-government. However, drivers behind these services, use cases, and circumstances differ significantly depending on the development stage of the region, in which these services are deployed. Accordingly, also the provided services themselves differ significantly. The provided overview has also shown that differences among developing countries or among developed countries are not that considerable. For instance, in most developing countries, the health-care sector is very active in providing mobile services to improve the health-care situation in underdeveloped regions. Similarly, most developed countries provide citizens app-based solutions to supply them with relevant information in a convenient and user-friendly way.

In general, it is unsurprising that developed countries increasingly make use of mobile apps for the provision of m-government services. Compared to legacy technologies such as SMS, mobile apps allow for more powerful services. Another argument for an increased use of mobile apps is the steadily growing smartphone penetration in developed countries. In consideration of the continuing popularity of mobile apps for smartphones and tablet computers, it is at a first glance quite astonishing that there are currently hardly any transactional m-government procedures available that include a reliable user authentication and a provision of written consent by the user. This is interesting, as mobile technologies have been employed by eID and electronic-signature solutions for many years. However, all these solutions have originally been designed for classical end-user devices and use the mobile phone as second end-user device only. Hence, they can in most cases not be ported to and used on mobile end-user devices. For this reason, existing eID and electronic-signature solutions are usually not applicable on mobile end-user devices, which in turn renders transactional m-government services infeasible.

The current lack of transactional m-government services is a significant drawback. Existing app-based m-government solutions show that the incorporation of mobile cutting-edge technologies and modern mobile end-user devices enables various new application opportunities and can significantly ease the life of both citizens and public administrations. Employing this potential for rather simple informational m-government services only must hence be regarded as a waste. The required transition from informational m-government to transactional m-government necessitates the availability of eID and electronic-signature solutions that can be applied and used on mobile end-user devices. Development of such solutions that pave the way for transactional m-government can hence be defined as the basic goal of this thesis.

Chapter 3

Security on Mobile End-User Devices

" There is no such thing as perfect security, only varying levels of insecurity."

[Salman Rushdie, British Indian Novelist and Essayist.]

Even though the above statement by Salman Rushdie is not necessarily restricted to IT solutions, it recalls that absolute security can hardly be achieved in practice. This fact must not be neglected during design and implementation of IT solutions that store, process, and transmit security-critical data. To a certain extent, the security of these solutions always depends on capabilities and limitations of their underlying technologies. Understanding the characteristics of employed technologies is hence crucial for the development of solutions that need to meet security requirements. In particular, this also applies to m-government solutions that typically rely on various mobile technologies to implement required functionality. Understanding capabilities and limitations of mobile technologies is crucial for the development of secure m-government solutions.

The current m-government landscape shows that there is a clear trend towards m-government services based on mobile apps. App-based m-government solutions benefit from the various cutting-edge technologies that are integrated into current smartphones and tablet computers. Examples are location-based services such as the Alternative Fueling Station Locator[1] provided by the US Department of Energy[2], or the Dutch app BuitenBeter[3], which integrates photography features to enable citizens to conveniently report issues in public space. Furthermore, app-based m-government solutions benefit from the usability of current mobile end-user devices. As usability has been identified as one of the five pillars of successful m-government, smartphones or tablet computers seem to be the most suitable end-user devices for successful m-government solutions.

Due to their various advantages, it can be expected that app-based m-government solutions will continue to gain relevance in future. The validity of this expectation is underpinned by the fact that several SMS-based m-government services are gradually complemented by app-based solutions. An example is the m-government service Text4Baby[4], which has started as a pure SMS-based service and can now also be accessed by means of an iPhone app[5]. So far, the trend towards app-based m-government mainly applies to well-developed countries, where nationwide mobile broadband networks are available and where the market penetration of smartphones is already high. However, it can be expected that app-based solutions will also gain popularity in developing countries, once required mobile broadband communication infrastructures and cheap smartphones are available.

[1] https://itunes.apple.com/us/app/alternative-fueling-station/id718577947
[2] http://www.energy.gov/
[3] http://www.buitenbeter.nl/english
[4] https://www.text4baby.org/
[5] https://itunes.apple.com/us/app/text4baby/id894749779?mt=8

While smartphones and tablet computers offer various new opportunities, they also raise several security challenges that need to be considered. In contrast to classical mobile phones, smartphones and tablet computers are typically connected to the Internet and support the installation of third-party software. In this aspect, smartphones and tablet computers rather resemble desktop PCs and laptops than classical mobile phones. This makes modern mobile end-user devices also prone to attacks and malware. Especially on Google Android, malware has evolved to a severe problem during the past years [Kelly, 2014]. Even though the situation is usually less critical on other platforms, modern mobile end-user devices must in general not be assumed to be secure. This is problematic, as security has been identified as another crucial pillar of successful m-government. Development of secure m-government solutions therefore requires a deep understanding of the mobile platforms, which these solutions are intended for. Only if all opportunities and limitations of these platforms are known, m-government solutions can be tailored to the platforms' special characteristics and a sufficient level of security can be assured.

Providing an appropriate level of security and hence achieving a deep understanding of current mobile platforms is especially important for the development of mobile eID and electronic-signature solutions. These solutions must be regarded as critical, as successful attacks on these services potentially lead to impersonation and enable attackers to create legally binding electronic signatures on behalf of the legitimate user. Development of a sufficiently secure mobile eID and electronic-signature solution is the fundamental goal of this thesis. As a preparation for that, relevant aspects of modern mobile end-user devices and underlying operating systems are analyzed in this chapter. This way, their current state of the art is examined and a deep understanding of opportunities and limitations is achieved. All analyses presented in this chapter are based on own published research. In particular, security features and limitations of current popular mobile platform are investigated by focusing on m-government use cases and their requirements. Furthermore, different approaches to improve the provided level of security of mobile end-user devices are discussed. Finally, first attempts to implement electronic-signature functionality on mobile end-user devices are presented to show the basic feasibility of this approach.

3.1 Security Features and Limitations of Mobile Platforms

Security is one of the five fundamental pillars of successful m-government. This raises challenges for app-based m-government services that are tailored to a use on smartphones or tablet computers, as these devices are potentially prone to attacks and malware. Raised security challenges can be met by making use of security features provided by mobile platforms. Unfortunately, these platforms differ significantly in terms of their vulnerability against attacks and also in terms of provided security features. This complicates the development of secure m-government solutions, as a deep understanding of potential vulnerabilities and of provided security features is necessary, and because several platform specifics need to be taken into account.

To overcome this problem, we provide a systematic analysis and assessment of the three mobile platforms Google Android, Apple iOS, and Microsoft Windows Phone 8. This analysis is based on own scientific work, in which we have assessed the suitability of the three mentioned mobile platforms for m-government [Zefferer et al., 2013b]. While the general analysis of current mobile platforms has been a joint activity with colleagues, the author of this thesis has finally applied the mapping of analyzed platform capabilities and limitations to concrete m-government-related use cases. In this section, main findings of this publication are recapped. For each of the three considered platforms, available security features are analyzed that can be employed by m-government solutions to achieve the required level of security. To follow a systematic approach, relevant use cases are identified first. From these use cases, four research questions are derived. Subsequently, relevant security features are identified. The three investigated platforms are then analyzed by means of these security features. Finally, results of this analysis are used to assess the three platforms according to the previously defined research questions. This way, capabilities and limitations of the three platforms Google Android, Apple iOS, and Microsoft

Windows Phone 8 to provide m-government solutions an appropriate level of security are systematically assessed.

3.1.1 Relevant Use Cases

As a first step towards a systematic analysis and assessment of different mobile platforms, relevant m-government use cases are identified. In general, the rather broad field of m-government provides various opportunities to integrate and to make use of smartphones and related mobile end-user devices. For the scope of this analysis, focus is mainly put on the following two use cases:

- **Internal usage:** Providing employees of public administrations with powerful mobile end-user devices can speed up internal processes and increase efficiency. Mobile end-user devices enable access to required information and data everywhere and at any time. For instance, the use of smartphones enables access to e-mails around the clock and independent of the current location. During the past few years, different strategies to provide employees with mobile end-user devices have emerged. These strategies are applied in both the public and the private sector. Overall, three strategies can be distinguished. Pursuing the first strategy, employers provide their employees with mobile end-user devices. These devices are owned and managed by the employer and may be used for business matters only. The second strategy is similar to the first one, but allows employees greater latitude regarding the use of provided devices. In particular, employees are allowed to use these devices also for private affairs. Accordingly, this second strategy is referred to as Corporate Owned Personally Enabled (COPE). The third strategy that has gained popularity during the past years is called Bring Your Own Device (BYOD). Pursuing this strategy, employees are allowed to connect their private mobile end-user devices to the corporate infrastructure. On the one hand, this is beneficial for employers, as they can save investment costs. On the other hand, this strategy potentially raises several security issues, as the employer's control over used devices is limited. Concretely, employees can hardly be prevented from using insecure and potentially malware-infected devices. Despite its disadvantages, BYOD is still continuously gaining popularity. Irrespective of the pursued strategy, the use of mobile end-user devices by employees of public administrations is a relevant use case and needs to be considered in detail.

- **Citizen applications:** This use case covers scenarios, in which citizens use their private mobile end-user devices to access services and applications provided by public administrations. This use case is gaining relevance, as mobile end-user devices are gradually replacing classical ones. Security is a relevant aspect for this use case, as service providers have no control over used client devices and hence must not presume a certain level of provided security. This situation is even complicated by the fact that potential security threats and provided security functions differ significantly between mobile platforms.

From the two identified use cases, a set of research questions can be derived. By finding answers to these questions, a deep understanding of differences between current mobile platforms regarding relevant security aspects can be acquired. Concretely, the following research questions can be derived:

- **RQ-1:** Which mobile platforms shall be chosen when providing employees of public administrations with mobile end-user devices?

- **RQ-2:** Which mobile platforms shall be supported by public administrations, if a BYOD strategy is pursued?

- **RQ-3:** For which mobile platforms shall security-critical citizen applications be provided by public administrations?

- **RQ-4:** For which mobile platforms shall non-critical citizen applications be provided by public administrations?

3.1.2 Relevant Security Features

To enable a systematic comparison of different mobile platforms and a derivation of answers to the derived research questions, relevant platform properties need to be identified first. These properties can be derived from assets deserving protection and from threats that apply to these assets. Considering the two identified relevant m-government use cases, data being stored and processed on mobile end-user devices can be identified as basic asset. The capability to protect this asset from attackers is hence the main quality measure for mobile platforms. In addition to complex and hence expensive approaches such as side-channel analysis, attackers typically pursue two promising strategies to compromise security-critical data stored and processed on mobile end-user devices. First, attackers steal mobile devices in order to gain access to stored data. Due to their mobile nature, mobile end-user devices are more prone to theft and loss compared to stationary devices such as desktop PCs. Second, attackers use malware to compromise data on mobile devices. As modern mobile platforms enable users to dynamically extend the functionality of their devices with third-party software, malware is an omnipresent threat. In addition, frequent disclosures of powerful exploits increases the threat potential of malware as well. From the basic asset data and from the two identified threats theft and malware, relevant platform properties can be derived that have an impact on the platform's capabilities to protect identified assets and to counter imminent threats.

Relevant platform properties can be subdivided into two categories. These categories correspond to the two identified threats, i.e. theft and malware. Accordingly, the two categories data protection and malware resistance have been defined. Each of these categories contains a set of relevant platform properties. Regarding the category data protection, the following properties can be identified as crucial:

- **Access-protection mechanisms:** Access protection is the first line of defense in scenarios, in which the mobile end-user device gets stolen. Activated access-protection mechanisms prevent the thief from gaining access to the device through its user interface. Common implementations of these mechanisms are password-based or fingerprint-based user-authentication schemes. The quality of provided access-protection mechanisms is a key property to counter the threat theft.

- **Encryption support:** Although appropriate access-protection mechanisms prevent attackers from accessing the graphical user interface of mobile end-user devices, they cannot prevent direct access to locally stored data. To guarantee the confidentiality of these data, encryption must be applied. The support for data-encryption mechanisms is hence another crucial platform property to counter the threat theft.

- **Support for secure storage of credentials:** Credentials such as PINs, passwords, or cryptographic keys are highly-critical data, for which confidentiality must be guaranteed when being stored on the mobile device. Therefore, the provision of dedicated credential stores that integrate reliable protection mechanisms represents an important platform property.

- **Mobile Device Management (MDM) capabilities:** MDM is a platform feature that is especially relevant for use cases, in which mobile devices are provided to employees for business affairs. MDM enables the owner of the mobile device, i.e. the employer, to retain control over issued devices. Concretely, MDM assures that certain device configurations can be determined before the device is handed over to the employee. For instance, MDM can be used to enforce the use of access-protection mechanisms or encryption. In most cases, these configurations can also be dynamically adapted, when the device is already in the field. Most importantly, the user can usually not disable or modify configurations defined by MDM. This way, MDM enables device owners

to retain control over their own devices and to prevent weak and potentially insecure configurations. The support for MDM is hence an important platform property that is especially relevant for m-government use cases, in which public administrations provide their citizens with mobile end-user devices.

In addition to these platform properties, which are important to defeat threats imposed by the theft of a mobile device, the following platform properties can be identified as crucial for the protection against malware:

- **Provided Application Programming Interfaces (APIs) and Inter-Process Communication (IPC) capabilities:** Malware residing on a mobile end-user device has basically the same capabilities to access APIs and capabilities for IPC as any other third-party app. The set of provided APIs and IPC capabilities hence influences the potential power of malware. If a platform provides third-party apps more capabilities to access system resources and functionalities, also malware on this platform can employ these capabilities and must hence be regarded as more powerful. The set of provided APIs and IPC capabilities is therefore an important property that influences the vulnerability of a mobile platform to malware.
- **Resistance against rooting:** As capabilities of malware are technically limited to those of ordinary third-party apps, attackers frequently try to exploit known security flaws, in order to gain root access to the mobile operating system. Once attackers or malware have gained root access to a targeted device, they are theoretically able to circumvent all enforced security mechanisms. Thus, resistance against rooting is an important property of mobile platforms that influences its resistance against powerful malware.
- **Integrated security features:** Mobile platforms typically integrate various security features ranging from restrictions of potential sources for third-party apps, over security measures on operating-system level, to permission systems that control capabilities and access rights of installed apps. Their availability and reliability contribute to the overall security of a mobile end-user device. Hence, the set of integrated security features is a crucial property of mobile platforms.
- **Availability of updates:** The past has shown that it is virtually impossible to develop and provide an error-free operating system. Hence, timely and reliable update mechanisms are crucial, as they represent the only opportunity to correct discovered errors and security flaws on mobile end-user devices in the field and to keep operating systems up to date. The availability of suitable update mechanisms and the provision of timely updates are hence relevant properties of mobile platforms and influence the platforms' malware resistance.

3.1.3 Platform Analysis

In this section, the three mobile platforms Google Android, Apple iOS, and Microsoft Windows Phone 8 are analyzed according to the platform properties that have been identified as relevant. This way, security features and capabilities of these three platforms are examined and possible limitations are revealed. The conducted analysis enables a subsequent assessment of the three platforms according to the defined research questions.

3.1.3.1 Analysis of Google Android

During the past years, Google Android has evolved to the most popular mobile platform and currently holds the largest market share of all mobile platforms [IDC, 2014]. From a developer's perspective, the key advantage of Android is its openness. For instance, Android provides a rich set of APIs for third-party apps that enable access to various system resources and functions. This openness is advantageous

in terms of functionality, but it also enables powerful attacks. This becomes apparent when analyzing the identified relevant platform properties for the special case of Android.

- **Access-protection mechanisms:** Reliable access-protection mechanisms have been identified as relevant property of mobile platforms, which influences their resistance against threats imposed by theft of the mobile device. Android provides a broad spectrum of access-protection mechanisms. They range from password-based approaches over biometric solutions to mechanisms that require the user to enter a graphical pattern to unlock the device. All methods are disabled by default. It is up to the user to choose, enable, and configure the preferred method. Alternatively, access-protection mechanisms can also be enabled and configured by an MDM solution in place.

- **Encryption support:** File-system encryption is available for Android since version 3.0. However, this feature is disabled by default and needs to be activated. Android's encryption system applies to the entire internal file system. A selective encryption of single files or directories is not supported. The key-derivation process used to derive the encryption key does not include a device-specific key. Instead, the encryption key is solely derived from a passcode entered by the user. Thus, brute-force attacks on the key are not bound to the mobile device but can also be outsourced to powerful external resources. This way, attacks on the user's passcode and hence on the encryption key can be significantly sped up. We have provided a more detailed review and analysis of Android's encryption system in Teufl et al. [2014a].

- **Support for secure storage of credentials:** In addition to file-system encryption, Android additionally provides third-party apps an especially protected credential store called Android KeyChain[6]. The Android KeyChain stores credentials in encrypted form. Encryption is based on the Advanced Encryption Standard (AES) [NIST, 2001] and on an encryption key derived from the user's access-protection passcode. Activation of a passcode-based access-protection mechanism is hence a prerequisite for the Android KeyChain. The key-derivation function used to derive the AES key from the user's passcode again does not include a device-specific hardware key. Hence, brute-force attacks on the key can again be sped up by outsourcing resource-intensive computations to external resources.

- **MDM capabilities:** Android supports MDM. However, provided capabilities are very limited, as only few configuration options can be defined by MDM. Several vendors of mobile end-user devices address this issue by adding vendor-specific proprietary MDM capabilities. This has resulted in a heterogeneous ecosystem of MDM solutions for Android, which complicates the deployment of MDM infrastructures and the support for multiple Android-based devices in practice. Another issue is the lack of integrated MDM clients under Android. Such clients are necessary to enforce centrally defined policies on the mobile device. Android requires the installation of a separate app that assumes the role of the MDM client. However, this app is subject to the same security flaws as any other third-party app. Hence, the lack of an MDM client that is deeply integrated into the mobile operating system is another shortcoming of Android's MDM support.

- **Provided APIs and IPC capabilities:** Another drawback of Android with regard to security is the platform's comprehensive API that can be used by third-party apps to access system functionality. While third-party apps benefit from this comprehensive API, also malware can employ provided access to system functionality. In contrast to most other platforms, Android additionally provides rather broad support for inter-process and inter-application communication. For this purpose, Android has introduced the concept of Intents[7]. An Intent is a data object that can be used to exchange data between applications and processes. If misused, Intents can be a potential data leak, which

[6] http://developer.android.com/reference/android/security/KeyChain.html
[7] http://developer.android.com/reference/android/content/Intent.html

can be exploited by malware. In summary, Android's broad support for powerful APIs and IPC enables powerful third-party apps but also raises several security risks.

- **Resistance against rooting:** To limit the threat potential of malware, resistance against rooting has been identified as crucial property. For the special case of Android, its resistance against rooting must be rated as rather poor. In fact, the rooting of Android devices is currently common practice and is facilitated by several freely available tools. In many cases, Android devices are intentionally rooted by device owners in order to gain access to additional functionality. However, known security flaws can also be exploited by malware to gain full access to the targeted device's operating system. In Android version 5, several modifications have been applied to the operating system that complicate a successful rooting process. However, the majority of the Android devices currently in the field are still prone to rooting, as they rely on older Android versions.

- **Integrated security features:** So far, a quite sobering picture of Android's platform properties related to security has been drawn. Nevertheless, Android also provides and implements several security features, which assure a certain level of security for apps running on the mobile device. For instance, Android implements a sandboxing mechanism to separate and isolate different apps from each other. This mechanism assures that each app can only access its own data and that this data is protected from access by other applications. Furthermore, Android implements a complex permission system, which restricts access to resources and features of the mobile device. Apps that want to access protected resources or functions need to request the corresponding permission. This permission must be granted by the user during the app-installation process. This way, access rights of installed apps remain under control of the user. While this works fine in theory, users are in practice often not aware of implications of granted permissions. Furthermore, technically inexperienced users often do not completely understand this security feature. Practical limitations of Android's permission system have been discussed in detail by Felt et al. [2012].

- **Availability of updates:** Finally, Android also faces several problems regarding the availability of timely updates. This is mainly due to the fragmentation, which Android is suffering from. Several vendors equip their Android-based mobile devices with modified versions of the operating system. Examples are the mobile-device vendors HTC[8] or Samsung[9], which implement proprietary user interfaces on top of the original Android system. Applied modifications have to be re-implemented and integrated for each new release of Android. This takes time and often delays the release of important updates. Furthermore, the provision of updates causes costs for device vendors but does not directly produce profit. For this reason, updates are often provided on an irregular basis only.

In summary, it must be concluded that the mobile platform Google Android suffers from several security-related drawbacks. The vulnerability to rooting, the poor availability of timely updates, and the provided API-based access to system functionality are among the most problematic properties of this platform. Android hence appears to be only partly suitable for security-critical m-government use cases.

3.1.3.2 Analysis of Apple iOS

Apple iOS differs significantly from Google Android in several aspects. Overall, iOS provides third-party apps less opportunities to access system functionality and to make use of inter-application or inter-process communication. While this limits features of third-party apps, it leads to a higher overall level of security. This also becomes apparent when analyzing identified relevant platform properties for the special case of iOS.

[8] http://www.htc.com
[9] http://www.samsung.com

- **Access-protection mechanisms:** Similar to Android, iOS provides different access-protection mechanisms. This includes locking mechanisms based on numeric or alphanumeric passcodes, as well as a biometric solution based on the user's fingerprint. As for Android, access-protection mechanisms need to be manually selected and activated by the user or by an MDM solution in place. By default, no access protection is enabled.

- **Encryption support:** In contrast to Android, iOS provides two separate encryption systems. A file system based encryption system can be used to encrypt the entire file system. This system is similar to the one provided by Android. In addition, iOS also features a file-based encryption system that can be used by mobile apps to encrypt single files. For each file, the app can chose one of several protection classes. A protection class defines the encryption key to be used, the applied cryptographic method, and the key-derivation function that is used to derive the required cryptographic keys. It is hence the responsibility of the app developer to select an appropriate protection class for the files to be encrypted. Depending on the chosen protection class, iOS integrates device-specific keys into the key-derivation function. This renders the outsourcing of brute-force attacks to external resources infeasible, as required key material is available on the mobile device only. In summary, the two encryption systems provided by iOS can be regarded as appropriate as long as they are correctly used, i.e. suitable protection classes are chosen by responsible app developers. We have reviewed and discussed encryption systems provided by iOS in more detail in Teufl et al. [2013b].

- **Support for secure storage of credentials:** In addition to the two encryption systems, iOS also features a secure credential storage. Similar to Android, this credential storage is called iOS KeyChain. The KeyChain can be used to securely store cryptographic keys, passwords, and other credentials. As for the file-encryption systems, the security of credentials stored in the KeyChain again depends on its correct use.

- **MDM capabilities:** For its use in managed environments, iOS provides broad support for MDM. In contrast to Android, iOS features an MDM client, which is deeply integrated into the mobile operating system. Furthermore, iOS supports the remote configuration of various system properties. This enables corporate device owners to adapt mobile devices issued to employees to different use cases and related requirements.

- **Provided APIs and IPC capabilities:** Another aspect, in which iOS differs significantly from Android, is the set of provided APIs and supported capabilities for inter-application and inter-process communication. Apple's mobile operating system provides third-party apps a reduced API only. For instance, it does not enable third-party apps to access SMS functionality. Similar restrictions apply to capabilities provided for IPC. While this reduces the capabilities of third-party apps, it also limits the threat potential of malware.

- **Resistance against rooting:** Similar to Android, also iOS is in principle prone to rooting or jailbreaking. Jailbreaking of iOS devices has become common practice during the past years as it enables installation of more powerful applications. There are several tools publicly available that facilitate the application of jailbreaks and make rooting possible even for technically inexperienced users.

- **Integrated security features:** Similar to Android, iOS implements a sandboxing mechanism to separate installed apps and their data from each other. Capabilities and rights of single apps are managed by a permission system. In contrast to Android, required permissions must not be granted by the user during the app-installation process, but during runtime. Download and installation of apps are possible through the official Apple App Store[10] only. Apps provided in this App Store

[10] https://itunes.apple.com/

are subject to security checks and quality-assurance mechanisms. This increases security, as it complicates the distribution of app-based malware.

- **Availability of updates:** Relevant updates are frequently provided for iOS-based devices. This is in stark contrast to Android, where customization and fragmentation of the mobile operating system often complicate the provision of timely updates. In this regard, iOS is advantageous, as both hardware and operating system are under control of one single vendor. The rather low number of different iOS-based mobile end-user devices facilitates an efficient provision of relevant updates.

In summary, the mobile platform Apple iOS is advantageous compared to Google Android in terms of security. This is mainly due to the fact that iOS provides third-party applications only limited capabilities to access system functionality. While this reduces the power of third-party apps, it also limits the threat potential of malware. The broad support of MDM and the availability of timely updates also contribute to an adequate level of security provided by Apple iOS.

3.1.3.3 Analysis of Microsoft Windows Phone

In 2012, Microsoft has launched its mobile platform and operating system Windows Phone 8. Although being introduced as the official successor of Windows Phone 7, Windows Phone 8 represents a completely new platform that bases to a certain extent on the Microsoft Windows 8 operating system for classical end-user devices. Hence, despite their similar names, there is almost no compatibility between Windows Phone 7 and Windows Phone 8. As Microsoft puts focus clearly on Windows Phone 8, this platform is the basis of the analysis conducted in this section. The analysis itself is mainly based on information published by Microsoft [Microsoft Corporation, 2012].

In many aspects, the mobile platform Microsoft Windows Phone 8 is comparable to Apple iOS. It provides third-party applications only very limited access to system functionality and hence reduces potential attack vectors. At the same time, this also reduces capabilities of third-party apps. This also becomes apparent when having a more detailed look on identified relevant platform properties for the special case of Microsoft Windows Phone 8.

- **Access-protection mechanisms:** Similar to Google Android and Apple iOS, Microsoft Windows Phone 8 supports a PIN-based access protection mechanism. In contrast to the other two analyzed platforms, Windows Phone 8 is however less flexible regarding the support of alternative access-protection mechanisms.

- **Encryption support:** Windows Phone 8 supports file-system encryption based on Microsoft's BitLocker technology. Required encryption keys are stored in a Trusted Platform Module (TPM). This assures that only trusted components are capable to apply decryption operations. File-system encryption is mainly intended for business scenarios. Accordingly, it can only be activated using appropriate MDM policies.

- **Support for secure storage of credentials:** Windows Phone 8 provides a data-protection API that enables the secure storage of credentials in a so-called isolated storage. Stored data is encrypted with a key that is unique for each application. It is created upon first start of the application. The key-generation function takes as input the user's personal credentials and a unique application identifier. The key itself is derived with the help of the device's TPM.

- **MDM capabilities:** Similar to iOS, Windows Phone 8 supports MDM. Required MDM functionality is fully integrated in the operating system. Installation of an additional MDM client is not necessary.

- **Provided APIs and IPC capabilities:** In contrast to Android, Windows Phone 8 provides a restricted API to third-party applications only. In this regard, Windows Phone 8 is hence comparable to iOS. Furthermore, Windows Phone 8 does not provide broad support for background tasks.
- **Resistance against rooting:** Among the three analyzed platforms, Windows Phone 8 provides the best protection against rooting or jailbreaking. Hardly any techniques are known so far that allow for root access to the operating system. One reason for that is the fact that Windows Phone 8 includes a secure booting mechanism based on the Unified Extensible Firmware Interface (UEFI). Thus, during the boot sequence, the integrity of each component of the operating system is cryptographically checked. So far, this has proven to be an efficient means to prevent rooting.
- **Integrated security features:** Similar to Android and iOS, also Windows Phone 8 follows a sandboxing approach to separate apps and their resources from each other. In addition, Windows Phone 8 also implements a permission system that restricts capabilities of apps to access certain functions and features. Another security feature of Windows Phone 8 is its restrictive behavior towards third-party apps. Similar to iOS, these apps can only be provided through the official Windows Phone Store[11]. There, third-party apps are subject to a strict review process. This way, the distribution of malware is impeded.
- **Availability of updates:** With regard to fragmentation, Windows Phone 8 can be seen as a combination of Android and iOS. Similar to Android, Windows Phone 8 is deployed on mobile end-user devices of different vendors. Hence, Microsoft has no full control over the hardware, on which the mobile operating system is running. However, in contrast to Android, hardware vendors must not modify or customize the original operating system. This way, Microsoft retains full control over the update process and no delays are induced due to required modifications of the operating system.

From a security perspective, the mobile platform Microsoft Windows Phone 8 is comparable to Apple iOS. As it provides third-party applications only limited access to system functionality, it significantly reduces the attack potential of malware. Furthermore, Windows Phone 8 is still quite resistant to rooting, which also prevents powerful attacks. The main drawback of Windows Phone 8 is probably its small market share compared to the market-dominating platforms Android and iOS.

3.1.4 Platform Assessment

Findings of the analysis of the three mobile platforms Google Android, Apple iOS, and Microsoft Windows Phone 8 can be used to answer the four above-defined research questions. From the answers to these questions, those platforms can be identified that are best-suited for the two identified relevant m-government use cases.

The goal of Research Question RQ-1 is to determine, which platform is best to be chosen when providing employees of public administrations with mobile end-user devices. Taking into account obtained results of the conducted platform analyses, Apple iOS turns out to be the best choice followed by Microsoft Windows Phone 8. While both platforms provide a sufficient level of security, Apple iOS seems slightly advantageous due to its larger market share [IDC, 2014]. Google Android seems inappropriate for several reasons. First, Android provides only weak support for MDM. However, MDM support is an important aspect, as MDM is the only opportunity for public administrations to retain control over issued end-user devices. Only if MDM is adequately supported, security features such as encryption or access protection can be enforced on issued devices. Second, Android is more prone to malware than other platforms. This raises additional risks, when these devices are used for official affairs. For these reasons, Research Question RQ-1 must be answered as follows: When providing employees of public

[11] http://www.windowsphone.com/en-US/store

administrations with mobile end-user devices, the mobile platforms Apple iOS and Microsoft Windows Phone 8 should be preferred to Google Android.

Similar considerations also apply to Research Question RQ-2. This research question aims to identify the platform best suited for BYOD strategies pursued by public administrations. When deploying a BYOD solution, platform fragmentation is the most relevant issue. An increasing number of different devices and operating-system versions complicates their infrastructural support and their integration into existing services. The conducted platform analyses have revealed that Google Android is clearly disadvantageous in terms of fragmentation. In contrast, Apple iOS is advantageous, as it is the only platform, for which hardware and software are provided by one vendor only. This limits the potential number of different devices and mobile operating systems to be integrated into the corporate infrastructure and hence reduces complexity and maintenance costs. For this reason, Research Question RQ-2 must be answered as follows: When pursuing a BYOD strategy, public administrations should primarily support Apple iOS followed by Windows Phone 8.

While the first two research questions are mainly relevant for use cases, in which employees of public administration make use of mobile end-user devices, Research Question RQ-3 focuses on classical m-government services provided by public administrations for citizens. Concretely, Research Question RQ-3 asks, for which mobile platforms security-critical applications should be provided by public administrations. Analyses of the three platforms Google Android, Apple iOS, and Microsoft Windows Phone 8 have shown that the latter currently appears to be the most secure platform. Windows Phone 8 provides a similar security level to Apple iOS. Additionally, there are currently hardly any known security flaws that could be used to root or jailbreak Windows Phone 8 based devices. This again raises the provided level of security and makes Windows Phone 8 currently the most secure platform. Accordingly, Research Question Q3 can be answered as follows: Development and provision of security-critical m-government applications for citizens should be mainly based on the platforms Microsoft Windows Phone 8 and Apple iOS. Android should only be supported, if the application does only rely on functionality and features, for which the platform is able to provide the required level of security.

For non-critical applications, the situation is completely different. For m-government solutions that do not store, process, or transmit any security-critical data, the provided level of security is less important. For these applications, especially the set of provided features and functionality is relevant. According to the obtained analysis results, Google Android is advantageous in this regard. From all analyzed platforms, Android provides third-party apps the most flexible and powerful API to access system functionality. Furthermore, Android is most powerful with regard to inter-application and inter-process communication. Hence, Research Question RQ-4, which asks for the most suitable platform for non-critical m-government applications, can be answered as follows: Google Android is the best choice for m-government applications that do not process, store, or transmit security-critical data.

	Google Android	Apple iOS	Microsoft Windows Phone 8
RQ-1	• Weak support of MDM features • Prone to malware • Wide spread	• Good support of MDM features • Sufficient security • Sufficient spread	• Good support of MDM features • Sufficient security • Spare spread
RQ-2	• Different customized OS versions • Different hardware vendors	• One OS vendor • One hardware vendor	• One OS vendor • Some hardware vendors
RQ-3	• Weak security	• Sufficient security	• Good security
RQ-4	• Good flexibility	• Weak flexibility	• Weak flexibility

Figure 3.1: Apple iOS and Microsoft Windows Phone 8 are advantageous for security-critical applications, whereas Google Android is beneficial in terms of functionality.

Figure 3.1 summarizes the obtained assessment results. From this figure, it becomes apparent that the platforms Apple iOS and Microsoft Windows Phone 8 are advantageous especially for security-critical

applications. On the other hand, Google Android is beneficial for non-critical applications, as it provides third-party apps the broadest set of functionality.

3.1.5 Lessons Learned

From the conducted platform analyses and the subsequent assessment, several findings can be derived. These findings are relevant for providers of m-government solutions, as they provide a useful basis for the choice and support of appropriate mobile platforms. In particular, the following findings have been derived:

- **Heterogeneous ecosystem of mobile platforms:** The conducted analyses have focused on the three most widely spread platforms only. Other platforms such as BlackBerry[12] have not been considered in detail. Although the scope of the analyses has been limited to three platforms, a very heterogeneous picture of the current mobile-platform ecosystem has been drawn. The three analyzed platforms differ significantly in terms of provided functionality and security. Although most of them share several similar concepts and ideas, their actual implementation differs in various aspects. The heterogeneity of the current mobile-platform landscape is also emphasized by the fact that all analyzed platforms are incompatible to each other to a certain extent. This means that native third-party apps need to be developed for each platform individually. A future convergence of the different platforms is not to be expected.

- **Trade-off between functionality and security:** Analyses and assessments of the three platforms Google Android, Apple iOS, and Microsoft Windows Phone 8 have mainly focused on their provided levels of functionality and security. Obtained results have shown that for all platforms there is a trade-off between security and functionality. There is no platform that provides both a high degree of functionality and a high degree of security at the same time. Instead, platforms are either beneficial in terms of security or in terms of functionality. This is comprehensible to a certain extent, as functionality provided to third-party apps can theoretically also be exploited by malicious software. At the same time, a limited set of functionality provided to legitimate third-party apps also reduces the capabilities of malware.

- **Google Android for enhanced functionality:** Google Android has turned out to be the platform that provides third-party apps the highest degree of functionality. This applies to APIs provided for access to system functionalities as well as to the support of inter-application and inter-process communication. However, the rich set of provided features and functionalities also comes with a drawback: Google Android provides the poorest security of all analyzed platforms.

- **Apple iOS and Microsoft Windows Phone 8 for enhanced security:** Apple iOS and Microsoft Windows Phone 8 have turned out to clearly outperform Google Android in terms of security. Both platforms implement similar security functions to provide an adequate level of security. First and foremost, they radically restrict provided functionality for third-party apps. Furthermore, they limit the distribution of third-party apps to official application stores, where all apps are subject to rigorous review processes. While these measures improve the provided level of security, they significantly reduce the provided functionality for third-party apps.

In summary, it can be concluded that there is currently no mobile platform that is able to provide both an adequate level of security and functionality. This is especially problematic for complex mobile applications such as mobile eID and electronic-signature solutions, which typically require both a high level of functionality and security. To pave the way for such solutions, we have investigated approaches

[12] http://www.blackberry.com/

and techniques to improve the security of mobile platforms and to achieve an appropriate level of security for applications running on potentially insecure devices. A selection of own work on this topic is presented in the following section.

3.2 Securing Applications on Mobile Devices

Own research and related scientific work reveal that current mobile platforms must not be assumed to be secure. While researchers such as Enck et al. [2009], Barrera et al. [2010], Shabtai et al. [2010], Enck and Octeau [2011], or Rogers and Goadrich [2012] have provided analyses and comparisons of mobile platforms and their security on a rather general level, own research has mainly focused on m-government-related use cases. Nonetheless, drawn conclusions resemble each other and reveal that that currently no mobile platform is able to provide absolute security. Recent incidents involving mobile malware show that this not just a theoretic problem, but a real threat especially for security-critical mobile solutions [Check Point Software Technologies Ltd., 2012].

The situation is even aggravated by the fact that classical security-enhancing solutions such as anti-virus software are in many cases not applicable on mobile end-user devices. For the special case of anti-virus software, sandboxing mechanisms, which are implemented by most mobile platforms, prevent the realization of reliable malware-detection mechanisms. Similar restrictions also apply to other security-enhancing concepts that are well-known from classical end-user devices but cannot be applied on mobile platforms. In most cases, these restrictions are due to the significant architectural and conceptual differences between classical end-user devices and mobile platforms.

To overcome this problem, several alternative approaches and methods have been proposed that aim to enhance the security of mobile platforms. For instance, Bläsing et al. [2010] have proposed an Android application sandbox that performs static and dynamic analyses of Android applications, in order to identify malware. A characterization of Android malware including more than 1,200 malware samples has been provided by Zhou and Jiang [2012], who have shown that current security software is able to detect a subset of existing malware only. We have also contributed to the ongoing research on mobile security. In particular, we have focused on security challenges currently present on Android, as this platform is most prone to security threats. In the following subsections, a brief overview of own publications on this topic is provided in order to draw a representative picture of the current state of research. For each publication, the thesis author's contribution is emphasized.

3.2.1 Detection of SMS Sniffers and SMS Catchers

The conducted survey on m-government services has revealed that SMS has been one of the predominating technologies for several years. In various developing countries, SMS is still one of the main enabling technologies of m-government solutions. In developed countries, SMS is slowly being replaced by more powerful technologies. Nevertheless, SMS is still the technology of choice for various scenarios and use cases. For instance, SMS is frequently used by mobile eID and electronic-signature solutions. SIM-based solutions rely on SMS technology to transfer relevant data between the involved mobile-network operator and the user's SIM. Server-based eID and electronic-signature solutions such as the Austrian Mobile Phone Signature make use of SMS messages to deliver one-time passwords during the user-authentication process. In both cases, the confidentiality and integrity of exchanged SMS messages is crucial for the security of the eID and electronic-signature solution. These concrete examples show that the security of SMS can be critical for m-government applications.

For many years, SMS technology has been implicitly assumed to be sufficiently secure. In the pre-smartphone era, SMS-receiving functionalities and SMS-sending functionalities were directly integrated into the operating system of mobile phones. There was no opportunity to access SMS messages with other means than those provided by the mobile phone's operating system. Although several attacks on

security mechanisms of the underlying GSM standard were known [Barkan et al., 2008], mounting of these attacks required expensive equipment and was rather complicated. In practice, SMS technology was hence justifiably assumed to be sufficiently secure.

This situation has suddenly changed with the introduction of smartphones. Smartphone operating-systems such as iOS or Android enable the easy installation of additional software in the form of mobile apps. Additionally, they provide an API that can be used by third-party apps to access functionality and resources of the mobile device. Under Android, this includes access to SMS functionality. Concretely, third-party apps can use a public Android API to intercept incoming SMS messages and to send SMS messages to arbitrary recipients. While provision of this API gives third-party apps the opportunity to include SMS functionality into their business logic, it also bears risks. Obviously, also malware being installed on the mobile device can use this API to eavesdrop security-critical data received via SMS. In 2012, the potential of this kind of malware has been shown by the attack campaign Eurograbber [Check Point Software Technologies Ltd., 2012]. Eurograbber has been based on a mobile variant of the Zeus banking trojan [Symantec, 2010] and has targeted Android-based devices to intercept incoming SMS messages containing one-time passwords for financial transactions. Within a short period of time, Eurograbber has stolen more than 36 million Euro from European bank accounts.

Eurograbber is a prime example to emphasize the relevance of SMS security on Android-based mobile end-user devices. Unfortunately, an appropriate level of security for SMS messages is difficult to achieve in practice. In theory, Android users have the power to prevent attacks on SMS messages by refusing to grant third-party applications the required permission to access SMS functionality. Unfortunately, this approach usually does not work well in practice, as users often do not understand Android's permission system or are too generous in granting requested permissions. This has been discussed in detail by Felt et al. [2012]. Hence, the presence of malware spying on incoming SMS messages, i.e. SMS sniffers, and of malware intercepting and subsequently suppressing received SMS messages, i.e. SMS catchers, cannot be precluded. It is therefore important to provide reliable and efficient means to detect the presence of such malware at runtime, in order to be able to prevent the execution of security-critical SMS-based application on potentially insecure device. For this reason, we have proposed and discussed different methods to detect SMS sniffers and SMS catchers on Android-based devices [Teufl et al., 2014b]. The author of this thesis has contributed to these activities by collaborating during the conceptual design and by establishing relevant relationships to the topic of m-government and mobile electronic signatures. Fundamental approaches followed by the proposed detection methods are briefly sketched in the following subsection.

3.2.1.1 Approaches

In total, we have proposed three approaches to detect SMS sniffers and SMS catchers on Android devices at runtime. Each of these approaches can be used to detect certain types of SMS sniffers and SMS catchers. Their detectability mainly depends on the malware implementation and on realized countermeasures that prevent a successful detection. Hence, none of the three proposed approaches guarantees a reliable detection of all SMS sniffers and SMS catchers. However, by combining these approaches, the probability for a successful detection can be considerably increased.

The first approach to detect malware targeting SMS messages is rather simple and simply checks the manifest file of all installed apps. The manifest file is a mandatory component of each Android app. Among other app-specific properties, it contains a list of all permissions that have been granted by the user to the app during its installation. Hence, by analyzing the manifest files of all installed apps, those apps can be identified that have the permission and hence the theoretic capability to spy on or to intercept SMS messages. As this analysis does not distinguish between benign and malicious apps, it can rather be used to shrink down the set of suspicious apps than to clearly identify malware.

In addition to the set of granted permissions, the manifest file of an Android app also contains declarations of static broadcast receivers. Under Android, broadcast receivers can be registered by apps

for certain events such as the reception of an SMS message. If this event occurs, the operating system broadcasts a message to notify all apps that have registered a respective broadcast receiver. Static broadcast receivers are a convenient way to intercept incoming SMS messages. Hence, by analyzing the manifest files of all installed applications, those apps can be detected that declare a static broadcast receiver to intercept incoming SMS messages. As static broadcast receivers are not the only way to access SMS messages, this analysis does not guarantee detection of all apps with SMS-receiving capabilities. However, it reliably identifies those apps that make use of static broadcast receivers for this purpose.

The second approach that we have followed to detect SMS sniffers and catchers on Android devices makes use of two broadcast receivers that are able to receive incoming SMS messages. In general, broadcast receivers can be assigned with a priority. If multiple apps have registered a broadcast receiver for a specific event, e.g. the reception of an SMS message, the assigned priority determines the order, in which these apps are notified. Each notified app can decide if broadcast receivers with lower priority shall still be notified, or if the broadcast message shall be discarded. To detect installed SMS catchers, we have proposed the registration of a broadcast receiver with maximum priority and of another one with minimum priority. This assures that the first broadcast receiver is notified first, whenever an SMS message is received. Accordingly, the second registered broadcast receiver is always the last one to be notified. All other apps with registered broadcast receivers are notified between these two broadcast receivers. If there is an installed SMS catcher intercepting an incoming SMS message and subsequently discarding the respective broadcast message, the registered broadcast receiver with assigned minimum priority does never receive the broadcast message. Thus, by comparing the sets of broadcast messages received by the maximum-priority receiver and by the minimum-priority receiver, installed SMS catchers can be detected.

In practice, several limitations need to be taken into account, which reduce the reliability of this approach. For instance, the Android API documentation defines a valid range for assigned priorities but obviously also allows the definition of priorities outside of this range. This complicates the definition of broadcast receivers with minimum and maximum priorities. In addition, the Android API documentation states that broadcast receivers with equal priority are called in an arbitrary order. Hence, if an installed SMS catcher assigns its broadcast receiver with the maximum priority, it cannot be guaranteed that the implemented malware detection succeeds.

The third approach that has been followed to detect SMS catchers and sniffers has been based on static code analysis. This approach is the most powerful but also the most complex method. It analyzes the source code of installed apps in order to identify code fragments that indicate access to SMS functionality. Even though this approach relies on an elaborate technique, it still suffers from some limitations. First, it cannot properly analyze included program libraries or Android's service architecture. Furthermore, malware authors being aware of the capabilities of the used analysis tool can systematically avoid the detection of suspicious code by applying appropriate obfuscation techniques.

We have provided a more detailed introduction of proposed malware-detection techniques in Teufl et al. [2014b]. Although all proposed approaches suffer from some theoretic limitations, they are still able to detect SMS sniffers and SMS catchers in practice. This has been shown by means of a concrete implementation of the three approaches. Findings obtained from these implementations are discussed in the following section.

3.2.1.2 Results and Findings

Implementation of the three approaches to detect SMS sniffers and SMS catchers on Android devices has revealed that all approaches are feasible in practice. All required functionality such as access to manifest files of installed applications, the registration of broadcast receivers with arbitrary priorities, or access to required data for static code analysis is provided by the Android API. Thus, there are no technical hurdles to realize the proposed approaches in practice.

Their concrete implementation has shown that the three proposed approaches differ considerably in terms of their capabilities to reliably detect malware and also in terms of complexity. In general, the complexity of an approach is directly proportional to its malware-detection capabilities. Still, also simple approaches can be useful, as they enable a fast and efficient elimination of benign applications. For instance, simple manifest checks can be used to determine all apps that do not have the required permissions to access SMS messages. For these apps, more complex malware-detection mechanisms can be skipped in order to save resources. It hence makes sense to combine available malware-detection approaches such that simple approaches are applied first to efficiently reduce the set of suspicious applications so that complex approaches need to be applied to a few apps only. We have proposed a comprehensive workflow that pursues this strategy in Teufl et al. [2014b].

In summary, our contribution on the detection of SMS sniffers and SMS catchers has revealed several interesting findings. First, our work and research has substantiated the awareness that provision of secure and reliable SMS functionality on Android is still an open issue, as the infection of Android-based devices with SMS sniffers and SMS catchers can hardly be avoided in practice. Second, we have shown that there are various possible approaches to detect this kind of malware on mobile devices at runtime. Finally, our work has also revealed that even though these approaches are feasible in practice, they cannot provide absolute security, as they suffer from several conceptual and practical limitations. Hence, it must be concluded that SMS messages still must not be assumed to be absolutely secure on Android.

3.2.2 Malware Detection Using Side-Channel Information

Due to the special architecture of mobile platforms, classical malware-detection approaches such as antivirus software do not work reliably on smartphones and related mobile end-user devices. Concretely, implemented sandboxing mechanisms prevent an efficient and reliable application of approved malware-detection solutions. As access to resources of installed apps is restricted, these solutions are unable to access data needed to carry out required analyses. Ironically, implemented security features of mobile platforms hence render the application of a reliable malware detection infeasible.

This is problematic, as malware is an emerging threat also on mobile platforms. Especially Android-based devices and their users are faced with a multitude of malware [Kelly, 2014]. The continuous increase of malware and the incompatibility of classical malware-detection solutions raise the demand for alternative approaches to distinguish benign applications from malicious ones on mobile platforms. As relevant data cannot be accessed due to implemented sandboxing mechanisms, alternative data sources need to be employed for the analysis of installed apps.

In this context, underlying concepts of side-channel attacks appear to be an interesting alternative. Side-channel attacks have especially gained popularity due to the pioneering work by Kocher et al. [1999] and are also often referred to as implementation attacks. In contrast to classical crytanalysis, implementation attacks do not target a cryptographic algorithm itself, but rather its implementation. For this purpose, implementation attacks make use of additional information leaked by the implementation of the cryptographic algorithm. For instance, Differential Power Analysis (DPA), which is a subcategory of side-channel attacks, makes use of the power consumption of cryptographic devices to reveal secret key material that is stored and processed by the targeted device. Another example for implementation attacks are so-called timing attacks, which analyze timing behavior of cryptographic systems to reveal secret key material. Examples are attacks proposed by Kocher [1996] or Schindler [2000].

Although there are different variants of side-channel attacks, they all follow the same basic approach, i.e. they make use of unintentionally leaked information. We have shown that this basic approach can also be followed to classify apps running on mobile end-user devices [Zefferer et al., 2013d]. Concretely, we have shown that power-consumption information provided by a third-party tool can be used to classify different types of apps. Implementing tasks of this activity have mainly been undertaken by master students during a student project. Hence, the author of this thesis has mainly contributed by assisting in elaborating the conceptual design and by providing support for the involved students. In the following,

employed techniques to classify apps by means of available side-channel information are introduced and obtained results and findings are presented and discussed.

3.2.2.1 Techniques

Classical side-channel attacks relying on power-consumption information typically make use of sophisticated measurement techniques to obtain accurate side-channel information. These techniques usually involve digital storage oscilloscopes and other expensive equipment requiring a suitable laboratory environment. Such a setup seems inappropriate for the implementation of malware-detection solutions for mobile end-user devices. Such solutions should ideally be applicable directly on the mobile device and should produce immediate results without requiring the user to set up a sophisticated measurement equipment. For this reason, we have relied on rather coarse-grained power-consumption information provided by the third-party tool PowerTutor which has been introduced by Zhang et al. [2010]. PowerTutor is a mobile app that provides power-consumption information for all apps running on the same device. For each app, power-consumption information for the six components Central Processing Unit (CPU), Audio, Display, WiFi, 3G, and Global Positioning System (GPS) are provided. As an illustrative example, Figure 3.2 shows power-consumption measurements provided by PowerTutor for the two apps Android Browser and Lookout Mobile Security[13]. The obvious differences between the two displayed power traces show that the power-consumption is individual for each app.

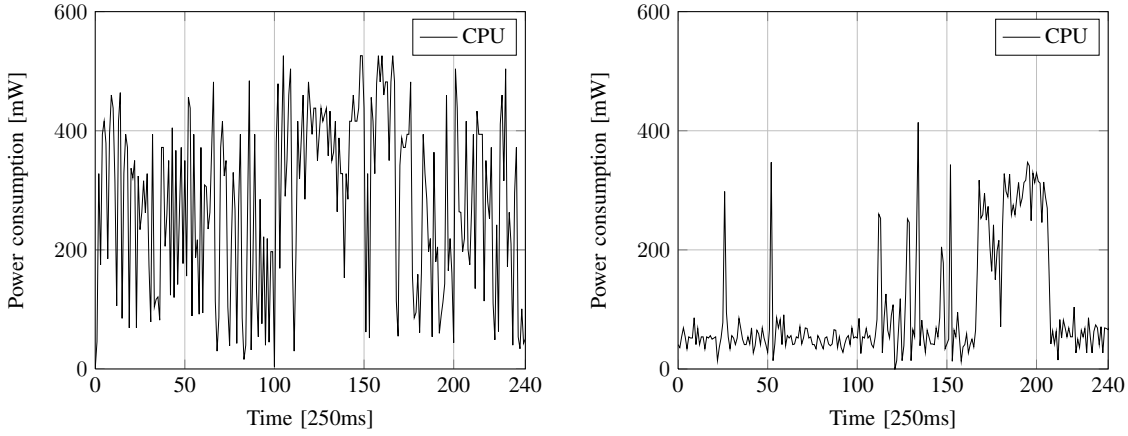

Figure 3.2: The measured power consumptions of the mobile apps Android Browser (left power trace) and Lookout Mobile Security (right power trace) differ considerably from each other.

While PowerTutor offers a convenient way to obtain app-specific power-consumption information, reliance on this data source also raises several issues. For instance, PowerTutor does not carry out real measurements but merely provides accurate estimations of the actual power consumption. Furthermore, a mobile app's power consumption is influenced by various factors including varying user inputs, the processed data, or different screen orientations. These issues render a reliable classification by means of one single power-consumption measurement difficult. This is also illustrated in Figure 3.3, which displays two power-consumption measurements of one and the same mobile app. Due to various influencing factors, these measurements slightly vary.

As a solution to this problem, we have employed two different techniques to combine and systematically analyze multiple power-consumption measurements of one and the same app in order to enable a reliable classification of apps. Both techniques rely on basic machine-learning concepts and hence comprise a learning phase and a classification phase. During the learning phase, a model is created and

[13] https://play.google.com/store/apps/developer?id=Lookout+Mobile+Security&hl=en

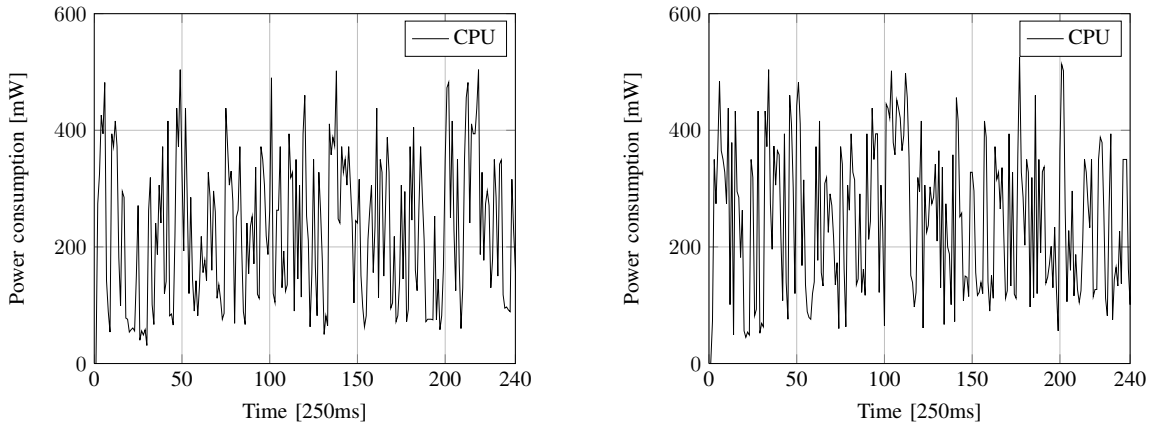

Figure 3.3: Measuring the power consumption of a mobile app twice usually leads to slightly different results.

trained using known input data. In the subsequent classification phase, the trained model is then used to classify unknown input data. The two employed techniques differ in terms of how the model is created, trained, and finally used to classify apps.

The first technique makes use of power-consumption histograms. A separate histogram is created for each kind of app that shall later be distinguishable. App-specific histograms are simply created by counting how often the application's power consumption is on or above a certain level. This way, histograms such as the one shown in Figure 3.4 can be created during the learning phase using power consumptions of known applications. In the subsequent classification phase, similar histograms are created for apps to be classified. The resulting histogram is then compared with available trained histograms. Comparisons can for instance be based on distance-measures such as cosine similarity.

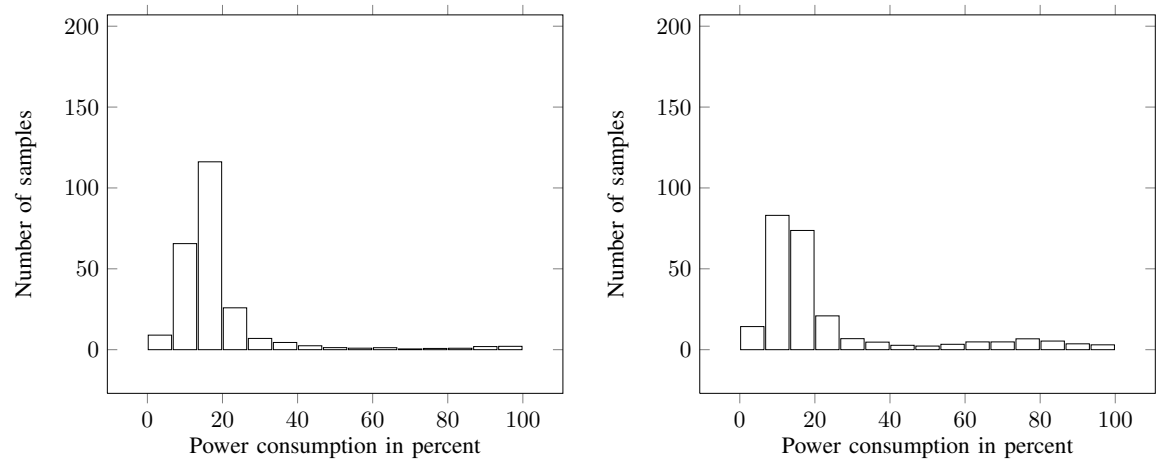

Figure 3.4: The power-consumption histogram of the idle process (left figure) differs from the power-consumption histogram of the mobile app Lookout Mobile Security (right figure).

The second technique employed to classify mobile apps according to their power consumption follows a more sophisticated and more complex approach. Concretely, this technique relies on Mel Frequency Cepstral Coefficients (MFCCs). MFCCs are extracted from collected power-consumption measurements. Their distribution is then used to create Gaussian Mixture Models (GMMs), which finally define a representation of the measured power consumption. This is exemplified in Figure 3.5, which il-

lustrates GMMs derived from the distributions of Coefficient #6 and Coefficient #7. MFCCs and GMMs are frequently used in speaker-recognition systems and music-similarity finders. As the problem of matching recorded voice to a certain person can be conceptually compared to the problem of mapping a measured power consumption to a certain mobile app, we have employed the concepts behind MFCCs and GMMs to classify apps.

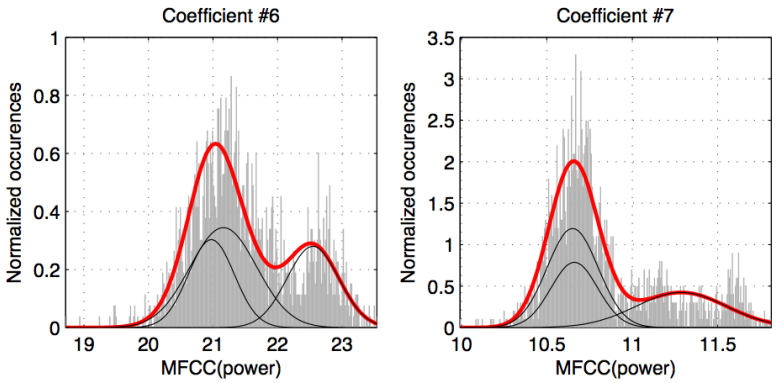

Figure 3.5: Gaussian Mixture Models can be derived from the distribution of MFCCs as exemplified for Coefficient #6 and Coefficient #7.

3.2.2.2 Results and Findings

In order to verify their capabilities and limitations, we have implemented the two proposed classification techniques. We have then used these implementations to classify mobile apps according to the six categories Games, Internet, Idle, Malware, Music, and Multimedia. For each of these categories, classification models have been created during a learning phase according to the two proposed techniques. In the subsequent classification phase, we have then divided unknown apps using the created classification models.

Obtained results show that both approaches are basically suitable for the power consumption based classification of mobile apps. From the obtained results, various findings can be derived. Malware being mainly executed in the background is distinguishable from applications being actively used at the moment. Furthermore, applications assigned to the categories Games, Internet, Music, and Multimedia can also be successfully distinguished. However, due to their similar usage pattern and functionality, music and multimedia applications are in general more difficult to distinguish correctly.

Obtained results also show that the MFCC-based technique works better for distinguishing apps from the categories Idle and Malware. Hence, this approach appears to be better suited for malware-detection purposes. On the other hand, the histogram-based technique constitutes a fast and efficient classification method, which can be especially suitable for mobile end-user devices with limited computational power.

In summary, it can be concluded that the proposed techniques represent a promising alternative for future malware-detection solutions. Even though first results are promising, several further improvements are required to further develop the current prototype towards a productive malware-detection solution. Even then, it remains unclear whether the quality of available power-consumption information on mobile devices is sufficient to enable a reliable and efficient identification of malware. So far, our solution, which we have discussed in more detail in Zefferer et al. [2013d], is merely a proof of concept and is not able to provide absolute security on mobile end-user devices.

3.2.3 Policy-Based Security-Assessment of Mobile End-User Devices

Related work and own research show that absolute security cannot be guaranteed on current mobile platforms. This especially applies to the mobile platform Android, which suffers from several security-

reducing characteristics. Concretely, its support for alternative application sources, the optional nature of integrated security features such as access protection or file-system encryption, its complex and in practice often ineffectual permission system, its rich API, and its support for rich runtime capabilities such as inter-process communication limit Android's overall security in practice. This leads to a situation, in which security-critical Android-based apps must not assume to be executed in a secure environment, and in which stored and processed security-critical data is potentially vulnerable to malware and related threats. Although the situation is less critical on other mobile platforms, they still must not be assumed to be absolutely secure either.

To relieve this unsatisfactory situation, we have proposed a policy-based security-assessment framework for mobile end-user devices [Zefferer and Teufl, 2013]. The author of this thesis has contributed to this activity by forming the basic concept, developing the proposed framework's architecture, and by evaluating the proposed solution by means of a concrete implementation. The resulting framework assesses the current level of security of a mobile end-user device and helps security-critical applications installed on this device to decide whether an execution of security-critical functionality is currently secure or not. This enables apps to refrain from exposing security-critical data on potentially insecure devices. In the following, the architecture of the proposed framework is briefly sketched and relevant results and findings that have been derived from its implementation and evaluation are discussed.

3.2.3.1 Architecture

The architecture of the proposed security-assessment framework is shown in Figure 3.6. From a conceptual perspective, the framework is located between the mobile operating system, diverse external information sources, and the security-critical mobile app. The framework provides an API, through which its functionality can be accessed by this app. Concretely, the API enables the app to define a security policy consisting of an arbitrary number of logically combined security properties. A security property represents a characteristic of the mobile device to be assessed. Examples for security properties are *'encryption activated'*, *'no alternative keyboard installed'*, or *'alternative application stores disabled'*. Security policies can combine an arbitrary number of security properties using the logical AND and OR operators. Furthermore, the API also supports the definition of sub policies, which can again be logically combined to build up more complex policies. A possible security policy including two sub policies is exemplified in Equations 3.1, 3.2, and 3.3.

$$SubPolicyA = OR(SecurityProperty1, SecurityProperty2, SecurityProperty3) \tag{3.1}$$

$$SubPolicyB = AND(SecurityProperty4, SecurityProperty5) \tag{3.2}$$

$$SecurityPolicy = AND(SubPolicyA, SubPolicyB, SecurityProperty6) \tag{3.3}$$

Defined security policies are forwarded to the framework's Policy Interpreter. This component extracts all relevant security properties from the defined policy and forwards them to the Assessment Core Module. The function of the Assessment Core Module is the assessment of all relevant security properties on the current platform. For this purpose, the Assessment Core Module makes use of different Assessment Plug-Ins. Each Assessment Plug-In is capable to assess one or more security properties. Therefore, the Assessment Core Module assigns security properties to be assessed to available Assessment Plug-Ins according to their capabilities. To assess a security property, Assessment Plug-Ins typically interface either the Android operating system or external information sources, in order to retrieve required information. For instance, in order to assess the security property *'encryption activated'*, the responsible Assessment Plug-In evaluates the respective configuration setting of the underlying operating system.

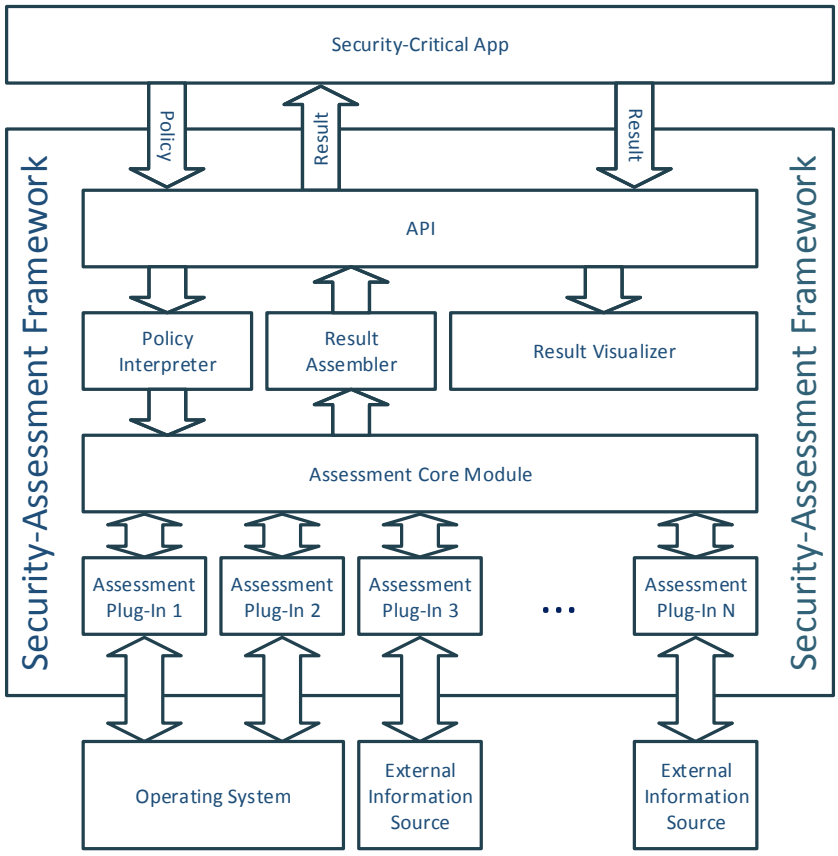

Figure 3.6: The proposed security-assessment framework supports the definition and assessment of arbitrary security policies.

The various Assessment Plug-Ins return their assessment results to the Assessment Core Module. This module collects all results and forwards them to the Result Assembler. This component combines the collected results according to the defined security policy using the logical AND and OR operators. This way, the Result Assembler determines, whether the defined security policy is met by the underlying mobile device or not. The determined result is returned via the framework's API to the security-critical app, which can rely on this result to decide whether or not execution of security-critical tasks on the given device is secure.

Another feature provided by the security-assessment framework is covered by the component Result Visualizer. The use of this component by the security-critical app is optional. The Result Visualizer can be used to graphically display assessment results to the user. This way, the user can be informed in a convenient and usable way about problematic device properties that potentially prevent the execution of security-critical applications and operations.

We have implemented the proposed security-assessment framework in order to assess the appropriateness of its architecture in practice. From this implementation, several useful findings have been derived. These findings are discussed in the following.

3.2.3.2 Results and Findings

To evaluate its feasibility, effectiveness, and efficiency, we have realized the proposed security-assessment framework in practice. For this purpose, we have chosen the Android platform mainly for two reasons. First, Android is currently vulnerable to various attack scenarios and frequently targeted by malware

[Kelly, 2014]. Second, Android provides powerful APIs and rich runtime capabilities, which facilitate the development of the proposed functionality. For these reasons, Android has been chosen for a first realization of the proposed framework.

Considering this first basic design decision, we have further decided to implement the security-assessment framework in the form of an Android Library Project[14]. This way, implemented functionality can be easily made available to arbitrary Android apps. As the concept of Library Projects is specific for Android, realizations for other platforms need to follow alternative approaches.

Pursuing the Library Project based strategy, all required functionality has been successfully implemented for Android. The developed implementation currently supports the assessment of 22 security properties by seven Assessment Plug-Ins. Due to the plug-in-based architecture, the set of supported security properties and Assessment Plug-Ins can be easily extended. In addition to required assessment functionality, our implementation also supports the visualization of assessment results. This is illustrated in Figure 3.7. Security properties and sub-policies composing the defined security policy are displayed in a tree-like structure. Users can navigate through the different levels of this structure. For each security property and sub-policy, detailed information on the respective assessment result can be displayed. This is illustrated in Figure 3.8. The implemented visualization component enables users to easily determine critical device properties and configurations that prevent a secure use of critical apps.

Figure 3.7: The implemented assessment framework supports the visualization of assessment results.

From the concrete implementation of the proposed security-assessment framework, several useful findings have been derived. First and foremost, the developed solution shows the general feasibility and applicability of the proposed framework. Its Android-based nature however reveals a potential issue. Concretely, the developed solution suffers from Android's fragmentation, i.e. from the continuously increasing number of different Android versions and devices in the field. The assessment of certain security properties potentially depends on the respective Android version. For instance, depending on the exact operating-system version, different API calls might be required to determine a particular device configuration. Therefore, version-specific implementations of assessment functionality might be required. This reduces the maintainability of the proposed solution in the long term.

[14] https://developer.android.com/tools/projects/index.html

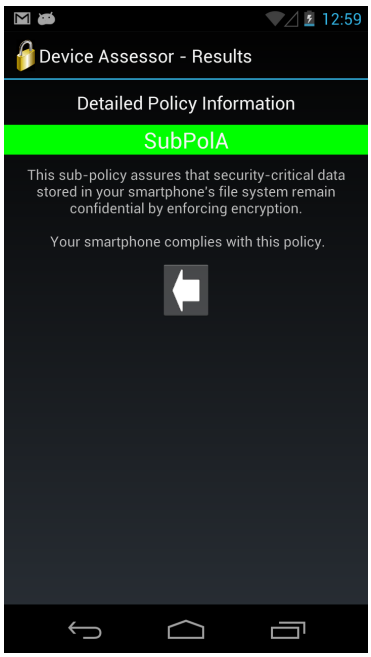

Figure 3.8: Detailed assessment results can be displayed for all security properties and sub-policies.

Another identified issue of the proposed and implemented solution is rooting. Implemented Assessment Plug-Ins may be assumed to work correctly on non-rooted devices only. In case an attacker gains root access to the mobile operating system, this attacker must be assumed to be able to circumvent any implemented security-assessment technique and to outsmart implemented Assessment Plug-Ins. To reduce threats imposed by rooting, the implemented solution features a root-detection plug-in. This plug-in checks whether the device is rooted or not. Accordingly, security-critical apps can include the requirement for a non-rooted device as security property into defined security policies. However, the provided root-detection plug-in suffers from the same limitations as all other Assessment Plug-Ins. An attacker with root access must be assumed to have the opportunity to circumvent all implemented root-detection mechanisms. For this reason, the proposed and developed security-assessment framework is able to provide reliable assessment results on non-rooted devices only. We have provided a more detailed discussion of capabilities and limitations of our proposed security-assessment framework and its implementation in Zefferer and Teufl [2013].

3.2.4 Lessons Learned

Improving the security of mobile end-user devices has been a topic of scientific interest for several years. In this section, an overview of own research and own contributions on this topic has been provided. These contributions range from mechanisms to detect SMS-based malware, over classification methods for mobile apps based on side-channel information and machine-learning techniques, to the provision of a security-assessment framework for mobile end-user devices. From these contributions, several useful lessons have been learned.

First, our contributions have increased the awareness that security on mobile end-user devices is of growing relevance but still difficult to achieve in practice. Furthermore, we have shown that modern mobile end-user devices enable various alternative and innovative approaches to improve security. All of the proposed approaches have the potential to raise the achievable level of security. On the other hand, all proposed approaches also suffer from several limitations in practice. In summary, it must hence be concluded that perfect security is still not available on mobile end-user devices.

The lack of absolutely secure execution environments on mobile end-user devices raises problems for security-critical apps. This also applies to m-government apps that store, process, or transfer security-critical data. In particular, this applies to mobile electronic-signature solutions, which have been identified as key for transactional m-government services. We have assessed potential limitations that arise with current mobile electronic-signature solutions when being applied on mobile end-user devices. Relevant own work on these assessments is sketched in the following section.

3.3 Qualified Electronic Signatures on Mobile Devices

As electronic signatures are a basic building block of transactional e-government services, their availability and applicability on mobile end-user devices is crucial for transactional m-government. During the past years, electronic-signature solutions have been developed and deployed in various European countries. However, nearly all of these solutions have been designed for classical end-user devices such as desktop PCs and laptops. Even though some of today's signature solutions also integrate mobile technologies, they are not intended to be applied on mobile end-user devices either. In some cases, applying these solution on mobile end-user devices is technically feasible in principle. However, this raises the question, if the required level of security can be achieved, as underlying security concepts usually do not assume the respective signature solution to be applied on mobile end-user devices.

To answer this question, we have evaluated the applicability of existing signature solutions on modern mobile end-user devices in more detail. Therefore, we have focused on both the creation of electronic signatures as well as on their verification. For both use cases, we have designed and developed respective solutions for mobile end-user devices. We briefly introduce these solutions and discuss findings obtained from their evaluation in the following subsections. For each solution, the thesis author's contribution is emphasized.

3.3.1 Signature Creation

To evaluate the feasibility of signature-creation solutions on mobile end-user devices, we have mainly focused on the productive signature solution Austrian Mobile Phone Signature[15]. As this solution follows a server-based approach and incorporates mobile technologies, this solution seems perfectly suitable to be applied on mobile end-user devices. We have evaluated the application of the Austrian Mobile Phone Signature on mobile end-user devices by means of several concrete applications. These applications are briefly sketched in the following subsections.

3.3.1.1 Integrating Electronic-Signature Solutions into SMS-Based Services

Surveys of current m-government solutions show that SMS-based services are still popular. This especially applies to developing countries, where mobile 3G networks are often not available. Interestingly, SMS-based services are also still popular in developed countries, where these services offer a simple, cheap, and convenient alternative to app-based solutions. In most cases, SMS-based services are still rather simple and do not integrate electronic-signature functionality. Thus, these services usually have a purely informational character. Transactional SMS-based m-government services are rare. To overcome this limitation, we have developed an SMS-based service that incorporates electronic signatures [Zefferer et al., 2012a]. This way, we have tested the hypothesis that transactional m-government services can be implemented on unmodified mobile devices, even if available communication technologies are limited to SMS. The author of this thesis has contributed to this activity by defining the developed solution's architecture and by carrying out a thorough evaluation by means of a fully functional implementation.

[15] http://www.handy-signatur.at/

The developed solution has been realized as web application, which features both a web-based and an SMS-based user interface. The latter has been realized by means of an external SMS gateway. The implemented web-based interface is used for administrative tasks only and basically enables the registration and maintenance of user accounts. Once registered at the application, users can carry out complete transactional procedures by sending and receiving SMS messages only. Concretely, the developed application supports the remote generation of documents in Portable Document Format (PDF) based on input data provided by SMS, the signing of generated documents, and the delivery of signed documents by e-mail.

Capabilities of the developed application have been shown by means of a concrete procedure, which enables users to generate, sign, and deliver sick notes using SMS messages. By sending a well-defined SMS message, users can trigger the generation of a PDF-based sick note in the central web application. Subsequently, the generated sick note is electronically signed. The signed sick note is then sent to the user's employer by e-mail. Generated, signed, and delivered sick notes are stored by the central web application and can later be accessed by the user through the application's web interface.

From an implementation perspective, integration of signature-creation functionality, i.e. supporting the signing of created PDF documents, is the most challenging task. To enhance the developed web application with signature-creation capabilities, approved signature solutions of the Austrian e-government infrastructure have been used. Concretely, the solutions Austrian Mobile Phone Signature[16] and Modules for Online Applications - Signature Creation (MOA-SS)[17] have been employed for this purpose. Although both components can be used to create electronic signatures, they still differ in several aspects that need to be considered. The Austrian Mobile Phone Signature allows users to create qualified electronic signatures. For this purpose, the Mobile Phone Signature centrally stores a unique signature key for each user. The Mobile Phone Signature requires direct user interaction, as the user needs to authorize each signature-creation process. In contrast, MOA-SS is a pure server-signature solution. MOA-SS does not produce qualified signatures and does not require user interaction during the signature-creation process. We have discussed functionalities, capabilities, and limitations of the Austrian Mobile Phone Signature and of MOA-SS in more detail in Posch et al. [2011].

Core building blocks of the developed web-application are illustrated in Figure 3.9. The web application consists of three basic building blocks. Most functionality is implemented by the component Business Logic, which has API-based interfaces to other building blocks of the web application. Furthermore, the Business Logic interfaces several external components. This includes the Mobile Phone Signature, which is used to create qualified electronic signatures, an SMS Gateway, which enables SMS-based communication with the User, and the Simple Mail Transfer Protocol (SMTP) Server, which is used to deliver signed documents via e-mail.

Based on this architecture, carrying out an SMS-based procedure, e.g. creating, signing, and delivering a sick note, consists of the following steps. First, the User sends a well-defined SMS message to the Business Logic via the SMS Gateway. The SMS message triggers the document-generation process in the web application. Variable parts of the document are defined by data included in the sent SMS message. In addition to these variable data, the sent SMS message also needs to include information regarding the preferred signature-creation method. In case MOA-SS is selected as the preferred method, the Business Logic accesses the required signature-creation functionality through its API-based interface to MOA-SS. If the procedure requires a qualified electronic signature, the Mobile Phone Signature is typically used instead of MOA-SS. The Mobile Phone Signature requires the User to authorize each signature-creation process. For this purpose, the User needs to authenticate at the Mobile Phone Signature. The authentication process consists of two steps. First, the User has to provide his or her mobile phone number and a secret password through a web-based interface. As access to a web-based interface is not possible for pure SMS-based services, the User has to include this password in the sent SMS mes-

[16] http://www.handy-signatur.at/

[17] https://joinup.ec.europa.eu/software/moa-idspss/home

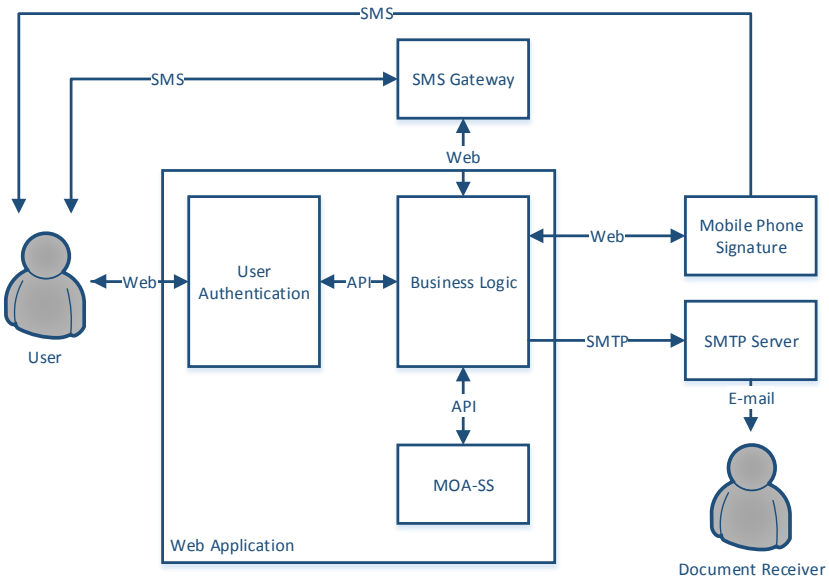

Figure 3.9: The developed web application enables the user to carry out electronic signature based procedures via SMS.

sage. The Business Logic forwards this password together with the User's mobile phone number to the Mobile Phone Signature, in order to complete the first authentication step. In the second authentication step, the Mobile Phone Signature sends a one-time password to the User via SMS. The User has to prove reception of this one-time password to complete the authentication process. For this purpose, the Mobile Phone Signature provides another web form, through which the received one-time password can be entered. Again, the web application's Business Logic has to assist the User to accomplish this task, as the User is limited to SMS-based communication. Hence, the User sends the received one-time password to the Business Logic, which forwards it to the Mobile Phone Signature. This finally completes the authentication process and authorizes the signature-creation process. The signed document is returned to the Business Logic and finally delivered with the help of the SMTP Server. The user is notified about the successful process via SMS.

The implemented web application shows that the integration of electronic signatures into SMS-based applications is in principle feasible. However, the web application also reveals several severe drawbacks of this approach. Among others, the following issues can be identified:

- **Omission of second communication channel:** The security of the Austrian Mobile Phone Signature relies partly on the fact that two separate communication channels are used during user authentication. While one-time passwords are delivered via SMS, credentials need to be provided through a web-based interface. This separation of communication channels is not possible for the developed web application, as the User needs to carry out the entire procedure including the creation of electronic signatures by exchanging SMS messages only. Due to this restriction, the developed web application is used as sort of proxy component between the User and the Mobile Phone Signature during user authentication. As the User relies on SMS-based communication only, a separation of communication channels cannot be achieved. This reduces security.

- **Additional intermediary components:** Due to the restriction to SMS-based communication, additional intermediary components are necessary. Figure 3.9 shows that all SMS-based communication is routed over an external SMS Gateway. As exchanged SMS messages potentially contain confidential data, the SMS Gateway needs to be a trusted component. Furthermore, the developed

web application acts as intermediary between the User and the Mobile Phone Signature during user authentication. As user authentication includes the provision of a secret password, the User also needs to trust the web application.

- **Display of signature data:** As the Mobile Phone Signature enables the User to create a legally binding qualified electronic signature, the User must be given the opportunity to check the data to be signed. The Mobile Phone Signature allows this during the user-authentication process by displaying the data to be signed in the web form that is used to obtain credentials from the user. Due to the restriction to SMS-based communication, the user cannot access this web form and has hence no opportunity to check the data to be signed. As a solution to this problem, the web application could send the data to be signed to the user via SMS. However, this works for simple text-based data of limited length only. In practice, this approach is inappropriate.

We have provided a more detailed security analysis of the proposed and developed solution in Zefferer et al. [2012a]. In general, it can be concluded that integrating electronic signatures into pure SMS-based services is feasible in principle. However, the restriction to SMS-based communication raises several security issues, when using integrated signature solutions through SMS-based communication channels only. This is mainly due to the special characteristics of these signature solutions, which have not been designed with pure SMS-based use cases in mind. In practice, a secure use of electronic signatures in pure SMS-based services is hence hard to achieve.

3.3.1.2 Employing Electronic Signatures for Trusted Location-Based Services

The integration of signature solutions that enable the creation of qualified electronic signatures has turned out to be difficult for pure SMS-based services. It hence seems reasonable to focus on more powerful technologies instead. In this context, mobile apps for smartphones and tablet computers appear to be an appropriate choice. Mobile apps have access to various functionality of the underlying mobile device and operating system and hence allow for more powerful applications.

We have assessed possibilities to integrate qualified electronic signatures into mobile apps by means of different applications. Our first attempt has targeted LBSs. These services have become popular during the past years. They incorporate available location information to provide context-sensitive services. Popular examples of LBSs are car-navigation apps that make use of location information to route users to arbitrary destinations. LBSs are also used for m-government-related use cases. For instance, the Dutch m-government app BuitenBeter[18] incorporates location information to facilitate the reporting of issues in public space for citizens.

Despite their growing popularity, mobile LBSs suffer from a basic drawback: assuming a potentially compromised mobile end-user device, location information provided on these devices must not be assumed to be correct and trustworthy. Even though employed positioning technologies and solutions such as GPS work reliably, there are various ways to forge location information that is provided to apps running on mobile devices. Provided location information can be theoretically forged by both, the legitimate owner and user of the mobile device, as well as by installed malware. Hence, location information available in mobile apps must not be assumed to be correct. This renders mobile applications that require trustworthy location information infeasible.

As a solution to this problem, we have proposed and introduced the concept of Trusted Location-Time Tickets (T-LTTs) [Teufl et al., 2012]. T-LTTs are data structures containing a time stamp together with location information. Each T-LTT is unambiguously assigned to a specific user. This way, a T-LTT proves that a specific user was at a certain location at a certain point in time. The correctness of the T-LTT is assured by a Trusted Third Party (TTP). The author of this thesis has contributed to the elaboration of the concept of T-LTTs and to the development of a suitable architecture.

[18] http://www.buitenbeter.nl/

The basic architecture behind the proposed concept of T-LTTs is shown in Figure 3.10. Application of the proposed concept requires three involved parties. A Service Provider requesting a T-LTT, a User providing the T-LTT, and a TTP confirming the correctness of the T-LTT. According to the shown architecture, the basic process flow is as follows. First, the User aims to access a resource from the Service Provider. Second, the Service Provider requests the User to present a valid T-LTT in order to gain access to the requested resource. Third, the User requests a T-LTT from the TTP. Fourth, the TTP issues the User the requested T-LTT. Fifth, the User presents the Service Provider the issued T-LTT. After positive verification of the presented T-LTT, the Service Provider finally grants the User access to the requested resource.

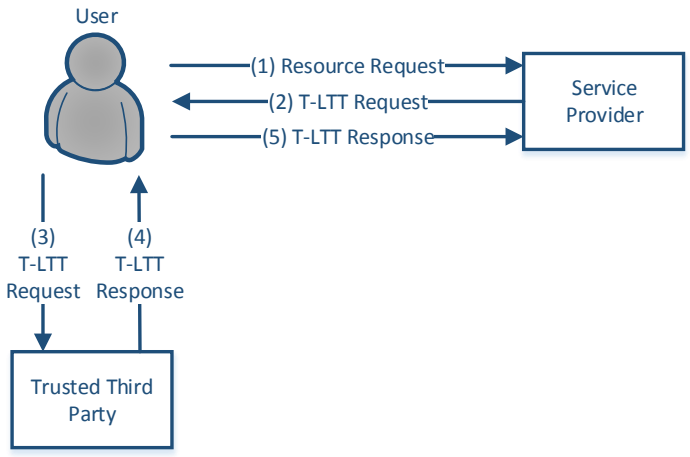

Figure 3.10: The proposed concept for the implementation of trusted location-based services defines five consecutive processing steps.

The feasibility and practical applicability of the proposed concept has been demonstrated by means of two concrete implementations. These implementations mainly differ in terms of the underlying use case but still both rely on the general architecture shown in Figure 3.10. Concretely, the two implementations differ in terms of the realization of the TTP. The TTP is implemented either by another smartphone user or by cryptography-enabled Near Field Communication (NFC) tokens that are located at fixed positions. We have discussed the concrete architecture of the two implementations in more detail in Teufl et al. [2012].

Independent of the concrete implementation of the TTP, both implementations share a basic property: they require the User to electronically sign time and location information on his or her mobile end-user device. This is necessary to bind the User's identity to the created T-LTT. To integrate the required signature-creation functionality, the two implementations again rely on the Austrian Mobile Phone Signature. While this approach works fine from a technical perspective, it raises problems from a security point of view. The Mobile Phone Signature has been developed for a use on two separate end-user devices. Only in this case, the requirement for two separate communication channels can be met. The two implementations of the proposed T-LTT concept use the Mobile Phone Signature on one single end-user device, i.e. the User's smartphone, only. Thus, these implementations ignore part of the Mobile Phone Signature's security concept, i.e. the separation of communication channels. This renders an application of these implementations in productive scenarios and use cases difficult.

3.3.1.3 Qualified PDF Signatures on Mobile End-User Devices

During the past decades, PDF has emerged to one of the most relevant formats for document exchange. Due to its popularity and frequent use, PDF is also relevant for e-government use cases. The frequent use of PDF in e-government-related use cases raises the need for mechanisms to assure their integrity, authenticity and non-repudiation. Electronic signatures have turned out to be the ideal solution for that. Hence, solutions that enable the electronic signing of PDF documents have evolved to important building blocks of e-government infrastructures. Especially in Europe, PDF-signing solutions relying on qualified electronic signatures are of relevance, as they assure that electronically signed PDF documents are legally equivalent to paper-based documents containing handwritten signatures.

Facing the recent emergence and growing relevance of mobile computing, we have investigated opportunities to apply existing PDF-signing solutions on mobile end-user devices. For this purpose, we have focused on the Austrian e-government infrastructure, in which PDF signatures have been a relevant building block for years. Due to incompatibilities of Austrian laws and deployed Austrian e-government solutions with existing PDF-signature standards, a proprietary standard has been introduced in Austria. This standard is called PDF-AS and enables the creation of qualified electronic PDF signatures. Relevant concepts behind PDF-AS have been discussed by Leitold et al. [2009] in more detail. An implementation of the PDF-AS standard is available as open-source software[19] in the form of a Java library. Based on this library, a desktop application[20] and a web application are provided that can be used to electronically sign PDF documents. The provided Java library mainly implements PDF-processing functionality. For the creation of legally binding electronic signatures, the library relies on the Austrian national eID and electronic-signature infrastructure. For instance, the library supports integration of the Austrian Mobile Phone Signature for the creation of qualified electronic signatures.

For a better understanding, the interaction between PDF-AS components, users, the Austrian Mobile Phone Signature, and external service providers is illustrated in Figure 3.11. According to this architecture, the creation of a qualified PDF signature comprises the following steps. First, the PDF document to be signed is provided to the PDF-AS Library. This can be done by using the provided PDF-AS Desktop Application or the PDF-AS Web Application. The PDF-AS Web Application can also be accessed and used by external service providers. For this purpose, it offers a web-based interface relying on Java Servlet Technology[21], through which external services can upload PDF documents to be signed. Irrespective of the way PDF documents to be signed are provided, PDF-AS makes use of the Austrian eID and electronic-signature infrastructure to create the electronic signature. The architecture shown in Figure 3.11 assumes the Mobile Phone Signature to be used for this purpose. When being provided with data to be signed, the Mobile Phone Signature authenticates the user to authorize the signature-creation process. After successful completion of this process, the created signature is returned to the PDF-AS Library. There, the signature is finally included into the PDF document.

According to the architecture shown in Figure 3.11, PDF-AS functionality can be either accessed by means of a web-based interface or by means of a desktop application. To complete the set of provided tools for the creation of PDF-based signatures, we have proposed a PDF-AS-based PDF-signing solution for mobile end-user devices [Zefferer et al., 2012b]. This way, the underlying concept of PDF-AS has been mapped to modern communication technologies and to the current state of the art. The author of this thesis has considerably contributed to the development of the proposed solution's conceptual design as well as to its implementation.

In order to make PDF-AS functionality available on mobile end-user devices, a distributed approach has been followed. Concretely, our proposed solution comprises a central web application called PDF-AS Wrapper and a mobile PDF-Signing App. This is also shown in Figure 3.12, which illustrates the architecture of the proposed solution. By relying on a distributed approach, required functionality on

[19] https://joinup.ec.europa.eu/software/pdf-as/home
[20] http://webstart.buergerkarte.at/PDF-Over/index.html
[21] http://www.oracle.com/technetwork/java/index-jsp-135475.html

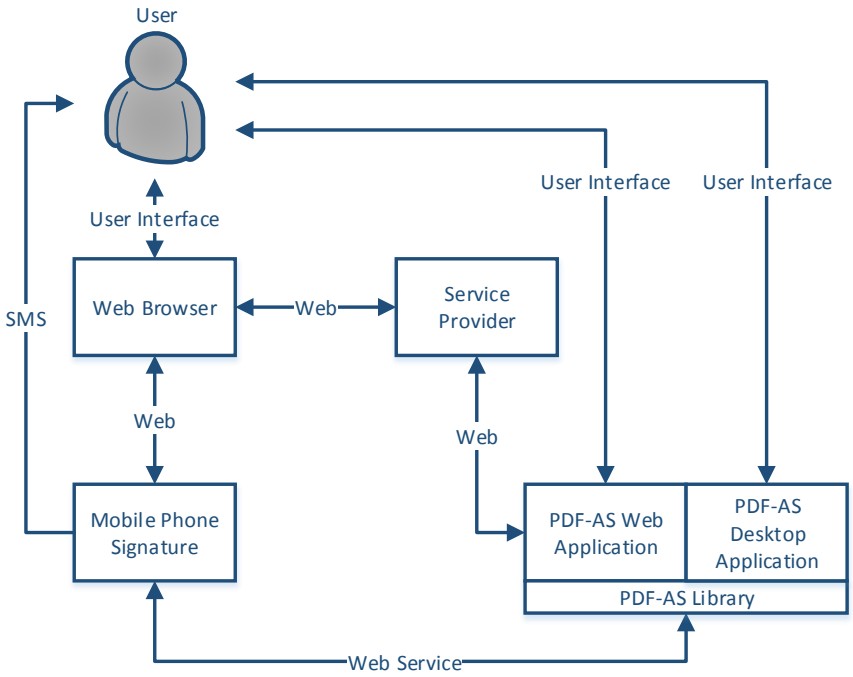

Figure 3.11: PDF-AS support various interfaces and application scenarios.

the client can be minimized. This improves maintainability, as necessary updates of client software on a growing number of different mobile end-user devices and operating systems are minimized.

In general, signing a PDF document using the proposed solution requires the following steps. First, the user starts the PDF-Signing App and selects the PDF document to be signed. The PDF-Signing App then transmits the selected file to the PDF-AS Wrapper, which forwards it to the PDF-AS Web Application. For this purpose, the PDF-AS Wrapper makes use of the Servlet-based interface provided by the PDF-AS Web Application. The PDF-AS Web Application prepares the received PDF file for signing. The data to be signed is sent to the Austrian Mobile Phone Signature. User-authentication requests from the Mobile Phone Signature are forwarded to the PDF-Signing App, which reads required credentials from the user. This way, the Mobile Phone Signature's two-factor authentication is carried out. First, the user is requested to enter his or her secret password. Second, a one-time password is sent to the user via SMS. After entering this one-time password, user authentication is complete and the Mobile Phone Signature creates the requested signature. The created signature is returned to the PDF-AS Web Application, which finally integrates it into the PDF document. The signed PDF document is then returned to the PDF-AS Wrapper, which forwards it to the PDF-Signing App. The PDF-Signing App stores the obtained signed document on the mobile device and notifies the user about the successful signature-creation process.

In order to evaluate its feasibility and applicability, we have implemented the proposed solution shown in Figure 3.12 for the Android platform. This implementation, which we have described in more detail in Zefferer et al. [2012b], shows that the proposed solution facilitates the creation of qualified electronic signatures on smartphones. At the same time, it also reveals a severe shortcoming of the proposed solution. As the Mobile Phone Signature is used with one end-user device, its underlying security concepts and requirements, i.e. the separation of communication channels, cannot be met. This shortcoming has also been identified for the realization of trusted location-based services. Hence, similar conclusions also need to be drawn for the proposed mobile PDF-signing solution. While it is in principle technically feasible to electronically sign PDF documents on mobile end-user devices, the proposed

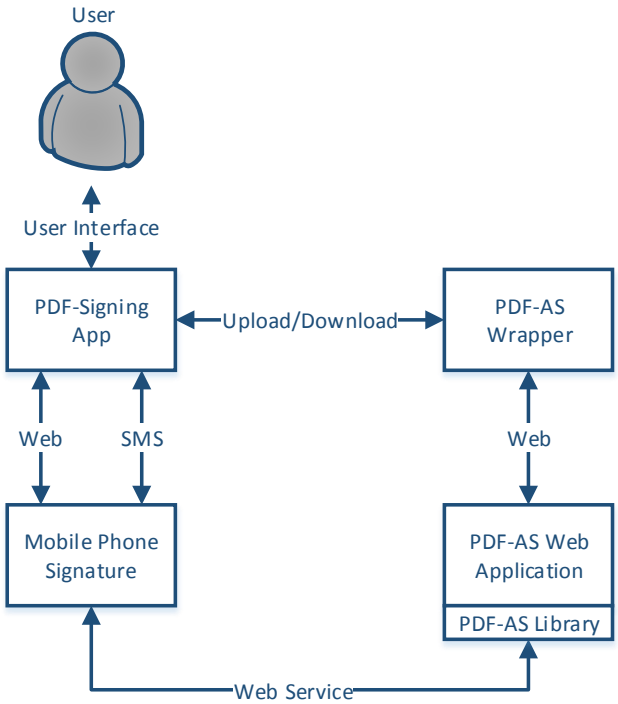

Figure 3.12: The proposed PDF-signing solution relies on components and application scenarios of PDF-AS.

solution cannot achieve a sufficient level of security and is hence difficult to apply in practice.

3.3.2 Signature Verification

One of the key advantages of electronic signatures compared to handwritten signatures is their unambiguous verifiability. As electronic signatures rely on asymmetric cryptographic algorithms, their validity can be reliably determined by cryptographically applying the correct public key. This way, the integrity, authenticity, and non-repudiation of electronically signed content can be verified. While this sounds simple in theory, the implementation of reliable signature-verification mechanisms often turns out to be complex in practice. This is mainly due to complex PKIs, which are required to establish a link between signatories' identities and public keys, and also due to a steadily increasing number of different signature and document formats. We have tackeled the problem of secure, reliable, and efficient signature verification by means of several concrete solutions. With regard to mobile government, special focus has been put on solutions for the verification of electronic signatures on mobile end-user devices. We elaborate on these solutions in the following subsections in more detail.

3.3.2.1 Preparing Signature-Verification Tools for Mobile Access

In practice, the verification of electronically signed documents is complicated by the increasing number of different document and signature formats. For instance, document formats, on which electronic signatures can be applied, include PDF and the Extensible Markup Language (XML). Depending on the respective underlying legal and organizational basis, documents of these formats can be signed according to different signature formats. With regard to European law, especially advanced signature formats such as XML Advanced Electronic Signatures (XAdES), CMS Advanced Electronic Signatures (CAdES), and

PDF Advanced Electronic Signatures (PAdES) are of special relevance. In other application scenarios, alternative signature formats might be relevant.

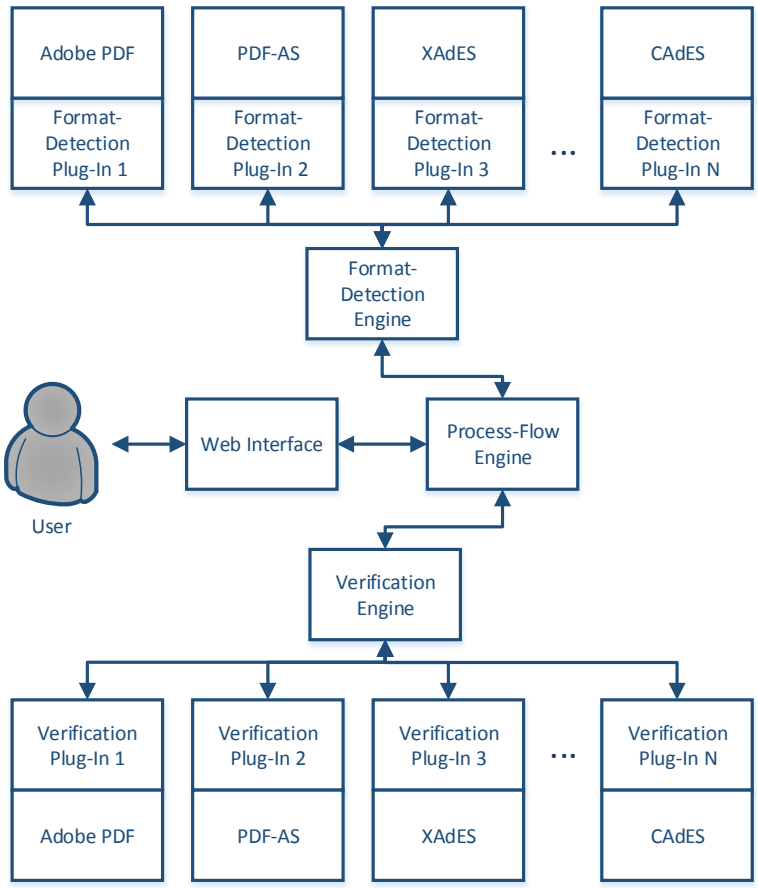

Figure 3.13: The proposed signature-verification tool implements a plug-in-based architecture to assure flexibility.

For each document and signature format, specifics regarding the verification of electronic signatures need to be taken into account. This increases the complexity of signature-verification solutions with a growing number of formats that need to be supported. To hide this complexity from the user, signature-verification tools are usually provided that support verification of various document and signature formats. Due to the various challenges that need to be overcome by these tools, their design and implementation are topics of scientific interest. We have contributed to ongoing research on this topic by proposing and implementing a signature-verification tool that is tailored to requirements of the Austrian e-government infrastructure [Zefferer et al., 2011b]. This tool is in productive operation and operated by the Austrian Regulatory Authority for Broadcasting and Telecommunications (RTR)[22]. It enables users to upload signed documents of various formats to a central web application for verification. After completion of the verification process, results are displayed in the user's web browser. Figure 3.13 shows the architecture behind this solution. The author of this thesis has mainly contributed to the implementation

[22] http://www.signaturpruefung.gv.at/

of this solution and to its continuous enhancement according to emerging requirements.

The architecture of the proposed signature-verification solution comprises basically four core components. A central Process-Flow Engine controls the entire signature-verification process. Required user interactions such as document uploads and the displaying of verification results are implemented by a Web Interface component. The Format-Detection Engine determines the format of documents to be verified. This enables users to upload signed documents of various formats without the need to provide additional meta information. Required signature-verification functionality is finally implemented by the fourth core component, i.e. the Verification Engine.

Figure 3.13 shows that scalability has been a key requirement for the proposed signature-verification tool from the beginning. Both the Format-Detection Engine and the Verification Engine follow a plug-in-based approach, which assures that support for additional document and signature formats can be added easily. Although the underlying architecture provides an efficient level of flexibility regarding different document and signature formats, it is restricted in terms of provided interfaces to external components. Concretely, it provides only one web-based interface, through which documents to be verified can be uploaded. This complicates a use of this solution e.g. in mobile scenarios.

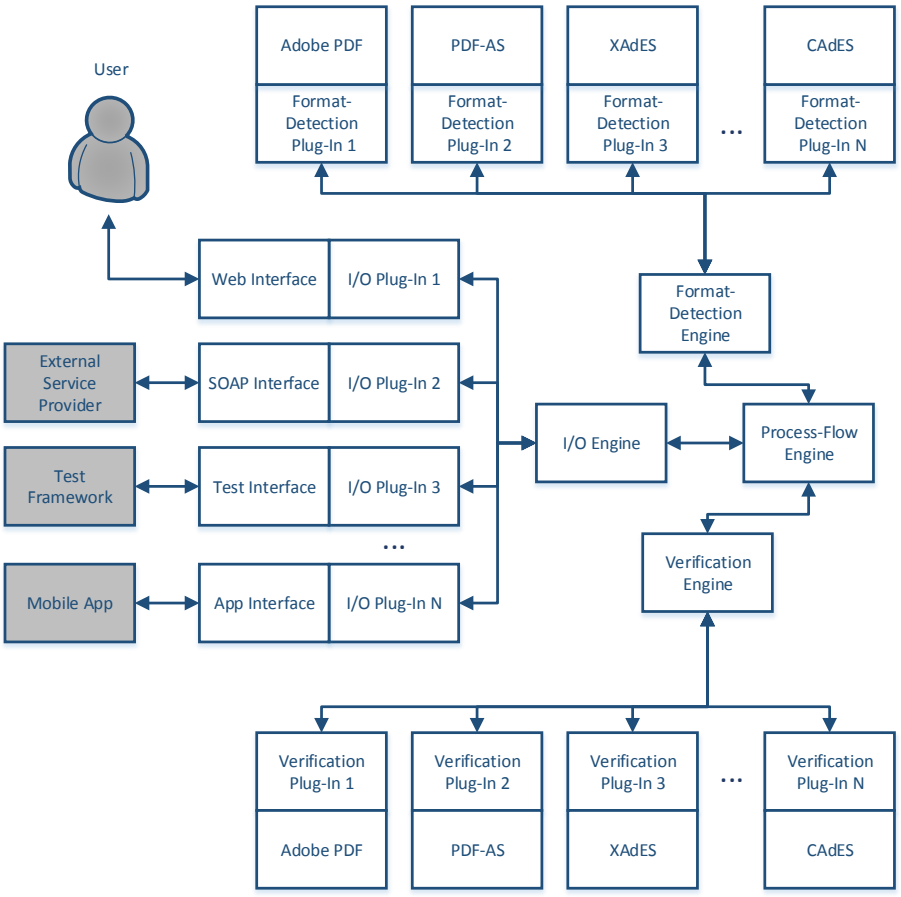

Figure 3.14: The proposed enhanced architecture for signature-verification tools assures that the tool's functionality is provided through various communication channels and technologies.

To address this problem, we have proposed an enhanced architecture for server-based signature-verification tools [Lenz et al., 2013b]. This architecture takes into account the requirement to access the functionality of the signature-verification tool through different communication channels and technologies. This has been achieved by replacing the Web Interface component of the architecture shown in Figure 3.13 by a flexible I/O Engine that follows the same plug-in-based approach as the Format-Detection Engine and the Verification Engine. The resulting enhanced architecture, to which the author of this thesis has considerably contributed, is shown in Figure 3.14.

The applicability of this enhanced architecture has been shown by means of a concrete implementation, which has been based on the initial solution implementing the architecture shown in Figure 3.13. We have also used capabilities of the enhanced solution to make signature-creation functionality available on mobile end-user devices. This is elaborated in more detail in the following section.

3.3.2.2 Signature Verification on Mobile Devices

Smartphones and other powerful mobile end-user devices provide the capability to receive signed documents, e.g. via e-mail. Hence, the provision of signature-verification capabilities on these devices is of special importance. In many cases, signature-verification tools follow a web-based approach and enable users to upload documents to be verified through a web interface. While this is convenient on classical end-user devices, it raises problems e.g. on smartphones, which suffer from limited input and output capabilities.

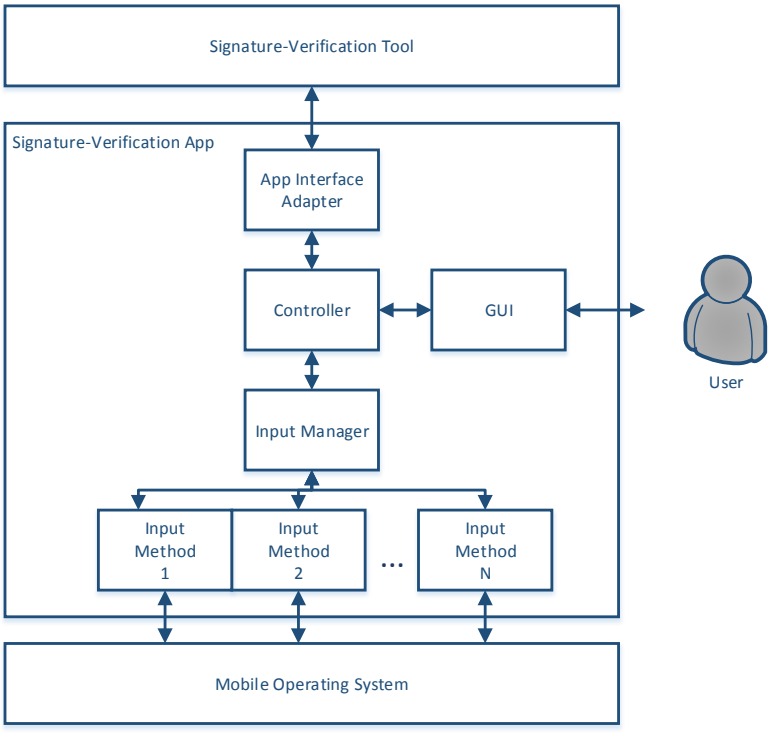

Figure 3.15: The Signature-Verification App relies on functionality provided by the proposed signature-verification tool and supports different input mechanisms to fetch documents from the mobile operating system.

To address this issue, we have proposed and implemented a signature-verification solution for mobile end-user devices [Zefferer et al., 2013a]. While the author of this thesis has formed the concept behind this solution and developed its architecture, related implementation tasks have mainly been undertaken

by a bachelor student.

To take into account limitations of mobile end-user devices, the proposed solution outsources complex operations such as cryptographic computations and PKI integrations to a server component. The client component is limited to functionality requiring user interaction. Furthermore, the proposed solution has been defined on an abstract level to consider specifics of different mobile platforms and operating systems. In particular, the proposed solution is flexible regarding the way documents to be signed are provided by the underlying mobile operating system. The architecture of the proposed solution shown in Figure 3.15 is completely technology-agnostic and can hence be applied to and implemented on arbitrary mobile platforms.

The applicability of the proposed abstract solution has been shown by means of a concrete implementation for the mobile platform Google Android[23]. For the server component, the implementation relies on the enhanced signature-verification tool, whose architecture is shown in Figure 3.15, and which is in productive operation in Austria. Functionality of the client component has been implemented by an Android-based Signature-Verification App. This app enables the user to select documents and to transmit these documents to the server component for verification. After the verification process, the Signature-Verification App receives verification results from the server component and displays them. This way, the developed solution provides smartphone users a convenient app-based opportunity to verify signed documents e.g. received via e-mail. Furthermore, it shows that the verification of electronic signatures is feasible and applicable on current mobile end-user devices.

3.3.3 Lessons Learned

Electronic signatures have been a crucial building block of transactional e-government services for years. Especially in Europe, various countries have deployed electronic-signature solutions, which enable citizens to create legally binding electronic signatures. However, most of these solutions rely on technologies that are hard to be applied on mobile end-user devices. Examples are smart card based solutions, which are common in Europe, but difficult to use e.g. on smartphones.

The incompatibility of smart card based electronic-signature solutions with modern mobile end-user devices is unsurprising, as most of these solutions have been developed and deployed in the pre-smartphone era. At this time, applicability on mobile end-user devices was no requirement. Still, several European countries have deployed mobile signature solutions as an alternative to smart card based approaches. These solutions make use of the user's mobile phone during the signature-creation process in order to avoid the necessity of smart cards. Although these solutions seem to be well-suited at a first glance for a use on mobile end-user devices, their use on mobile end-user devices also raises several challenges. As also mobile signature solutions have been designed for classical end-user devices, their underlying security concepts often become ineffective when being applied on mobile end-user devices.

In summary, it must be concluded that there is currently no satisfying signature solution that can be securely applied on mobile end-user devices. This conclusion can be derived from several concrete electronic signature based solutions. These solutions show that signature verification is feasible on mobile end-user devices but signature creation is usually not. The lack of signature solutions for mobile devices renders the development and use of transactional e-government services on these devices virtually impossible and represents a severe hurdle for transactional m-government in Europe.

3.4 Chapter Conclusions

For several years, technical capabilities of mobile end-user devices have been limited to telephony, text messaging, and simple pre-installed applications. This situation has considerably changed with the in-

[23] http://www.android.com/

troduction of smartphones. Today, smartphones and related mobile end-user devices feature various enhanced technologies and support the installation and use of third-party software by means of mobile apps. It is unsurprising that enhanced capabilities of modern mobile end-user devices are increasingly employed to improve m-government services. In this chapter, capabilities of current mobile end-user devices to meet requirements of m-government services have been assessed in detail. Concretely, security features and limitations of different mobile platforms have been evaluated and strategies to secure applications on mobile end-user devices have been assessed. Finally, opportunities to implement electronic signature based applications on smartphones and related end-user devices have been investigated.

Unfortunately, results of conducted evaluations, assessments, and investigations are sobering for several reasons. First, the heterogeneity of the current mobile-platform ecosystem in general complicates the development of mobile solutions. Second, the development of mobile solutions is also complicated by an inevitable trade-off between security and functionality. Mobile platforms offering third-party applications an enhanced set of functionality usually suffer from reduced security. Similarly, an increased level of security typically goes hand in hand with mobile third-party applications that are limited in terms of functionality. Third, the special characteristics of mobile end-user devices and operating systems narrow possibilities to implement security-enhancing techniques. We have shown by means of concrete implementations that such techniques exist, but that they cannot provide absolute security on mobile end-user devices either. Fourth, mobile applications based on qualified electronic signatures are hardly feasible on current mobile end-user devices. We have shown by means of several concrete solutions that application of existing signature solutions on mobile end-user devices is theoretically feasible but incompatible with underlying security concepts.

From the obtained results, it must be concluded that current smartphones and related mobile end-user devices do not enable the realization of m-government applications that make use of qualified electronic signatures. This, in turn, eliminates the feasibility of transactional m-government services, for which qualified electronic signatures represent a central building block. Design, development, and implementation of an electronic-signature solution that supports the creation of qualified electronic signatures and that is also applicable on mobile end-user devices can hence be identified as inevitable basis of transactional m-government services. At the same time, the current lack of such a solution defines the main problem tackled by this thesis. A solution to the defined problem is developed and proposed in Part II of this thesis.

Part II

Problem Analysis and Proposed Solution

Chapter 4

Approaches Towards Electronic Signatures on Mobile End-User Devices

" Stay committed to your decisions, but stay flexible in your approach."

[Tony Robbins, American Author.]

Documents have always played a central role in scenarios and use cases that require written form. In the course of time, forms and implementations of documents have changed according to the current state of the art. Depending on the particular use case, the authenticity and integrity of a document can be crucial. This applies for instance to documents that represent legally binding contracts or official records. Similar to the form and implementation of documents, also applied means to assure their authenticity and integrity have changed in the course of time. For instance, wax seals have been one of the preferred means in the Middle Ages. In these times, wax seals were used in most cases to authenticate the sender of a document. Seals were also applied to document envelopes, in order to add another level of protection to a document. As opening the envelope was not possible without breaking the seal, unauthorized disclosure and potential modifications of the document were detectable by recipients. This way, wax seals already provided simple means to assure the authenticity and integrity of documents hundreds of years ago. Although applied methods have changed in the course of time, authenticity and integrity still represent key requirements of certain types of documents.

As a more convenient alternative to wax seals, handwritten signatures have evolved early. While the concept of historic wax seals has only partly survived up to now e.g. in the related concept of official stamps, handwritten signatures are still frequently used in various situations of everyday life. Well-known examples are the signing of contracts or the authorization of credit-card payments. Irrespective of the particular use case, a handwritten signature confirms the signer's consent on the signed document's content. This way, the hand-written signature assures the authenticity of the signed document.

Although they have been used for hundreds of years, hand-written signatures suffer from several limitations. First, they do not ensure integrity, as they cannot prevent subsequent modifications on documents or make such modifications detectable. Second, handwritten signatures are neither forgery-proof nor unambiguously verifiable. The computer-aided verification of handwritten signatures has been a topic of scientific interest for several years. An early survey of possible approaches has been provided by Dimauro et al. [2004]. During the past years, several additional solutions have been proposed. Examples are approaches presented by Nguyen et al. [2007], Govindarajan and Chandrasekaran [2011], or de Medeiros Napoles and Zanchettin [2012]. Still, there is currently no absolutely reliable technical mean to verify the validity of handwritten signatures. Another issue is raised by the ongoing digitalization of everyday processes. Nowadays, documents are increasingly created, exchanged, processed, and stored in electronic form. Application of handwritten signatures to electronically processed documents causes

media breaks in electronic procedures and reduces efficiency [Leitold et al., 2009]. For all these reasons, hand-written signatures are far from being the ideal solution to assure the authenticity and integrity of documents.

To overcome limitations of handwritten signatures, the need for an adequate alternative has been identified early. Considering identified limitations of handwritten signatures, electronic signatures have turned out to be a promising solution. Electronic signatures rely on asymmetric cryptographic primitives such as RSA [Rivest et al., 1978] or ECDSA [ANSI, 2005]. Based on these primitives, standards for the electronic signing of electronic document formats have been developed during the past decades. Examples are PAdES [ETSI, 2009b] for PDF documents or XAdES [ETSI, 2009a] for XML-based documents. Based on these standards, several solutions have been developed during the past years that enable the electronic signing of electronic documents and render the use of handwritten signatures unnecessary.

Electronic signatures have several advantages compared to handwritten signatures. Relying on asymmetric cryptographic methods, electronic signatures provide both authenticity and integrity of electronic documents. Furthermore, electronic signatures enable the reliable detection of subsequent modifications of signed content. Due to their electronic nature, they also integrate smoothly into electronic procedures. Finally, electronic signatures are also unambiguously verifiable by technical means. Thus, electronic signatures remove most drawbacks of handwritten signatures.

Due to their various advantages compared to handwritten signatures, electronic signatures have also become a crucial building block of transactional e-government services. Existing solutions surveyed in Part I of this thesis show that this especially applies to countries, in which electronic signatures are legally equivalent to handwritten signatures. Such legal equivalence is for instance given in the European Union, where so-called qualified electronic signatures have the same legal status as handwritten signatures. This has already been defined by the Signature Directive [The European Parliament and the Council of the European Union, 1999] and is now enforced by the eIDAS Regulation. According to this regulation, qualified electronic signatures are a subset of electronic signatures and need to meet several additional requirements. During the past years, qualified electronic signatures have increasingly gained importance for transactional e-government services, as they enable users to provide legally binding written consent in online procedures.

The need for qualified electronic signatures for transactional e-government services raises problems, when these services shall be made available on mobile end-user devices to carry out the transition towards transactional m-government. Existing solutions for the creation of qualified electronic signatures are usually not applicable on modern mobile devices due to their limited capabilities or due to inappropriate security concepts of signature solutions that are tailored to a use on classical devices. For instance, smart card based solutions cannot be applied on mobile devices due to the lack of necessary card-reading devices. Similarly, existing signature solutions relying on mobile phones such as the Austrian Mobile Phone Signature are not applicable on modern mobile end-user devices either, as underlying security concepts demand two separate end-user devices and communication channels. So far, there is no solution available that enables a secure creation of qualified electronic signatures on mobile end-user devices.

Considering the importance of qualified electronic signatures for transactional e-government solutions, provision of a signature solution for mobile end-user devices is crucial. Only if such a solution is available, transactional m-government services can be accessed by and used on mobile end-user devices. As a first step towards development of a signature solution for mobile end-user devices, this chapter identifies possible approaches to implement such solutions. This way, this chapter takes up Tony Robbin's statement quoted at the beginning of this chapter, in which he suggests that even though one shall stay committed to one's decisions, flexibility regarding followed approaches shall be maintained. All identified approaches to implement signature solutions for mobile end-user devices are assessed in terms of feasibility, security, and usability. This way, the most suitable approach for realizing signature solutions for mobile end-user devices is determined. Results obtained from this chapter and conducted assessments will later build the basis for the development of a concrete signature solution.

The structure of this chapter, which bases on own published work [Zefferer, 2015b], reflects the systematic methodology that has been followed to identify and assess different approaches of mobile signature solutions. First, an abstract model of signature solutions for mobile end-user devices is defined. From this abstract model, possible approaches to create electronic signatures on mobile end-user devices are derived. All derived approaches are subsequently assessed in terms of feasibility, security, and usability. From the obtained assessment results, the most promising approach is finally determined.

4.1 Abstract Model

The multitude of technologies available on modern mobile end-user devices usually offers developers different alternatives to solve one and the same problem. In particular, this also applies to the concrete problem of creating qualified electronic signatures. Different approaches to create such signatures on mobile end-user devices are conceivable. In this chapter, the best available approach is determined. For this purpose, a complete set of all possible approaches needs to be identified first.

The identification of possible approaches to create qualified electronic signatures on mobile end-user devices is based on an abstract model. This is advantageous for two reasons. First, an abstract model can be kept as simple as desired by choosing a suitable level of abstraction. A simple model, in turn, enables a systematic and hence complete identification of possible approaches. Second, an abstract model hides implementation-specific and technology-dependent details. This, in turn, reduces the number of possible approaches that need to be considered. This way, reliance on a technology-agnostic and implementation-independent abstract model facilitates the systematic identification of possible approaches to create electronic signatures on mobile end-user devices.

To develop the required abstract model, existing signature-creation models are surveyed first. This survey considers both related standards and scientific publications. In addition, a set of design principles is defined. Based on obtained results of the conducted survey and on the defined design principles, an abstract technology-agnostic and implementation-independent model for signature-creation solutions is finally defined and proposed.

4.1.1 Existing Signature-Creation Models

The definition of models for signature-creation processes is a frequently followed approach in both scientific literature and standardization works. In most cases, defined models use similar notations for involved parties and components. This applies to most models irrespective of their level of abstraction. In the European context, which is especially relevant for the special case of qualified electronic signatures, terms and notations for involved parties and components are usually aligned with definitions given in the EU Signature Directive [The European Parliament and the Council of the European Union, 1999], which represents the legal basis for electronic signatures in the European Union. The EU Signature Directive will soon be replaced by the EU eIDAS Regulation [The European Parliament and the Council of the European Union, 2014]. The eIDAS Regulation reuses established terms and notations defined by the EU Signature Directive. Furthermore, the eIDAS Regulation adds further definitions of parties and components involved in signature-creation processes and related use cases. In the near future, the eIDAS Regulation will assume the role of the Signature Directive and serve as a basis for common terms and notations related to electronic signatures in Europe.

Besides legislations, also standards define standing terms of parties and components involved in processes related to electronic signatures. For instance, ETSI[1] provides a set of standards related to electronic signatures [ETSI, 2014b]. Examples are the ETSI standards for CAdES [ETSI, 2013], PAdES [ETSI, 2009b], and XAdES [ETSI, 2009a]. These standards focus on the definition of different formats for electronic signatures. Even though they do not define concrete signature-creation models, these

[1] http://www.etsi.org/

standards still provide definitions for involved parties and components. In particular, they identify a set of major parties involved in business transactions supported by electronic signatures. Identified major parties include the Signer, the Verifier, Trusted Service Providers (TSPs), and the Arbitrator. Depending on the concrete standard, involved TSPs are also listed. Still, these standards identify involved parties on a rather abstract level only, hardly taking into account details regarding their concrete realization or implementation. Hence, the mentioned standards provide a solid basis for the identification of relevant parties and components involved in signature-creation processes, but do not assemble these components to concrete models of signature-creation solutions.

Concrete models of signature-creation processes and solutions are for instance provided by documents published by the European Committee for Standardization (CEN)[2]. For instance, the CEN Workshop Agreement (CWA) 14169 on Secure Signature Creation Devices Evaluation Assurance Level (EAL) 4+ [CEN, 2004a] defines security requirements for Secure Signature Creation Devices (SSCDs). For this purpose, core components of signature-creation solutions that include an SSCD are identified and assembled to a simple model. As CWA 14169 mainly focuses on the SSCD, other relevant parties and components involved in signature-creation solutions are only briefly sketched in this document. More details on such components are provided and considered by a model defined by the CEN Workshop Agreement 14170 on security requirements for signature creation applications [CEN, 2004b]. This document introduces a functional model for signature-creation solutions relying on an SSCD and a so-called Signature Creation Application (SCA), which provides other entities access to the SSCD. Similar to related standards, CWA 14170 relies on common notations, terms, and definitions of involved parties and components. While other standards mainly use these terms on rather abstract levels, CWA 14170 assembles identified parties and components to provide a comprehensive functional model for signature-creation solutions. A slightly modified version of the model introduced by CWA 14170 is also used in the draft version of ETSI's conformity assessment for signature creation and validation [ETSI, 2014a]. There, the rather complex model from CWA 14170 is reduced to a few components including the Signer, the SCA, and the SSCD. In addition, exchanged information between these components is also considered by this model.

Identification, definition, and development of models of signature-creation solutions are not limited to relevant standards. In addition, the development of signature-creation models has also been a topic of interest for the scientific community. In several scientific publications, different models have been used to systematically discuss concepts related to electronic signatures. For instance, Leitold et al. [2002] have introduced the security architecture of the Austrian Citizen Card concept with the help of a simplified model of a signature-creation system. This model identifies and arranges components of smart card based signature solutions, such as a PIN pad, an SSCD, and a card-acceptor device. To serve the purpose of the presented paper, Leitold et al. [2002] use a rather implementation-specific model that is tailored to the special use case of smart card based solutions. In contrast, Arnellos et al. [2011] use a more abstract model of a signature-creation solution. This model is used to discuss the structural reliability of signed documents on a semantic level. These two contrary examples show that for scientific work on electronic signatures similar considerations apply as for published standards on this topic: the employed model heavily depends on the context and the use case, for which it has been defined and developed. Still, most models share several similarities and are based on common definitions, terms, and notations. Common terms are usually provided by legal foundations such as the EU Signature Directive [The European Parliament and the Council of the European Union, 1999] or the eIDAS Regulation [The European Parliament and the Council of the European Union, 2014], and by related standards.

The conducted survey on existing models for signature-creation solutions shows that there already exists a set of established terms and notations for involved parties and components. Most models that are employed in scientific work and standards rely on these common terms and notations. Nevertheless, employed models differ in various aspects depending on the context, in which they are used. This has

[2] https://www.cen.eu

led to the development of various models that differ in the set of considered parties and components, and also in the granularity, in which parties and components are modeled. Hence, it can be concluded that there is no universally valid and applicable model for all contexts. Instead, an adequate model needs to be developed for each specific use case. This way, used models can be tailored to specific needs and consider all relevant aspects.

4.1.2 Design Principles

Following the results of the conducted survey, a specific model for signature-creation solutions needs to be defined for each context. Hence, a specific model also needs to be defined for the provision of a signature solution for mobile end-user devices. For this purpose, two aspects are considered. First, the defined model is based on the findings of the conducted survey. Second, a set of design principles is taken into account. These design principles assure that the defined model is suitable for the given context. Concretely, the following design principles are taken into account:

- **Compliance with established notations and definitions:** The model shall rely on established terms and notations. Furthermore, it shall rely on established core components that are defined in related legal frameworks and standards. This is reasonable in order to improve its comprehensibility and to guarantee that it can be easily set in relation to other existing models.
- **Reliance on existing models:** The model shall rely as much as possible on existing and approved models. This way, their positive characteristics shall be inherited.
- **Simplicity:** The model shall be kept as simple as possible. This way, unnecessary complexity can be avoided and focus can be kept on relevant aspects.
- **Completeness:** The model shall include all relevant parties and components that are involved in a signature-creation process. This is reasonable, as the model will later be used to derive different approaches for mobile signature solutions. Hence, completeness is crucial to assure that all possible approaches can be derived.

Based on these design principles, an abstract model for mobile signature solutions is proposed. It is later used to systematically derive possible approaches to create qualified electronic signatures on mobile end-user devices. The proposed abstract model is introduced and discussed in the following section in more detail.

4.1.3 Proposed Model

A suitable model needs to be chosen, in order to enable a systematic identification of possible approaches for the creation of qualified electronic signatures on mobile end-user devices. Unfortunately, existing models are usually tailored to specific use cases and must hence be regarded as inappropriate for this purpose. Therefore, the alternative model shown in Figure 4.1 is proposed.

The proposed model has been defined according to the predefined design principles. To satisfy the design principle regarding reliance on existing models, the model has been based on CWA 14170 [CEN, 2004b]. This way, it builds on an elaborate and approved basis. The model also meets the design principle regarding compliance with established notations and definitions. Where possible and reasonable, its components have been named according to terms and notation used in related legal frameworks and standards. New notations have only been chosen for components that are unique for the proposed model. Such components have been added mainly for two reasons. First, new components have been added to achieve completeness according to the above-defined design principle. Second, for the sake of simplicity, which has also been defined as relevant design principle, several components defined by other

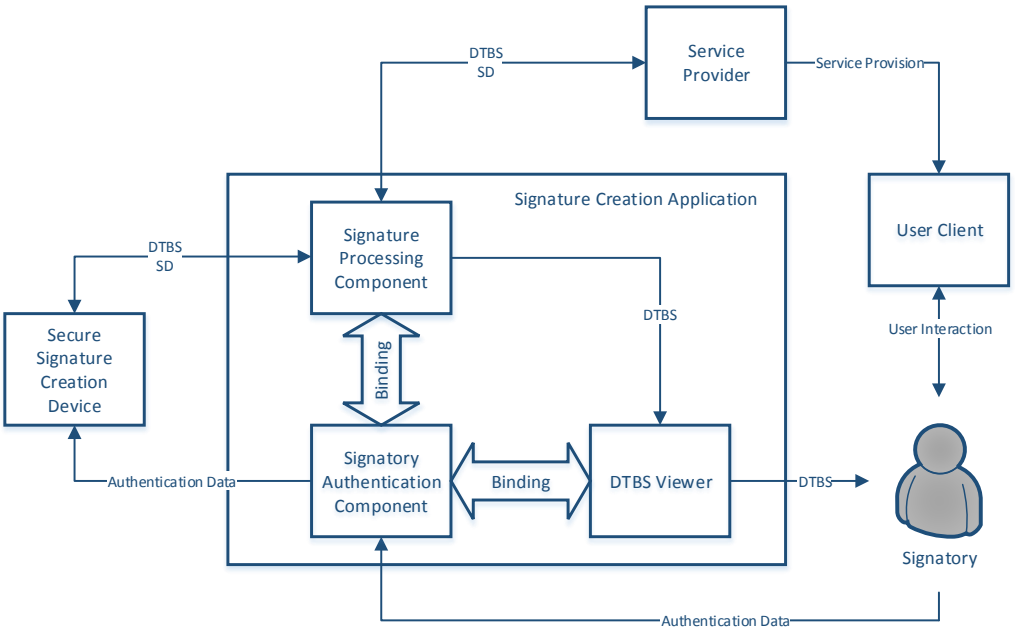

Figure 4.1: The proposed abstract model identifies involved parties and components of signature-creation solutions and defines basic interfaces between these components.

models have been combined to more generic components. To avoid confusion with components of other models, these combined components have been assigned with new names. This way, the proposed model complies with all predefined design principles.

In general, the model shown in Figure 4.1 consists of the Signatory and four top-level components. The Signatory is the only non-technical entity. According to the EU Signature Directive, Signatory *'means a person who holds a signature-creation device and acts either on his own behalf or on behalf of the natural or legal person or entity he represents'* [The European Parliament and the Council of the European Union, 1999]. Following this definition, the Signatory is the user who aims to create an electronic signature. Apart from the Signatory, four top-level components can be identified. These top-level components are the SSCD, the SCA, the Service Provider, and the User Client. SSCD and SCA are established terms for key components of signature-creation models. The components Service Provider and User Client have been added, as they will play a relevant role in the subsequent assessment of possible signature-creation approaches. In the following subsections, the four top-level components of the proposed model are introduced in detail.

4.1.3.1 Secure Signature Creation Device

SSCDs represent a crucial component for the creation of qualified electronic signatures. The need for SSCDs to create qualified electronic signatures is defined by the Signature Directive [The European Parliament and the Council of the European Union, 1999]. The concept of SSCDs is also used by the eIDAS Regulation. However, this regulation introduces the term QSCD for this purpose. As the term SSCD is still more established, this term is used throughout this thesis.

A model of an SSCD, which also illustrates its functionality as defined in Annex III of the EU Signature Directive, is for instance provided in CWA 14169 [CEN, 2004a]. The SSCD securely stores the Signatory's cryptographic signing key, i.e. the so-called Signature-Creation Data (SCD), and cryptographically computes electronic signatures using this key. As the Signatory's signing key must not be extractable from the SSCD, the SSCD is the only component that can access and use this key for crypto-

graphic operations. Additionally, the SSCD must implement appropriate means to protect the Signatory's signing key from unauthorized access. These means must guarantee that access to this key and hence to the capability to create electronic signatures using this key are limited to the legitimate Signatory. In general, the SSCD needs two inputs for the creation of an electronic signature. First, the SSCD needs to be supplied with the Data To Be Signed (DTBS). Second, the SSCD needs to be supplied with the correct authentication data provided by the Signatory in order to authorize creation of an electronic signature. If both inputs are provided accordingly, the SSCD computes an electronic signature over the provided DTBS and returns the result of this signing operation, i.e. the Signed Data (SD). Relevant interfaces of the SSCD are also shown in Figure 4.1.

4.1.3.2 Signature Creation Application

The SCA is the central component of the proposed model. This becomes apparent from Figure 4.1. The SCA connects the Signatory, the SSCD, and the Service Provider. The central role of SCAs also becomes apparent from CWA 14170 [CEN, 2004b], which defines security requirements for this component. According to CWA 14170, an SCA contains various subcomponents that cover its functionality. For the sake of simplicity, the model shown in Figure 4.1 uses a simplified representation of the SCA's functionality and defines only three subcomponents for the SCA. These are the Signature Processing Component, the Signatory Authentication Component, and the DTBS Viewer. These three subcomponents combine the functionality of multiple subcomponents of SCAs as defined in CWA 14170. This way, the proposed model assembles the SCA from only three subcomponents, while the entire functionality of an SCA is still covered.

Figure 4.1 shows that the SCA is the only component with a direct interface to the SSCD. Hence, the SCA needs to provide means to supply the SSCD with the required inputs for signature-creation processes. In particular, the SCA needs to supply the SSCD with DTBS and with authentication data. For provision of DTBS, the SCA's Signature Processing Component plays a central role. The Signature Processing Component receives DTBS from the Service Provider and forwards these data to the SSCD. After completion of the signing process in the SSCD, the Signature Processing Component receives the SD from the SSCD. The Signature Processing Component forwards the SD to the Service Provider. Depending on the chosen implementation, the Signature Processing Component might also be responsible for doing preprocessing or reformatting of DTBS received from the Service Provider and of the SD received from the SSCD. This is typically the case, if the communication interface to the Service Provider bases on different communication protocols and standards than the communication interface to the SSCD.

As the SCA is the only component with a direct interface to the SSCD, the SCA is also responsible for supplying the SSCD with required authentication data. This functionality is covered by the SCA's Signatory Authentication Component. This component has a direct interface to the Signatory. Through this interface, the Signatory Authentication Component requests authentication data from the Signatory and forwards these data to the SSCD. Similar to the Signature Processing Component, also the Signatory Authentication Component might be required to reformat authentication data entered by the Signatory. This can be necessary to adapt authentication data entered by the Signatory to the technology used for communication between the SCA and the SSCD.

The DTBS Viewer represents the third subcomponent of the SCA. Through the DTBS Viewer, the SCA gives the Signatory the opportunity to check the DTBS. This allows the Signatory to review the data that is about to be signed before providing the required authentication data that finally enables the signature-creation process in the SSCD. Depending on the concrete implementation and used data formats, the DTBS Viewer might be required to preprocess the received DTBS in order to transform it into a human-readable format.

The Signature Processing Component, the Signatory Authentication Component, and the DTBS Viewer basically cover the functionality of the SCA. However, to achieve the intended functionality,

the SCA additionally needs to assure appropriate bindings between its subcomponents. This is also illustrated in Figure 4.1. In total, two bindings need to be guaranteed. First, the SCA needs to assure that the DTBS displayed to the Signatory by the DTBS Viewer corresponds to the DTBS sent to the SSCD for signature creation. Second, there must also be an unambiguous binding between the DTBS and the authentication data. This means that there must be an unambiguous and verifiable link between the DTBS and the authentication data that authorizes signature creation on these DTBS.

4.1.3.3 Service Provider

As indicated by its name, the Service Provider provides some kind of service to the Signatory. In the context of signature-creation processes, the proposed model shown in Figure 4.1 defines two additional roles for the Service Provider. In its first role, the Service Provider defines the data to be signed. At some point during service provision, the Signatory is requested to create a qualified electronic signature on data defined by the Service Provider. Hence, the Service Provider represents the source of the DTBS. In its second role, the Service Provider represents the final destination of SD. According to the proposed model, SD are forwarded to the Service Provider by the SCA after successful signature-creation processes. Combining source of the DTBS and destination of the SD into one component distinguishes the model shown in Figure 4.1 from other established models of signature-creation solutions. These models usually define separate components for the creation of the DTBS and for the consumption and further processing of the SD. Entities that act as recipients of the SD are usually referred to as Relying Party (RP) in these models. For the given context, distinguishing between creators of the DTBS and recipients of the SD is however not necessary. Hence, for the sake of simplicity both functionalities are assumed to be covered by a single component. This component is simply denoted as Service Provider, in order to avoid confusion with components of other signature-creation models.

4.1.3.4 User Client

The User Client represents the fourth top-level component of the proposed model. The User Client is used by the Signatory to consume services provided by the Service Provider. Similar to the Service Provider, the User Client is no integral component of the signature solution itself. However, the User Client becomes relevant when analyzing different approaches to create electronic signatures on mobile end-user devices. Depending on the actual implementation of the Service Provider and the User Client, the roles of these two components can also be combined and implemented by one component. This is for instance the case with typical mobile apps. However, there are also scenarios that require the Service Provider and the User Client to be regarded by separate components. Examples are for instance web-based services, where the Service Provider is implemented by a central web application, whereas the User Client is implemented by a web browser. As subsequent assessments of possible approaches to implement the proposed model require a separation of the two components, the model shown in Figure 4.1 defines the Service Provider and the User Client as two separate components.

4.1.4 Limitations

The proposed model represents a common abstract basis for the systematic identification of possible approaches to create electronic signatures on mobile end-user devices. Following predefined design principles, the proposed model has been kept as simple and as minimalistic as possible. For this purpose, in particular the following two aspects of electronic-signature solutions are not covered by the model shown in Figure 4.1:

- **Signature validation:** Electronic-signature solutions must provide means for both signature creation and signature validation. However, the proposed model puts focus on signature creation only.

Even though the validation of electronic signatures is without doubt an important issue, validation aspects do usually not directly influence design decisions of signature-creation solutions. Validation services face different challenges that are only marginally influenced or caused by design or implementation decisions of the corresponding signature-creation solution. For these reasons and also for the sake of simplicity and clarity, signature validation has not been considered for the identification of relevant parties and components. The conceptual separation of signature creation and validation is actually common practice. For instance, also ETSI follows this approach in Draft SR 019 020 (Rationalised Framework of Standards for Advanced Electronic Signature in Mobile Environments)[3], for which signature-creation and signature-validation scenarios are clearly separated.

- **Registration:** In general, signature-creation solutions typically consist of two major phases: the registration phase and the usage phase. In the registration phase, signature-creation functionality is activated and made available for a specific Signatory. Activation of signature functionality typically includes the generation of a cryptographic key-pair, the definition of an authorization code that protects access to the private part of this key pair, and issuing of the Signatory's qualified electronic certificate. This certificate binds the created cryptographic key to the Signatory's identity. In the subsequent usage phase, the activated signature functionality can then be used to create electronic signatures. Typically, the registration phase needs to be run once by each Signatory in order to enable the creation of electronic signatures during the usage phase. After successful registration, the usage phase can be repeated arbitrary times. As the registration phase needs to be run only once, requirements of the registration process can also be covered by organizational means. This does not apply to the usage phase. The usage phase must be designed and implemented in a way that secure and usable signature-creation processes can be conveniently conducted by the Signatory arbitrary times. This implies that requirements of this phase mainly need to be covered by technical means. For these reasons, the proposed model focuses only on components that are required during the usage phase, i.e. during signature-creation processes.

Due to its abstract, implementation-independent, and technology-agnostic nature, the proposed model covers a broad spectrum of possible approaches to create electronic signatures. Different approaches can be derived from the proposed model by varying the concrete implementation of the model's identified components. Hence, each approach to create electronic signatures corresponds to a concrete implementation variant of the proposed model. In the following section, a systematic approach is followed to derive all possible implementation variants from the proposed abstract model.

4.2 Implementation Variants

The proposed abstract model for signature-creation solutions identifies components and relevant interfaces between these components. Due to its technology-agnostic and implementation-independent character, the model does however not make any assumptions regarding their concrete implementation. In practice, various components can be implemented differently, yielding different implementation variants of one and the same abstract model. In this section, possible implementation variants are derived systematically by combining possible implementations of all identified components.

In general, components can either be implemented locally on the Signatory's mobile end-user device, or remotely by means of a server component. To identify possible implementation variants, all identified components are split into two categories. The first category contains components that need to be implemented on the Signatory's local mobile end-user device in any case. For instance, this applies to the DTBS Viewer. As the Signatory needs to directly interact with the DTBS Viewer, this component needs to be implemented on the Signatory's mobile end-user device. In contrast, the second category contains

[3] http://docbox.etsi.org/ESI/Open/Latest_Drafts/sr_019020-v004-AdES-in-mobile-envt_STABLE-DRAFT.zip

components that can be implemented either locally on the Signatory's mobile end-user device, or remotely by a server component. This applies for instance to the Service Provider. The Service Provider can either be implemented remotely, e.g. by a web application, or locally, e.g. by a mobile app.

Apparently, the key classification criterion is the Signatory's need to interface the respective component. All components of the abstract signature-creation model that need to provide a user interface to the Signatory must be implemented locally. All other components are not subject to such restrictions and can be implemented either locally or remotely. According to this classification criterion, three components can be identified that can at least theoretically be implemented either locally or remotely: the Service Provider, the Signature Processing Component, and the Secure Signature Creation Device. This yields eight possible combinations of locally and remotely implemented components. These combinations are summarized in Figure 4.2.

ID	Service Provider	Signature Processing Component	SSCD	Implementation Variant
1	Local	Local	Local	*Implementation Variant A: The Classical Approach*
2	Local	Local	Remote	*Implementation Variant B: The Remote-SSCD Approach*
3	Local	Remote	Local	*Implementation Variant C: The SIM Approach*
4	Local	Remote	Remote	*Implementation Variant D: The Server-HSM Approach*
5	Remote	Local	Local	*Implementation Variant A: The Classical Approach*
6	Remote	Local	Remote	*Implementation Variant B: The Remote-SSCD Approach*
7	Remote	Remote	Local	*Implementation Variant C: The SIM Approach*
8	Remote	Remote	Remote	*Implementation Variant D: The Server-HSM Approach*

Figure 4.2: The applied systematic identification process yields four general implementation variants for signature-creation solutions.

The last column of the table illustrated in Figure 4.2 shows that the eight possible combinations can be reduced to four general implementation variants. For each of the four implementation variants, the component Service Provider can be implemented either locally or remotely. Thus, for each implementation variant, this component needs to be split into the two components Local Service Provider and Remote Service Provider. The four resulting implementation variants, which all consider both the Local Service Provider and the Remote Service Provider, are presented and discussed in the following subsections in more detail.

4.2.1 Implementation Variant A: The Classical Approach

Implementation Variant A can also be referred to as the Classical Approach, since it basically resembles the architecture of typical smart card based signature solutions for classical end-user devices. These solutions make use of smart cards to provide the functionality of the SSCD. Furthermore, these solutions typically rely on software being installed on the user's local system. This software enables service providers access to the locally connected smart card, reads required credentials from the user, and displays relevant information during signature-creation processes. Signature solutions for mobile end-user

devices that rely on Implementation Variant A follow a similar approach. As shown in Figure 4.2, such solutions implement both the SSCD and the Signature Processing Component locally on the Signatory's end-user device. From the abstract model and the design decision to realize both the SSCD and the Signature Processing Component locally, the general architecture of Implementation Variant A can be derived. This architecture is shown in Figure 4.3.

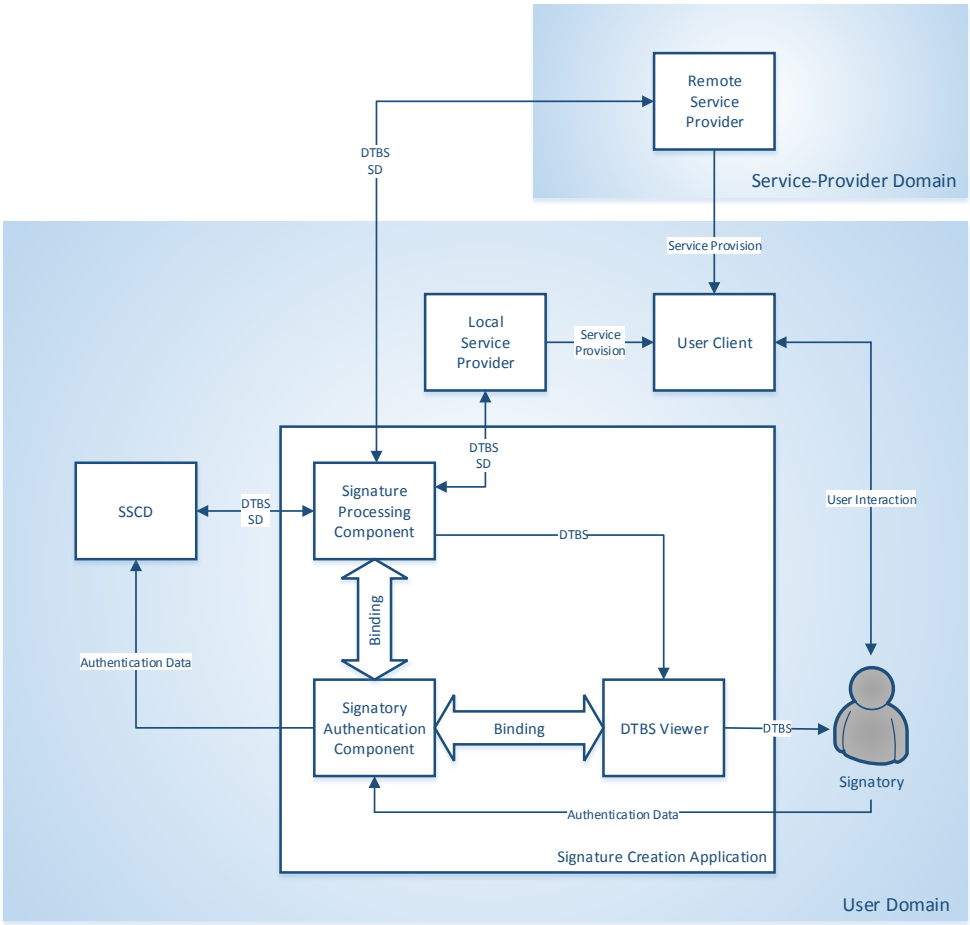

Figure 4.3: Implementation Variant A: The Classical Approach comprises solutions that implement both the SCA and the SSCD locally.

In addition to the abstract model shown in Figure 4.1 on page 96, the architecture of Implementation Variant A shown in Figure 4.3 defines different domains. All parties and components of the proposed model are assigned to one of these domains. Concretely, the architecture of Implementation Variant A defines the two domains User Domain and Service Provider Domain. The Service Provider Domain contains only the Remote Service Provider, which can for instance be realized by means of a web application or web service hosted on a central web server. All other roles and components of the proposed model are assigned to the User Domain. Hence, the SSCD, the Local Service Provider, the User Client, and the Signature Creation Application including all its subcomponents are implemented locally in the User Domain.

4.2.2 Implementation Variant B: The Remote-SSCD Approach

Implementation Variant B breaks with the strict local paradigm of the Classical Approach. In contrast to the Classical Approach, Implementation Variant B implements the SSCD remotely. Accordingly, Im-

plementation Variant B can also be referred to as the Remote-SSCD Approach. Removing the SSCD from the User Domain renders SSCD realizations on the local end-user device unnecessary. This improves the feasibility of signature solutions for mobile end-user devices with limited capabilities. As shown in Figure 4.2, the location of the SSCD is the only difference between the Classical Approach and the Remote-SSCD Approach. The Signature Processing Component is implemented locally by both approaches. The general architecture of Implementation Variant B, i.e. the Remote-SSCD Approach, is shown in Figure 4.4.

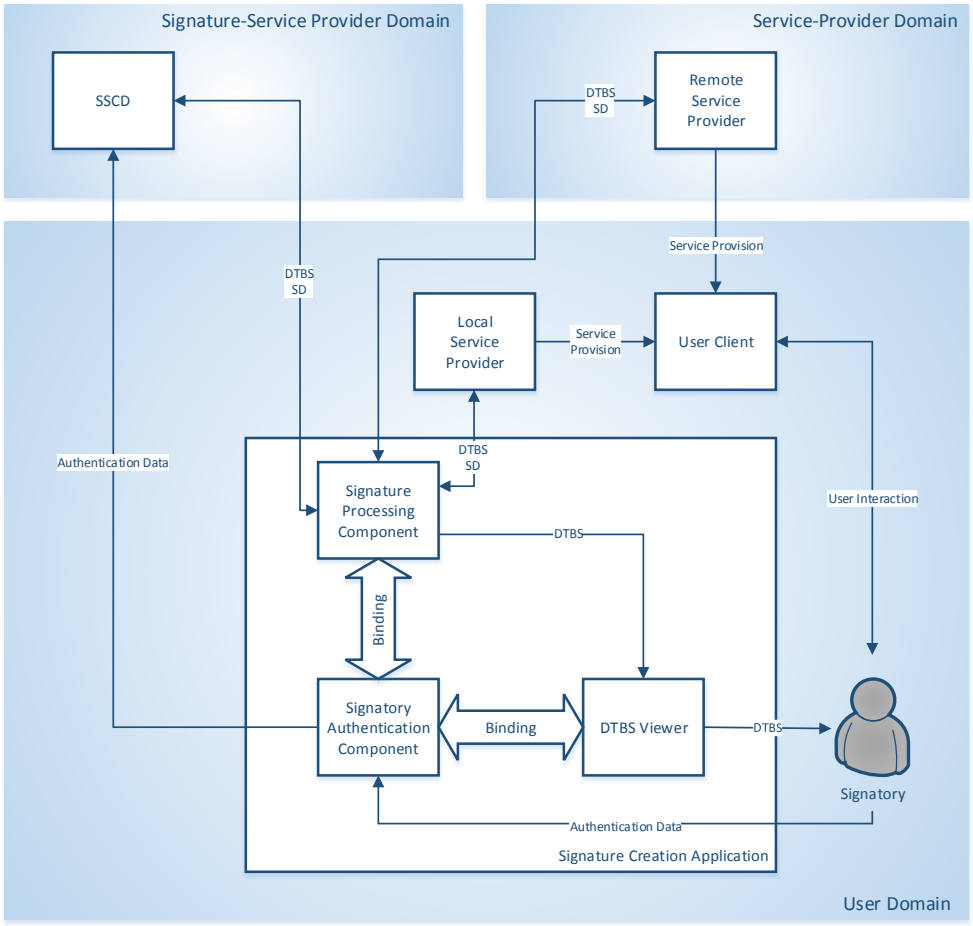

Figure 4.4: Implementation Variant B: The Remote-SSCD Approach comprises solutions that implement the SCA locally and the SSCD remotely.

Due to the remote nature of the SSCD, an additional remote domain needs to be defined. Thus, the architecture shown in Figure 4.4 defines the three domains User Domain, Service Provider Domain, and Signature-Service Provider Domain. Similar to the Classical Approach, the only component located in the Service Provider Domain is the Remote Service Provider. Also the newly defined Signature-Service Provider Domain contains only one component, i.e. the SSCD. All other entities including the SCA, the Local Service Provider, the User Client, and the Signatory are again located in the User Domain.

In contrast to the Classical Approach, which basically resembles smart card based signature solutions, the architecture of the Remote-SSCD Approach can be less frequently found in practice. So far, only few providers of remote SSCDs exist. The most popular example is probably the Austrian Mobile Phone Signature, which relies on a remote SSCD for creating qualified electronic signatures on behalf of Signatories. However, this solution also implements the Signature Processing Component remotely. Hence, the architecture shown in Figure 4.4 does not fully apply to the Austrian Mobile Phone Signature.

Other solutions that rely on remote components to implement cryptographic functionality are provided by some cloud-service providers. For instance, Amazon offers a product called Amazon CloudHSM[4], which basically provides cryptographic services via a cloud-based hardware security module. However, most of these solutions do not provide functionality for the creation of qualified electronic signatures yet. Hence, there are hardly any existing solutions that rely on the Remote-SSCD Approach shown in Figure 4.4.

The special characteristic of the Remote-SSCD Approach raises several challenges when further developing this approach towards a concrete signature-creation solution. One of these challenges is due to the local separation of the SCA and the SSCD. This raises the need for secure and reliable communication between these two locally dispersed components. Secure and reliable communication channels must be provided for DTBS and SD exchanged between the Signature Processing Components and the SSCD, as well as for authentication data that is sent from the local Signatory Authentication Component to the remote SSCD. Another challenge is imposed by the remote nature of the SSCD itself. Signature-creation solutions that support creation of qualified electronic signatures need to assure that the Signatory maintains sole control over personal signature-creation data, i.e. cryptographic signing keys. This requirement is implicitly met by e.g. smart card based approaches, as the smart card remains under physical control of the Signatory all the time. Sole control over cryptographic signing keys is assured in a similar way by the Classical Approach. There, the Signatory is in physical possession of the SSCD, which is part of the Signatory's mobile end-user device. Considering the slightly different architecture of the Remote SSCD-Approach, the situation is sort of special. Due to its remote nature, the SSCD is not under physical control, i.e. in possession, of the Signatory. Thus, alternative means need to be applied in order to meet the requirement for sole control over the Signatory's signature-creation data. Such alternative means are in principle feasible. This has for instance been shown by the Austrian Mobile Phone Signature, which relies on a remote SSCD and is still capable to create qualified electronic signatures. The underlying concept of this solution, which assures sole control over signature-creation data even in remote signing scenarios, has been introduced and discussed by Orthacker et al. [2010].

4.2.3 Implementation Variant C: The SIM Approach

The Remote-SSCD Approach shows that realizing components of signature-creation solutions in a remote domain is an attractive way to overcome limitations of mobile end-user devices. Also Implementation Variant C relies on this approach and implements the Signature Processing Component remotely. However, in contrast to the Remote-SSCD Approach, the SSCD is realized locally on the mobile end-user device. The resulting architecture of Implementation Variant C is shown in Figure 4.5. Similar to the Remote-SSCD Approach, relevant roles and components are assigned to the three domains User Domain, Service Provider Domain, and Signature-Service Provider Domain. The Service Provider Domain again contains only the Remote Service Provider. In contrast to the Remote-SSCD Approach, the Signature-Service Provider Domain now contains the Signature Processing Component instead of the SSCD. Hence, a remote service is provided that allows service providers to request signature-creation functionality provided by the SSCD. The SSCD itself is assigned to the User Domain and implemented locally on the Signatory's mobile end-user device.

There are several signature-creation solutions that rely on the architecture shown in Figure 4.5. Most of them rely on mobile phones. These solutions make use of special SIMs to implement the SSCD in the User Domain. Furthermore, they rely on a central service hosted either by the Signatory's MNO or by a separate service provider to implement the Signature Processing Component. SIM-based solutions that follow the architecture shown in Figure 4.5 are in productive operation e.g. in Estonia[5] or Finland[6]. Due to their similarity to SIM-based signature solutions, Implementation Variant C can also be referred to as

[4] http://aws.amazon.com/cloudhsm/
[5] http://mobiil.id.ee/
[6] https://www.gemalto.com/govt/customer_cases/finland_mobile.html

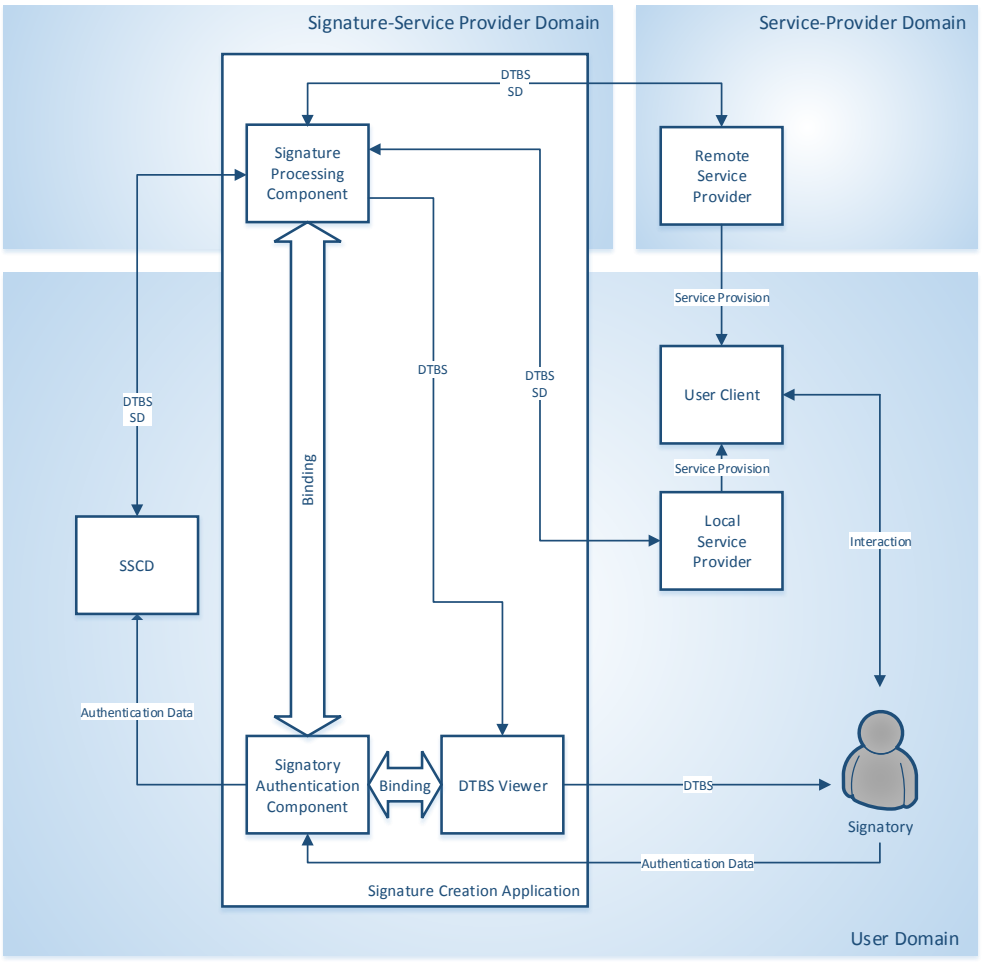

Figure 4.5: Implementation Variant C comprises solutions that implement the SCA partly remotely and the SSCD locally.

the SIM Approach.

Signature solutions that follow the SIM Approach need to overcome several challenges. Similar to the Remote-SSCD Approach, the Signature Processing Component and the SSCD are locally dispersed and implemented in different domains. Hence, secure and reliable communication channels between these locally dispersed components need to be provided. Another challenge that arises with the architecture shown in Figure 4.5 concerns the SCA. In contrast to the architectures of the Classical Approach and the Remote-SSCD Approach, the architecture shown in Figure 4.5 implements subcomponents of the SCA in different domains. Hence, subcomponents of the SCA are locally dispersed and implemented by different entities. This raises the need for means to assure the required binding between the three subcomponents of the SCA across different domains.

4.2.4 Implementation Variant D: The Server-HSM Approach

Implementation Variant D combines the approaches followed by the Remote-SSCD Approach and the SIM Approach. These two approaches implement either the SSCD or the Signature Processing Component remotely. In contrast, both components are implemented remotely according to Implementation Variant D. This way, Implementation Variant D basically reflects the architecture of the Austrian Mobile Phone Signature, which consists of a central web-based service that assumes the role of the Signature

Processing Component. The SSCD is implemented by means of an HSM that is attached to this remote service. The DTBS Viewer and the Signatory Authentication Component are implemented by simple web forms, which can be integrated easily into web-based applications that aim to integrate signature-creation functionality. Due to its similarity with the Austrian Mobile Phone Signature, Implementation Variant D can also be referred to as the Server-HSM Approach.

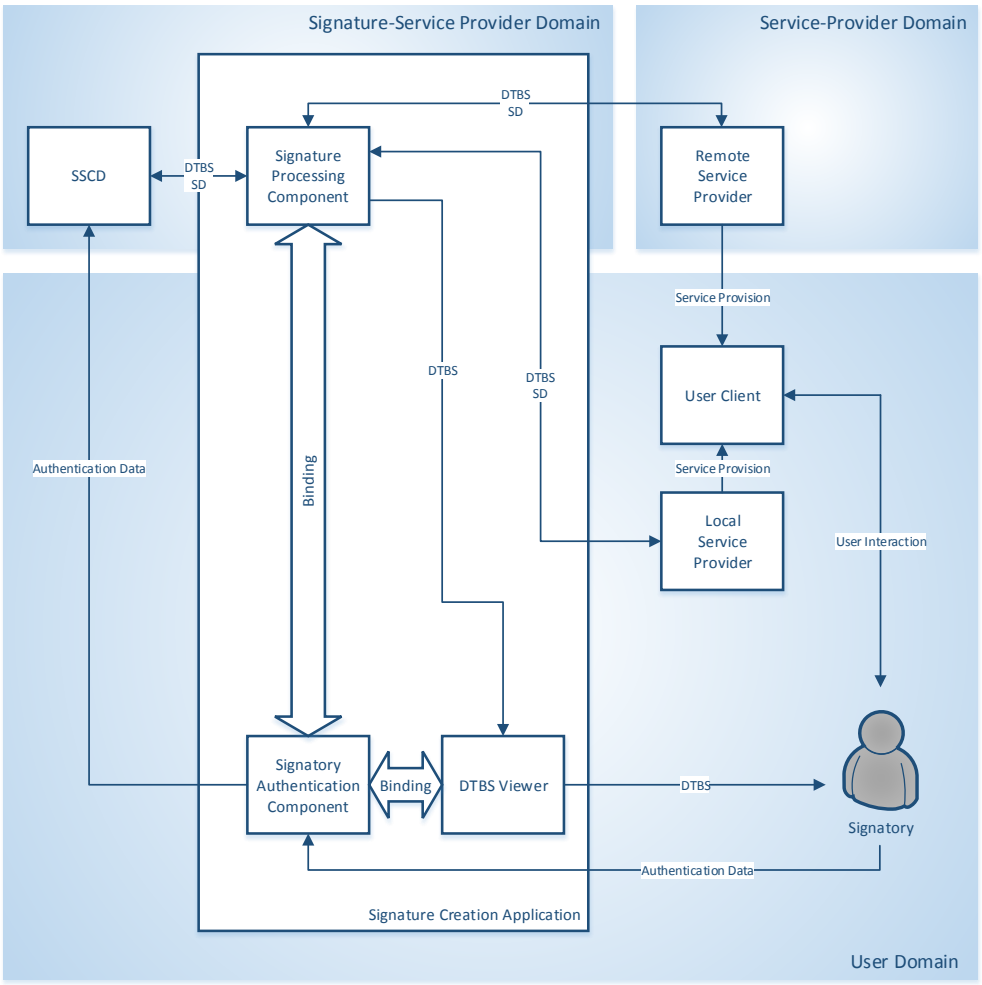

Figure 4.6: Implementation Variant D comprises solutions that implement the SCA partly remotely and the SSCD fully remotely.

The architecture of the Server-HSM Approach is illustrated in Figure 4.6. Relevant parties and components are assigned to the three domains User Domain, Service Provider Domain, and Signature-Service Provider Domain. As for all other implementation variants, the Service Provider Domain contains the Remote Service Provider only. In contrast to the Remote-SSCD Approach and the SIM Approach, the Signature-Service Provider Domain now contains both the SSCD and the Signature Processing Component. In theory, these two components can also be assigned to two different remote domains and can be provided by different entities. For instance, the provider of the Signature Processing Component could rely on a remote SSCD provided by a cloud-service provider. From the Signatory point of view, this scenario does not differ significantly from the scenario shown in Figure 4.6 in terms of feasibility, security, and usability. Hence, this scenario is not considered separately.

Like other implementation variants, also realizations that rely on the architecture shown in Figure 4.6 need to overcome several challenges. Similar to the SIM Approach, also the Server-HSM Approach

relies on an SCA, whose subcomponents are spread over different domains. This again raises the need for means to assure the required binding between the SCA's three subcomponents. Similar to the Remote-SSCD Approach, the SSCD and the Signatory Authentication Component are locally dispersed as shown in Figure 4.6. This raises the need for a secure cross-domain communication channel between these two components in order to enable a reliable transmission of authentication data. Another challenge that arises with the Server-HSM Approach is imposed by the remote nature of the used SSCD. Basically, the same considerations apply as for the Remote-SSCD Approach: as the Signatory does not physically possess the SSCD, concrete realizations need to implement alternative means, in order to assure the Signatory's sole control over remotely stored signature-creation data.

4.3 Feasibility Assessment

Possible approaches to create electronic signatures on mobile end-user devices can be classified into four general implementation variants. All of them base on the same abstract model and consist of the same set of components. These components are assigned to different domains and realized by different entities, depending on the architecture of the respective implementation variant. Similar to their common underlying abstract model, also the four implementation variants define components and interfaces on a rather abstract level. This way, each variant covers a broad spectrum of concrete realizations. For instance, Implementation Variant C, which has been denoted as the SIM Approach, covers existing mobile signature solutions relying on local SIMs. Similarly, Implementation Variant D, also denoted as Server-HSM Approach, covers server-based mobile signature solutions such as the Austrian Mobile Phone Signature.

Their abstract nature facilitates comparative assessments of the four implementation variants. As they all rely on the same set of components, they can be easily compared on conceptual level. As a first step in a series of comparative assessments, the feasibility of the four implementation variants is assessed in this section. The conducted assessment focuses on current mobile end-user devices and analyzes their capabilities to provide required functionality. In particular, opportunities to implement components of the User Domain on current mobile end-user devices are assessed. This way, the overall feasibility of the four implementation variants on current mobile end-user devices is assessed and compared.

4.3.1 Methodology

The heterogeneous ecosystem of mobile operating systems and end-user devices complicates accomplishment of a detailed feasibility assessment. Currently available mobile operating systems differ significantly in various aspects. Hence, a thorough feasibility assessment requires separate analyses for all mobile operating systems and needs to consider discrepancies between different end-user devices. To limit its complexity, the conducted feasibility assessment is restricted to the current major players on the mobile consumer-device market, i.e. Google Android[7], Apple iOS[8], and Microsoft Windows Phone 8[9]. This selection is based on current market and sales statistics [Fingas, 2014], which indicate a dominance of these platforms. By focusing on these platforms, the complexity of the conducted assessment remains manageable, while still more than 95% of all mobile end-user devices are covered.

To assess the feasibility of the four implementation variants, a thorough methodology is followed, which defines two consecutive assessment steps. In the first step, a component-specific assessment is conducted. There, the feasibility of all components identified by the abstract model, which all implementation variants are based on, is assessed separately. In the second step, results of the component-specific assessment are combined according to the architecture of the particular implementation variant.

[7] http://www.android.com/
[8] https://www.apple.com
[9] http://www.windowsphone.com

All conducted feasibility assessments have been based on publicly available information on technologies provided by different mobile platforms and on limitations of mobile operating systems. Available information has been collected from the following sources:

- **Official documentations:** Public information on mobile operating systems and mobile end-user devices represent an important basis for the conducted feasibility assessment. Information on mobile operating systems can be obtained from web resources such as official development portals provided by Google for Android [Google, 2014] or by Apple for iOS [Apple, 2014]. In addition, specifications of mobile end-user devices are usually also publicly available on vendor websites. This information has been used to identify capabilities, supported technologies, and available features of different mobile operating systems and mobile end-user devices.
- **Related scientific work:** Assessment of different mobile operating systems has been a topic of interest for the scientific community for several years. For instance, a comparison of Android and iOS has been provided by Rogers and Goadrich [2012]. A comparison of mobile platforms has also been given by Renner et al. [2011]. Results of these and related publications on capabilities and limitations of mobile platforms have also been incorporated into the conducted feasibility assessment.
- **Results of own work:** Conducted feasibility assessments have also been based on results of own scientific work. For instance, we have provided a detailed analysis of major mobile platforms and mobile operating systems in Zefferer et al. [2013b]. In this work, we have focused on relevant aspects for the development and deployment of e-government applications and services on current mobile platforms. Furthermore, also own work on analyses of encryption systems of the mobile operating systems iOS [Teufl et al., 2013b] and Android [Teufl et al., 2014a] has been considered.

Publicly available information, related work, and own research results draw a comprehensive picture of capabilities and limitations of major mobile operating systems. Based on this information, the feasibility of relevant components of mobile signature solutions is assessed in the following subsection.

4.3.2 Component-Specific Assessment

The four identified implementation variants of mobile signature solutions are all composed of the same set of components, which have been derived from the proposed abstract model for mobile signature solutions. A simplified version of this model is shown in Figure 4.7. The feasibility of all components identified by this model is assessed in the following subsections.

4.3.2.1 Service Provider

The Service Provider can be implemented either locally or remotely. As remote Service Providers are located off the local mobile device, the feasibility of this component is independent from available technologies on current smartphones and related mobile end-user devices. Hence, remotely implemented Service Providers can be assumed to be feasible.

Local Service Providers are realized on the mobile end-user device. The Signatory consumes the provided service with the help of the User Client, which also resides on the mobile end-user device. On all major mobile platforms, the use of local software is basically limited to mobile apps. In contrast to operating systems of classical end-user devices, mobile operating systems provide only limited support for inter-application communication. This renders the realization of locally implemented Service Providers, which provide User Clients a certain service through a well-defined interface, difficult. This especially applies to the mobile operating systems Apple iOS and Microsoft Windows Phone 8. On Google Android, basic support for inter-application communication is available [Chin et al., 2011], but far from

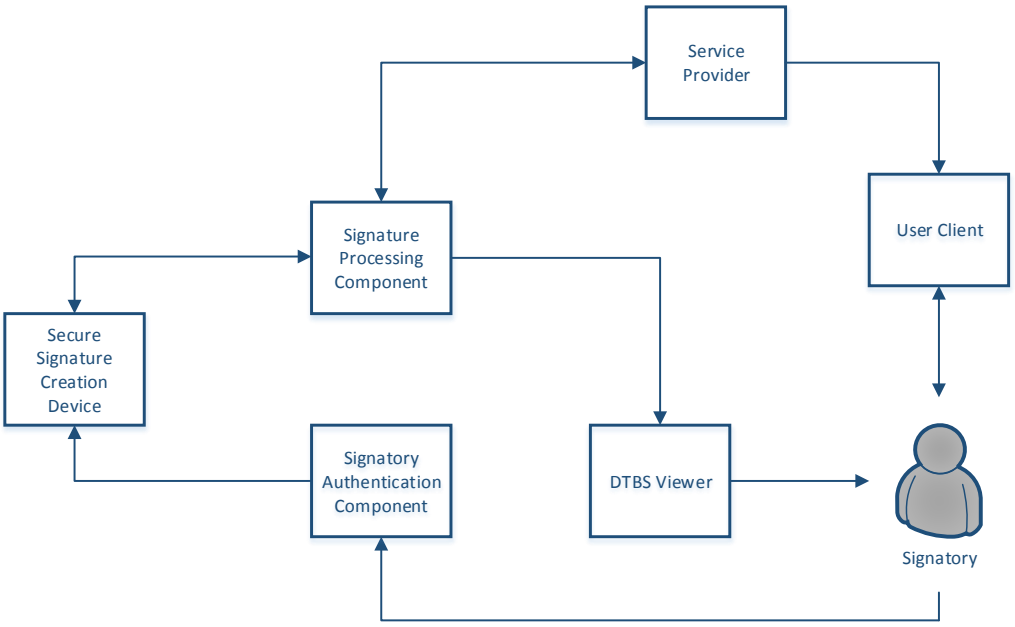

Figure 4.7: The simplified abstract model identifies basic components of signature-creation solutions.

being comparable with mechanisms available on classical end-user devices. Due to given limitations imposed by the architecture of current mobile operating systems, the realization of locally implemented Service Providers is practically limited to mobile apps. On most operating systems, these apps need to combine the roles of the Service Provider and the User Client, as inter-application communication is not widely supported. The only exception among all major mobile platforms is Google Android. As this platform and its underlying operating system support inter-application communication to a certain extent, Service Provider and User Client can also be realized by means of two separated apps.

In summary, it can be concluded that realization of a locally implemented Service Provider is feasible on all major mobile platforms as long as this component is combined with the User Client to one single app. Separated implementations of these two components are practically feasible on Google Android only. Remotely implemented Service Providers are possible on all platforms, as their feasibility is independent from capabilities of the mobile end-user device. Obtained results of the feasibility assessment of the component Service Provider are summarized in Figure 4.8.

Component	Google Android	Apple iOS	Microsoft Windows Phone 8
Local Service Provider	Feasible	Feasible if combined with User Client	Feasible if combined with User Client
Remote Service Provider	Feasible	Feasible	Feasible

Figure 4.8: The feasibility of the component Service Provider can be limited on certain platforms due to missing support for inter-application communication.

4.3.2.2 User Client

This component acts as intermediary between the Service Provider and the Signatory. In case of a locally implemented Service Provider, the User Client can be realized together with the Service Provider in one

single app.

In case of a remote Service Provider, the User Client must be implemented as a separate component residing on the mobile device. A common implementation of the User Client is the web browser. A web browser provides a user interface to the Signatory, through which services offered by a remote Service Provider, e.g. a web-application server, can be accessed. As web browsers are available on all major mobile operating systems, the general feasibility of the component User Client is evident for all these systems. Also other implementations of this component are possible. Limitations are theoretically only imposed by restrictions regarding supported communication technologies and protocols between remote Service Providers and the local User Client. Such restrictions can for instance be imposed by the mobile operating system.

In general, it can be concluded that realization of the component User Client is feasible on all major mobile platforms. Depending on the nature of the Service Provider, the User Client can either be realized as separate application or be combined with the Local Service Provider. Assessment results for the component User Client are summarized in Figure 4.9

Component	Google Android	Apple iOS	Microsoft Windows Phone 8
User Client	Feasible	Feasible	Feasible

Figure 4.9: The component User Client is feasible on all relevant platforms.

4.3.2.3 DTBS Viewer

The functionality of the DTBS Viewer is limited to displaying data to the Signatory. Displaying of data is usual functionality of mobile apps and hence feasible on all major mobile platforms. In order to obtain the data to be displayed to the Signatory, the DTBS Viewer needs to interface and communicate with the Signature Processing Component. If this component is implemented locally as a separate app, collection of data to be displayed requires inter-application communication. In this case, support of inter-application communication by the mobile operating system is crucial for the feasibility of the DTBS Viewer. Support for inter-application communication is currently mainly available on the Google Android platform. Other major mobile platforms such as Apple iOS provide only very limited means for inter-application communication [Gribsgy, 2009].

In summary, it can be concluded that realization of the DTBS Viewer is feasible on all major platforms. Some platforms might pose several restrictions regarding the implementation of the communication interface between the DTBS Viewer and the Signature Processing Component, as these platforms provide only minor support for inter-application communication. On these platforms, the Signature Processing Component and the DTBS Viewer must be implemented in a way that does not require inter-application communication for data exchange. For instance, these two components can be implemented by a single mobile app. The feasibility of the component DTBS Viewer is summarized in Figure 4.10.

Component	Google Android	Apple iOS	Microsoft Windows Phone 8
DTBS Viewer	Feasible	Feasible if Signature Processing Component is not implemented as separate app	Feasible if Signature Processing Component is not implemented as separate app

Figure 4.10: On some platforms, certain restrictions regarding the implementation of the Signature Processing Component need to be considered in order to assure the feasibility of the DTBS Viewer.

4.3.2.4 Signatory Authentication Component

The Signatory Authentication Component reads authentication data entered by the Signatory and forwards these data to the SSCD. All major mobile platforms provide user-input capabilities for this purpose. Hence, this functionality is feasible on all major mobile platforms.

Accessing the SSCD to forward entered authentication data is more challenging. This especially applies to scenarios, in which the SSCD is locally dispersed from the Signatory Authentication Component. As the Signatory Authentication Component needs to be local in any case, such scenarios comprise implementation variants relying on a remote SSCD. To enable communication between a local Signatory Authentication Component and a remote SSCD, appropriate communication protocols need to be applied. Both the Signatory Authentication Component and the SSCD must support these protocols. As the Signatory Authentication Component is typically realized in software, support for required communication protocols to access remote SSCDs is a matter of implemented functionality and hence in principle feasible. In case of a local SSCD, special communication protocols to access an SSCD residing in a remote domain are not required. In this case, communication capabilities between the Signatory Authentication Component and the SSCD heavily depend on the mobile operating system and on the implementation of the SSCD. Details on accessing local SSCDs on different mobile platforms are discussed in Section 4.3.2.6 in more detail.

In summary, it can be concluded that realizing the functionality of the Signatory Authentication Component is feasible on all major platforms in case the used SSCD provides interfaces that allow the Signatory Authentication Component to forward entered authentication data. This is summarized in Figure 4.11.

Component	Google Android	Apple iOS	Microsoft Windows Phone 8
Signatory Autentication Component	Feasible if SSCD provides appropriate interface	Feasible if SSCD provides appropriate interface	Feasible if SSCD provides appropriate interface

Figure 4.11: The Signatory Authentication Component is feasible on all platforms, if the SSCD provides communication interfaces to forward entered authentication data.

4.3.2.5 Signature Processing Component

The Signature Processing Component interfaces the Service Provider, the DTBS Viewer, and the SSCD. Provision of suitable interfaces to these three components is the most challenging aspect regarding the feasibility of the Signature Processing Component. In general, the Signature Processing Component can be implemented locally or remotely. Depending on its location, different aspects that influence the feasibility of the Signature Processing Component need to be taken into account.

In case of a remote Signature Processing Component, communication interfaces to other components are feasible on all platforms, if these components provide the required interfaces. This is typically not an issue for software-based components such as the DTBS Viewer, as their functionality can be easily extended. The situation is however more complicated for the SSCD, which is typically realized in hardware. Still, the SSCD needs to provide an interface, through which the remote Signature Processing Component can access the SSCD's functionality. Provision of such an interface can be especially challenging in case of a local SSCD. In this case, the local SSCD needs to provide an interface that can be accessed by the Signature Processing Component from a remote domain. Hence, both the local SSCD and the remote Signature Processing Component need to support communication protocols that enable cross-domain communication. This is typically not an issue for the software-based Signature Processing Component, but might be challenging for the SSCD. This will be discussed in more detail in Section 4.3.2.6.

In case of a local Signature Processing Component, feasibility is even more difficult to achieve. In this case, capabilities of the Signature Processing Component to communicate with other local components additionally depend on the platform's support for inter-application communication. Support for inter-application communication differs between mobile platforms and operating systems. Google Android currently provides the broadest support for inter-application communication and hence the highest degree of flexibility regarding realizations of the Signature Processing Component. Considering the required access to the SSCD, two scenarios need to be distinguished for a locally implemented Signature Processing Component. In the first scenario, the SSCD is also realized locally. In this case, communication capabilities between the Signature Processing Component and the SSCD heavily depend on the mobile operating system and on the concrete realization of the SSCD. The feasibility of different SSCD realizations is discussed in Section 4.3.2.6 in more detail. In the second scenario, the SSCD is realized remotely. In this case, both the SSCD and the Signature Processing Component need to support communication protocols that enable cross-domain communication. Being typically implemented in software, support of such protocols is technically feasible for a local Signature Processing Component. Capabilities of remote SSCDs to provide support for cross-domain communication protocols are discussed in Section 4.3.2.6.

In summary, support for inter-application communication and capabilities to access the SSCD can be identified as key aspects for the feasibility of the Signature Processing Component. This especially applies to local Signature Processing Components. Inter-application communication is currently mainly available for the Google Android platform. Other major mobile platforms provide only limited support for this feature. Access to the SSCD is basically feasible on all major platforms in case the SSCD provides interfaces for that purpose. The feasibility of the Signature Processing Component on different platforms is summarized in Figure 4.12.

Component	Google Android	Apple iOS	Microsoft Windows Phone 8
Local Signature Processing Component	Feasible if SSCD provides appropriate interface	Feasible if SSCD provides appropriate interface and no inter-application communication is required	Feasible if SSCD provides appropriate interface and no inter-application communication is required
Remote Signature Processing Component	Feasible if SSCD provides appropriate interface	Feasible if SSCD provides appropriate interface	Feasible if SSCD provides appropriate interface

Figure 4.12: Depending on the implementation of the Signature Processing Component, its feasibility depends on interfaces provided by the SSCD and on inter-application communication capabilities of the mobile platform.

4.3.2.6 SSCD

The SSCD is probably the most critical component in terms of feasibility. In order to be able to produce qualified electronic signatures, SSCDs need to satisfy several requirements. This limits possible realizations of SSCDs. In general, two aspects need to be considered when assessing the feasibility of SSCDs. The first aspect concerns the availability of components that are able to meet the strict requirements for SSCDs. Availability of such a component is crucial for the feasibility of the SSCD. The second aspect concerns the capability to access an available SSCD. Only if the Signature Processing Component and the Signatory Authentication Component can communicate with the SSCD, the SSCD can be regarded as feasible. In the following, these two aspects are analyzed in more detail.

Regarding the first aspect, i.e. the availability of suitable components to implement an SSCD, two scenarios need to be distinguished. In the first scenario, the SSCD is implemented remotely. Remote SSCDs do not depend on capabilities of the mobile end-user device. Furthermore, the Austrian Mobile Phone Signature proves that remote SSCDs are in principle feasible. Therefore, remote SSCDs can be

assumed to be available and feasible. Hence, this scenario does not need to be analyzed in more detail. In the second scenario, the SSCD is implemented locally on the mobile end-user device. This is the more challenging scenario, as the feasibility of the SSCD heavily depends on the capabilities of the mobile end-user device and the used operating system. Local SSCDs can be classified into internal and removable implementations. The first category comprises SSCDs that rely on secure hardware elements being an integral part of the mobile device. Such hardware elements are for instance used in several devices to implement secure key chains that can be used by mobile applications to securely store cryptographic keys. Some platforms also make use of such hardware components to implement secure encryption mechanisms [Teufl et al., 2013b]. A secure hardware element has for instance also been used by the mobile payment system Google Wallet[10] to store security-critical data. The second category covers SSCDs that can be connected to mobile end-user devices using available interfaces. The probably most frequently used implementation of this category is the SIM. Primarily intended to authenticate the user at the mobile network, special SIMs can also be employed as SSCDs. This has been shown in practice by several SIM-based signature solutions that are currently in productive operation. As an alternative to SIMs, also special microSD memory cards have been introduced during the past years that include a secure hardware element, which is theoretically capable of creating qualified electronic signatures. These memory cards can be used on mobile devices featuring an interface for this cards. Security-enhanced microSD cards are for instance offered by the companies DeviceFidelity[11] or GO-Trust[12]. Recently, also NFC-based security tokens[13] have been introduced as an alternative. These tokens integrate a secure element capable of providing cryptographic functionality. Access to the functionality of these tokens is provided via the contactless NFC interface, which is supported by an increasing number of mobile end-user devices. For the sake of completeness, also solutions relying on the use of special adapters to physically connect smart cards to mobile end-user devices have to be mentioned as possible implementations of this category. Examples are the iAuthenticate™ smart card reader for the Apple iPhone[14] or the BlackBerry smart card reader[15]. Regarding the availability of suitable components to implement local SSCDs, it can be concluded that such components exist for most mobile end-user devices.

Besides availability of suitable SSCDs, the capability to communicate with these SSCDs has also been identified as a crucial feasibility aspect. To systematically analyze this aspect for different SSCD implementations, two scenarios need to be considered. In the first scenario, the SSCD and the component that needs to access the SSCD are located in the same domain. This scenario applies for instance to solutions that implement all components locally on the mobile end-user device. In the second scenario, the SSCD and the component that needs to access the SSCD are located in different domains. This applies for instance to solutions that implement the SSCD remotely and all other components locally on the mobile end-user device. Depending on the scenario, different aspects regarding the communication between the SSCD and accessing components need to be considered.

In the first scenario, the SSCD and the component that requires access to the SSCD, i.e. the Signature Processing Component or the Signatory Authentication Component, are implemented in the same domain. If both the SSCD and the accessing component are implemented remotely, the feasibility of SSCD access is independent from technologies available on the mobile end-user device. Hence, this case does not need to be considered in more detail. If both components are implemented locally, feasibility to access the SSCD depends on the SSCD's implementation and on the mobile operating system. Access to secure hardware elements being an integral part of mobile end-user devices is not feasible for third-party applications on all major platforms. Removable SSCD implementations are in general easier to access from software components. Still, several restrictions apply depending on the SSCD implementation and

[10] https://wallet.google.com

[11] http://www.devifi.com/in2pay_microsd.html

[12] http://www.go-trust.com/products/swp-secure-microsd/

[13] http://www.yubico.com/

[14] http://www.identive-group.com/en/products-and-solutions/identification-products/mobility-solutions/mobile-readers/iauthenticate-smart-card-reader

[15] http://de.blackberry.com/business/topics/security/government.html

on the underlying mobile operating system. For instance, SIMs cannot be accessed directly from mobile applications on all major platforms. In contrast, memory cards featuring cryptographic functionality can also be accessed by locally installed mobile apps, as long as this functionality is supported by the underlying mobile operating system. This is currently the case for the Google Android platform, which basically supports access to secure elements via the Personal Computer/Smart Card (PC/SC) protocol[16]. The feasibility to access external SSCDs such as cryptography-enabled NFC tokens mainly depends on the support for the employed interface technology that is used to connect the secure element to the mobile end-user device. In the special case of NFC, an increasing number of mobile end-user devices are supporting this technology. Still, the feasibility of NFC-based SSCDs is limited, mainly due to the fact that Apple as a major player does not allow third-party app to use integrated NFC features.

In the second scenario, the SSCD and the component that needs to access the SSCD are implemented in different domains. Hence, access to the SSCD requires cross-domain communication. This potentially requires support for additional communication protocols, in order to bridge the gap between different domains using existing communication networks. Support for additional protocols is in general feasible for software-based components, as these components can be easily extended. In contrast, support of additional communication protocols can be problematic for typically hardware-based SSCDs, as hardware cannot be easily extended with additional functionality. Hence, if the SSCD does not support the required communication protocol itself, an additional software module needs to be attached to the SSCD. This module acts as an adapter and translates the required cross-domain communication protocol to a protocol supported by the SSCD. The feasibility of this additional software module hence defines a crucial requirement for implementations relying on locally dispersed software components and SSCDs. For remote SSCDs, realization of this additional module acting as proxy in front of the SSCD is considered feasible, as remote components are not limited by technology available on the mobile end-user device. For local SSCDs, the situation is more complex. Implementing the additional module as local third-party application requires again the mobile operating system to support access to the SSCD. As mentioned above, this is feasible on Google Android for a limited set of local SSCD realizations only. As an alternative, the required additional module can also be implemented by the mobile operating system itself. This approach is for instance followed by SIM-based signature solutions. Data received from a remote Signature Processing Component through the mobile network is directly forwarded to the local SIM by the mobile operating system. From a feasibility point of view, this approach is advantageous compared to reliance on third-party applications implementing the required additional module. However, due to reliance on features provided by the mobile operating system, implementation alternatives are limited.

Component	Google Android	Apple iOS	Microsoft Windows Phone 8
Local SSCD	Feasible	Restricted to SIM-based implementations	Restricted to SIM-based implementations
Remote SSCD	Feasible	Feasible	Feasible

Figure 4.13: While remotely implemented SSCDs are generally feasible, local SSCDs are restricted to SIM-based solutions on most platforms.

In general, it can be concluded that access to and communication with SSCDs can be problematic especially in the case of locally implemented SSCDs. Only Android provides support for accessing different types of local SSCDs. On all other platforms, the use of local SSCDs is basically restricted to special SIMs. As SIMs cannot be accessed directly from local third-party applications, SIM-based solutions are rather limited in terms of possible implementation variants. Most problems regarding the feasibility of SSCDs can be prevented by relying on remotely implemented SSCDs. The feasibility of remotely implemented SSCDs has been shown by the Austrian Mobile Phone Signature. By relying on

[16] http://code.google.com/p/seek-for-android/

a remote SSCD, most feasibility-related limitations that are imposed by restrictions of mobile end-user device and operating systems can be overcome. The feasibility of the component SSCD is summarized in Figure 4.13.

4.3.3 Assessment of Implementation Variants

The conducted component-specific feasibility assessment has yielded two basic findings. First, access to and communication with the SSCD is a crucial factor that affects feasibility. Depending on the realization of the SSCD and of components that need to access the SSCD, communication between the SSCD and these components can be challenging. This is mainly caused by limitations of mobile operating systems that often do not allow software components to access locally implemented SSCDs. Second, the conducted assessment has yielded the need for inter-application communication as another feasibility-reducing factor. This is mainly due to the fact that some mobile operating systems apply strict means to isolate applications running on mobile devices from each other.

To complete the feasibility assessment, the results of the component-specific assessment are combined according to the architectures of the four identified implementation variants for mobile signature solutions. This way, the feasibility of each implementation variant is assessed and the most feasible approach is determined.

4.3.3.1 Feasibility of Implementation Variant A

Implementation Variant A has been named the Classical Approach, as all components but the Remote Service Provider are implemented locally on the Signatory's mobile end-user device. This way, this approach basically reflects the architecture of classical smart card based solutions. Considering results of the component-specific feasibility assessment, two major drawbacks of the Classical Approach, i.e. Implementation Variant A, can be identified.

First, the Classical Approach defines a local implementation of the SSCD. This limits the feasibility of this implementation variant to mobile end-user devices and operating systems that provide support for implementing and accessing a local SSCD. An advantage of solutions following the Classical Approach is the fact that the Signatory Authentication Component, the Signature Processing Component, and the SSCD are all implemented in one and the same domain. Thus, no cross-domain communication between the SSCD and other components is required. Hence, feasibility of an additional module that implements required cross-domain protocols is not necessary.

Second, the Signature Processing Component is implemented locally according to the Classical Approach. This causes two potential problems. First, a local Signature Processing Component rules out techniques to access the local SSCD using a remote component, such as the mobile network operator in case of SIM-based SSCD implementations. Second, a local realization of the Signature Processing Component requires inter-application communication, as the Local Service Provider needs to exchange data with the Signature Processing Component. This can be problematic depending on the underlying mobile platform and operating system.

Implementation Variant	Google Android	Apple iOS	Microsoft Windows Phone 8
Implementation Variant A: The Classical Approach	Feasible	Infeasible	Infeasible

Figure 4.14: Implementation Variant A is feasible on Google Android only.

In summary, it can be concluded that Implementation Variant A, i.e. the Classical Approach, is applicable on a subset of current mobile end-user device only. Concretely, this approach can only be

applied on mobile end-user devices that rely on the mobile operating system Google Android. This is illustrated in Figure 4.14.

4.3.3.2 Feasibility of Implementation Variant B

Implementation Variant B relies on a remote SSCD and has hence been denoted as the Remote-SSCD Approach. Apart from the SSCD, all other components are implemented locally. As this approach relies on a remote SSCD, the underlying platform's support for local SSCDs does not influence the overall feasibility. In this aspect, this approach is advantageous compared to the Classical Approach represented by Implementation Variant A. Reliance on a remote SSCD however requires additional modules to enable cross-domain communication between the SSCD and locally implemented components. Concretely, this applies to the locally implemented Signature Processing Component and Signatory Authentication Component. According to the conducted component-specific feasibility assessment, such modules are feasible for these components. Being implemented remotely, also the SSCD can be easily equipped with an additional module that adds support for cross-domain communication. In summary, the remote implementation of the SSCD does hence not impose unsolvable problems in terms of feasibility.

However, the Remote-SSCD Approach shares another conceptual drawback with the Classical Approach represented by Implementation Variant A. As these two approaches differ only in the location of the SSCD, both the Remote-SSCD Approach and the Classical Approach rely on a local Signature Processing Component. Thus, in both cases the underlying mobile operating system needs to provide means for inter-application communication. These means are required to enable communications between a locally implemented Service Provider and the local Signature Processing Component. Hence, also the Remote-SSCD Approach is only feasible in scenarios that do not require inter-application communication.

In summary, the Remote-SSCD Approach is advantageous compared to the Classical Approach, as it releases the mobile end-user device from implementing the SSCD. However, due to the need for inter-application communication, the feasibility of this approach can still be limited on several platforms. Concretely, the Remote-SSCD Approach is restricted to platforms that provide sufficient support for inter-application communication. This is currently the case for Google Android only. The feasibility of Implementation Variant B, i.e. the Remote-SSCD Approach, is summarized in Figure 4.15.

Implementation Variant	Google Android	Apple iOS	Microsoft Windows Phone 8
Implementation Variant B: The Remote-SSCD Approach	Feasible	Feasible if no inter-application communication is required	Feasible if no inter-application communication is required

Figure 4.15: Implementation Variant B is only conditionally feasible on Apple iOS and Microsoft Windows Phone 8.

4.3.3.3 Feasibility of Implementation Variant C

The need for inter-application communication has been identified as feasibility-reducing aspect for approaches that rely on a local Signature Processing Component. Implementation Variant C, i.e. the SIM Approach, avoids the need for inter-application communication by implementing the Signature Processing Component remotely. Thus, solutions based on the SIM Approach are feasible irrespective of the underlying operating system's support for inter-application communication.

However, reliance on a local SSCD again limits the feasibility of the SIM Approach. Solutions following this approach are applicable on a subset of mobile end-user devices only. Concretely, applicability is limited to devices and platforms that are able to provide a local SSCD. Additional limitations are also imposed by the need to access the local SSCD from the remotely implemented Signature Processing

Component. The local SSCD must support communication protocols that allow remote components to access its functionality. The component-specific feasibility analysis has shown that SIM-based solutions currently represent the only implementation alternative for the SIM Approach that is feasible on all major platforms. Only on Google Android, also third-party apps can be used to act as intermediary between a remote Signature Processing Component and a local SSCD. On this platform, concrete realizations of the SIM Approach are hence not restricted to SIM-based solutions.

Access to the local SSCD is not only required for the remote Signature Processing Component. The local SSCD must also be accessed by the locally implemented Signatory Authentication Component. As most mobile operating systems do not provide third-party apps access to local SSCDs, solutions following the SIM Approach are again limited to the use of SIMs as SSCDs. SIM-based solutions typically rely on the SIM Application Toolkit to obtain authentication data from the Signatory. Support for the SIM Application Toolkit is currently provided by all major mobile platforms. Hence SIM-based solutions are currently feasible on all major platforms. However, considering the fact that the SIM Application Toolkit represents a rather old and hardly used technology, the future feasibility of SIM-based solutions is arguable.

Despite identified limitations regarding possible implementation alternatives, the SIM Approach is basically feasible on mobile end-user devices at present. Various existing signature solutions following this approach show its applicability in practice. Still, implementation alternatives are limited due to restricted technical opportunities to access the local SSCD from remote and local components. The feasibility of Implementation Variant C, i.e. the SIM Approach, is summarized in Figure 4.16.

Implementation Variant	Google Android	Apple iOS	Microsoft Windows Phone 8
Implementation Variant C: The SIM Approach	Feasible	Feasible but restricted to SIM-based solutions	Feasible but restricted to SIM-based solutions

Figure 4.16: Implementation Variant C is only conditionally feasible on Apple iOS and Microsoft Windows Phone 8.

4.3.3.4 Feasibility of Implementation Variant D

Local implementations of the SSCD or the Signature Processing Component have turned out to reduce feasibility. If these two components are implemented locally, the feasibility of concrete solutions is limited to certain mobile end-user devices and operating systems. In particular, operating systems need to provide means to implement and access SSCDs locally and to support inter-application communication. On mobile operating systems that are unable to provide these features, mobile signature solutions relying on local SSCDs or local Signature Processing Components are infeasible.

Implementation Variant D prevents these limitations by realizing both components in a remote domain. Due to its remote and server-based approach, Implementation Variant D has also been denoted as Server-HSM Approach. As both the SSCD and the Signature Processing Component are implemented remotely, only few technical requirements need to be met by the mobile end-user device. This assures a high degree of feasibility.

As the local Authentication Data Provider and the remote SSCD are locally dispersed, the SSCD needs to provide cross-domain access to its functionality. Hence, both the local Signatory Authentication Component and the remote SSCD need to implement an additional module that enables cross-domain communication between these two components. The conducted component-specific assessment has shown that this is feasible for both components.

In summary, the Server-HSM Approach is the most feasible approach among all four implementation variants. Compared to the SIM Approach, which is also feasible in practice, the Server-HSM Approach allows for more implementation alternatives, as critical components are implemented remotely. This

way, fewer requirements are defined for the local end-user device. The feasibility of Implementation Variant D, i.e. the Server-HSM Approach, is summarized in Figure 4.17.

Implementation Variant	Google Android	Apple iOS	Microsoft Windows Phone 8
Implementation Variant D: The Server-HSM Approach	Feasible	Feasible	Feasible

Figure 4.17: Implementation Variant D is feasible on all platforms.

4.3.3.5 Comparison of Implementation Variants

The feasibility of the four implementation variants is mainly determined by capabilities of the underlying mobile platform and operating system. The conducted assessment has focused on the three major platforms Google Android, Apple iOS, and Microsoft Windows Phone 8. The feasibility of each implementation variant has been assessed for each of these platforms. Figure 4.18 combines all obtained assessment results and shows, which implementation variant is feasible on which platform.

Implementation Variant	Google Android	Apple iOS	Microsoft Windows Phone 8
Implementation Variant A: The Classical Approach	Feasible	Infeasible	Infeasible
Implementation Variant B: The Remote-SSCD Approach	Feasible	Feasible if no inter-application communication is required	Feasible if no inter-application communication is required
Implementation Variant C: The SIM Approach	Feasible	Feasible but restricted to SIM-based solutions	Feasible but restricted to SIM-based solutions
Implementation Variant D: The Server-HSM Approach	Feasible	Feasible	Feasible

Figure 4.18: Implementation Variant D represents the most feasible approach.

A direct comparison of the three major platforms shows that only Google Android supports all four implementation variants. This is mainly due to the fact that this platform provides support for inter-application communication and enables realization of locally implemented SSCDs. The mobile operating systems Apple iOS and Microsoft Windows Phone 8 are more problematic in terms of feasibility, as these platforms provide only limited support for inter-application communication and for local SSCDs. On these platforms, Remote-SSCD Approaches and SIM Approaches are only partly feasible and limited to certain implementation alternatives. Classical Approaches must even be considered completely infeasible on these two platforms.

The direct comparison of different implementation variants shown in Figure 4.18 also reveals that the Server-HSM Approach provides the highest degree of feasibility on all major platforms. In fact, it is the only approach that is feasible without limitations on all investigated platforms. Hence, the Server-HSM Approach represented by Implementation Variant D is obviously the best solution from a feasibility point of view.

4.4 Security Assessment

Security is one of the key requirements and success factors of mobile government. Apparently, security is also a crucial aspect for mobile signature solutions, as these solutions are typically used in security-sensitive fields of application. The security of mobile signature solutions already needs to be considered

during their design and development. As a preparatory activity to the development of a concrete signature solution for mobile end-user devices, four possible implementation variants have been identified. In this section, the security of these implementation variants is assessed in detail. This way, the most secure approach is determined.

4.4.1 Methodology

In order to rely on an approved method and to follow a systematic approach, the conducted security assessment is loosely based on the concepts of Common Criteria [Common Criteria, 2013]. Development of security recommendations along the concepts of Common Criteria and protection profiles such as the protection profile for secure signature creation devices [CEN/ISSS, 2001] is common practice. For instance, the Council of Europe has applied this methodology in the context of a risk analysis for e-voting solutions [European Council, 2004]. However, the concepts of Common Criteria are usually used to assess and evaluate the security of one particular solution. These concepts are less suitable to compare the security of different solutions on conceptual level. Therefore, the concepts of Common Criteria are adapted where necessary in the course of this assessment. This leads to a methodology comprising two consecutive steps.

In the first step, a set of assumptions is defined in order to define and limit the scope of the conducted security analysis. Based on these assumptions, relevant assets are identified that need to be protected by mobile signature solutions. Based on the architectures of the four implementation variants, components and communication paths are identified, at which these assets are potentially exposed to threats. This way, the security of the four implementation variants is assessed and compared on conceptual level.

In a subsequent assessment step, the security of each identified component and communication path when being implemented using current mobile technologies is analyzed in detail. For this purpose, technologies that are currently available on mobile end-user devices are taken into account. The conducted analyses are based on results from own research on smartphone security presented e.g. in Zefferer et al. [2013b], Zefferer and Teufl [2013], and Teufl et al. [2013b]. In addition, related scientific work presented by Felt et al. [2012], Enck and Octeau [2011], or Barrera et al. [2010] is considered as well. Analysis results are finally assembled to systematically assess the security of the four implementation variants for mobile signature solutions. This way, their capabilities to protect assets by means of currently available technologies are assessed.

4.4.2 Assumptions

The conducted security assessment is based on a set of assumptions, which define and limit the scope of the security assessment to relevant aspects. Concretely, the following two general assumptions are made:

- **Security of server-based components:** Server-based components of mobile signature solutions are assumed to be secure and hence out of scope for this analysis. This assumption is valid for several reasons. First, server-based components can be operated in a secure environment. This environment can be protected by both technical and organizational means. Second, server-based components can be accessed through defined interfaces only. This is in contrast to locally implemented components, which are potentially fully exposed to attackers in case of compromised or stolen end-user devices. In general, it can be stated that even though possible attacks on server-based components must not be ignored, the security of these components is easier to assure. It is hence reasonable to limit the conducted security assessment to local components.

- **Security of SSCDs:** The SSCD is assumed to be secure and to be able to securely store and process data. These features are typically evaluated by means of standardized certifications for SSCDs. The conducted security analysis is based on the assumption that only certified SSCDs are

used and that these devices hence provide the required level of security. In this case, the made assumption regarding secure SSCDs is valid.

4.4.3 Assets

Assets are a key concept of most security assessments and basically define values that need to be protected by the assessed system. In this section, relevant assets are identified by means of the defined model for mobile signature solutions. These assets are then mapped to the four implementation variants that have been derived from the defined abstract model.

4.4.3.1 Identification of Assets

Figure 4.19 recalls the model that has been used to derive the four implementation variants for mobile signature solutions. In this section, the same model is used to derive a set of relevant assets. This is accomplished by identifying security-critical data that is processed by components of the model or transferred between components. This finally yields the three assets Authentication Data (AD), DTBS, and SD.

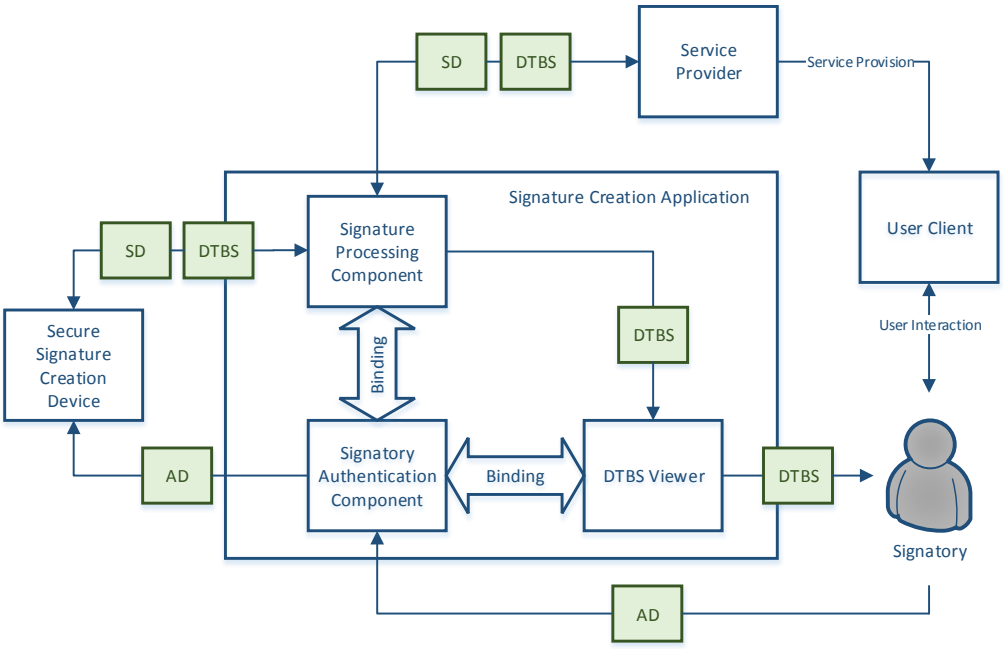

Figure 4.19: Three assets can be derived from the developed abstract model.

The three identified assets have also been added to the model shown in Figure 4.19. This way, it becomes apparent, which asset is transferred between which components. The three assets are described in the following in more detail.

- **AD:** This is data provided by the Signatory to authorize access to the Signatory's signature-creation data, i.e. the private cryptographic key. Thus, AD needs to be provided to authorize a signature-creation process in the SSCD. As control over private cryptographic keys must remain solely at the Signatory, authentication data may be known to the Signatory only and the confidentiality of these data must be maintained.

- **DTBS:** This is data created by the Service Provider and transmitted to the SSCD for the purpose of signing. The DTBS is displayed to the Signatory prior to completion of the signature-creation process. The integrity of the DTBS must be maintained during the entire signature-creation process. This applies to transmission of the DTBS from the Service Provider to the SSCD via the Signature Processing Component. Furthermore, this also applies to the DTBS while it is displayed to the Signatory by means of the DTBS Viewer. Even though modification of displayed DTBS does not necessarily affect the integrity of DTBS transmitted to the SSCD for signing, analogy between the two instances of these data must be guaranteed.
- **SD:** The SD is the result of the signature-creation process. The SD is created by the SSCD and returned to the Service Provider by means of the Signature Processing Component. To assure positive verifiability of the SD, its integrity needs to be maintained. Assuming the use of proper cryptographic methods, it is infeasible for an attacker to compute valid SD on behalf of the Signatory outside the SSCD. Still, SD represents a relevant asset, as unauthorized modifications of SD can negatively influence the verifiability of created signatures. By intentionally modifying intercepted SD, an attacker could sabotage the functionality of electronic signature based services. It is hence reasonable to consider SD, even though AD and DTBS are typically regarded as stronger assets.

In addition to these three assets, additional assets could be listed here. For instance, the protection profile for SSCDs [CEN/ISSS, 2001] defines the additional assets Signature Creation Data, Reference Authentication Data, or Signature-Creation Function. Most of these assets are also present in the model shown in Figure 4.19. However, all these assets are protected by the SSCD. As the SSCD is assumed to be secure, assets protected by the SSCD are not considered separately.

4.4.3.2 Mapping of Assets to Implementation Variants

The three identified assets must be protected by a signature solution during the entire signature-creation process. During this process, assets can be prone to attacks when being processed by components of the signature solution or when being transmitted over communication paths between these components. The set of components and communication paths, at which assets are prone to attacks, depends on the respective signature solution and its underlying architecture. In total, four different implementation variants for mobile signature solutions have been identified. In this section, the three relevant assets are mapped to these implementation variants. This way, for each implementation variant, components and communication paths are identified, at which assets are prone to attacks. This enables a comparison of the security of different approaches to implement mobile signature solutions on conceptual level.

Figure 4.20 shows an extended version of the architecture of Implementation Variant A representing the Classical Approach. In addition to the original architecture, communication channels have been amended with assets that are exchanged and transmitted over these channels. Additionally, all components that process or store identified assets have been marked accordingly.

The extended architecture shown in Figure 4.20 shows that the asset AD is present in the User Domain only. This asset is read from the Signatory by the Signatory Authentication Component and forwarded to the locally implemented SSCD. Hence, the Signatory Authentication Component is the only component that processes the asset AD.

The asset DTBS is created by the locally or remotely implemented Service Provider and transmitted to the local Signature Processing Component. This component finally forwards the DTBS to the SSCD. Additionally, the Signature Processing Component forwards the DTBS to the DTBS Viewer. Hence, the asset DTBS is processed by the Service Provider, the Signature Processing Component, and the DTBS Viewer. In case of a Local Service Provider, the asset DTBS is present in the User Domain only. In case of a Remote Service Provider, the asset DTBS is transferred from the remote Service Provider Domain to the local Signature Processing Component.

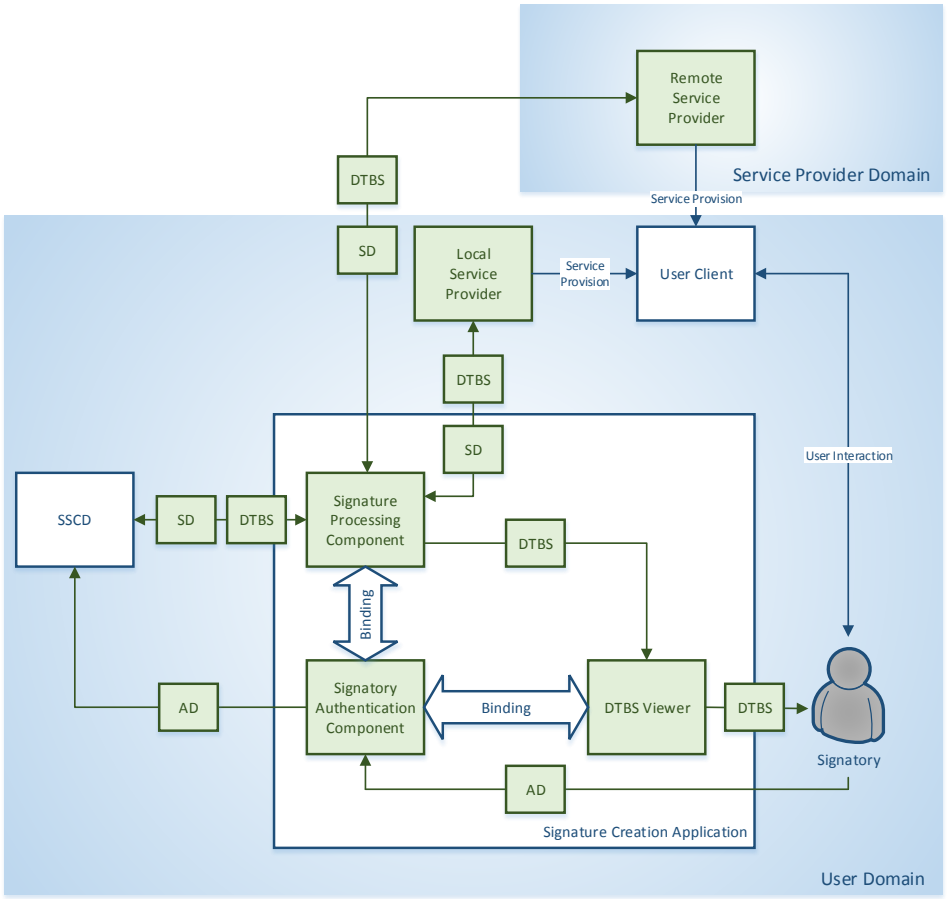

Figure 4.20: Mapping of assets to Implementation Variant A.

Similar considerations apply to the asset SD. The asset SD is created by the SSCD and transferred to the Signature Processing Component. The Signature Processing Component returns the SD to the Service Provider. Hence, the asset SD is present at the same components as the asset DTBS with the exception of the DTBS Viewer.

Identified assets have also been mapped to the general architecture of Implementation Variant B, which represents the Remote-SSCD Approach. The resulting extended architecture is shown in Figure 4.21. Components and communication paths, at which identified assets are present during a signature-creation process, have again been marked accordingly.

The key characteristic of Implementation Variant B is the remote implementation of the SSCD. This has an impact on all processed assets. Figure 4.21 shows that all assets need to be transferred between the User Domain and the Signature-Service Provider Domain. Hence, Implementation Variant B requires cross-domain transmission of all assets even in case of a locally implemented Service Provider.

Cross-domain transmission of assets is also required by solutions relying on Implementation Variant C and hence following the SIM Approach. This becomes apparent from Figure 4.22, which shows the extended architecture of this approach. Components and interfaces, at which relevant assets are present during signature-creation processes have again been marked accordingly.

Following the SIM Approach, the SSCD is implemented locally, while the Signature Processing Component resides in the remote Signature-Service Provider Domain. This way, the assets DTBS and SD need to be transmitted across two different domains. In case of a Remote Service Provider residing in a separate Service Provider Domain, the assets DTBS and SD even have to be shared between three

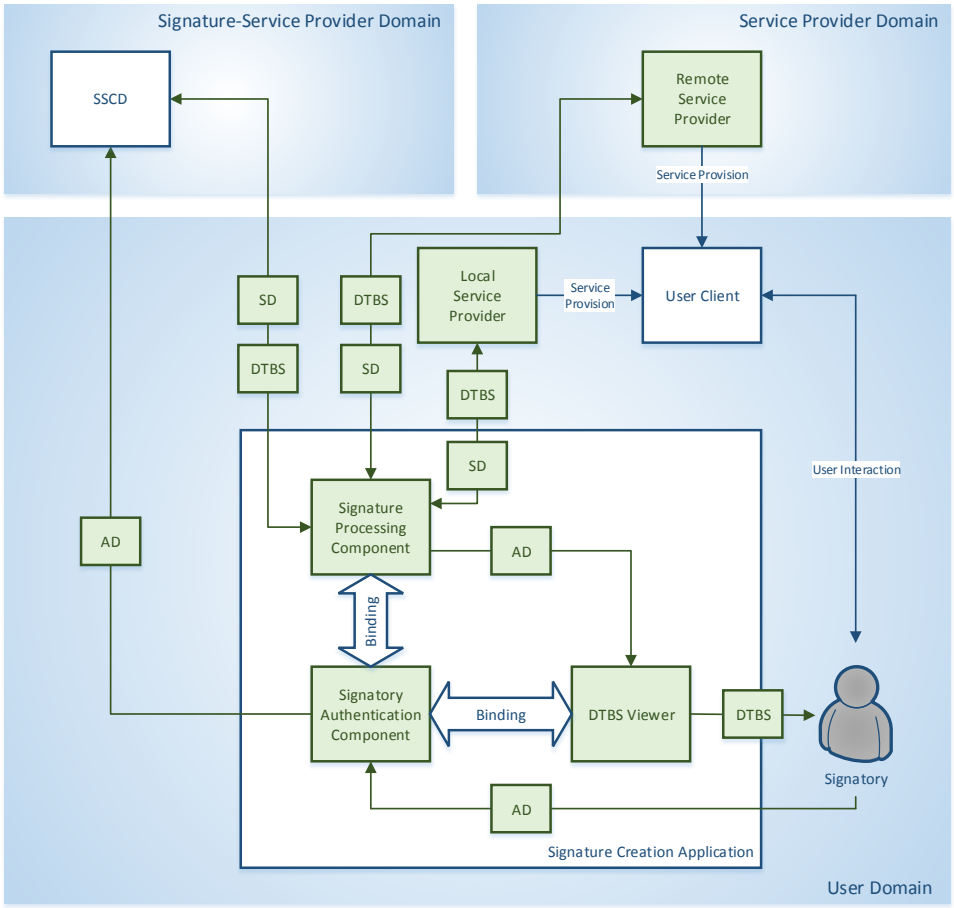

Figure 4.21: Mapping of identified assets to Implementation Variant B.

different domains.

As the SIM Approach relies on a local SSCD, the asset AD always remains in the User Domain during a signature-creation process. Similar to the Classical Approach, the asset AD is processed by the local Signatory Authentication Component only.

Finally, the three assets have also been mapped to the general architecture of Implementation Variant D, which represents the Server-HSM Approach. This yields the extended architecture shown in Figure 4.23. Following the Server-HSM Approach, both the SSCD and the Signature Processing Component are implemented remotely. Accordingly, both components are assigned to the remote Signature-Service Provider Domain. This has some interesting implications.

First, the asset AD has to be transmitted from the User Domain to the Signature-Service Provider Domain. In this regard, the Server-HSM Approach resembles the Remote-SSCD Approach represented by Implementation Variant B. Second, in case of a Remote Service Provider, the asset SD is never present in the User Domain. This can be beneficial in case of a compromised local end-user device. Similar considerations apply to the asset DTBS. In case of a Remote Service Provider, only a copy of the asset DTBS is present in the User Domain. This means that an attacker, who has compromised components of the User Domain, is able to intercept and reveal the asset DTBS. However, as the asset DTBS is directly sent from the remote Signature Processing Component to the remote SSCD, the DTBS cannot be modified by an attacker in the User Domain.

Mapping the three identified assets to the four implementation variants of mobile signature solutions already yields several interesting insights. In general, all identified assets are processed by the same

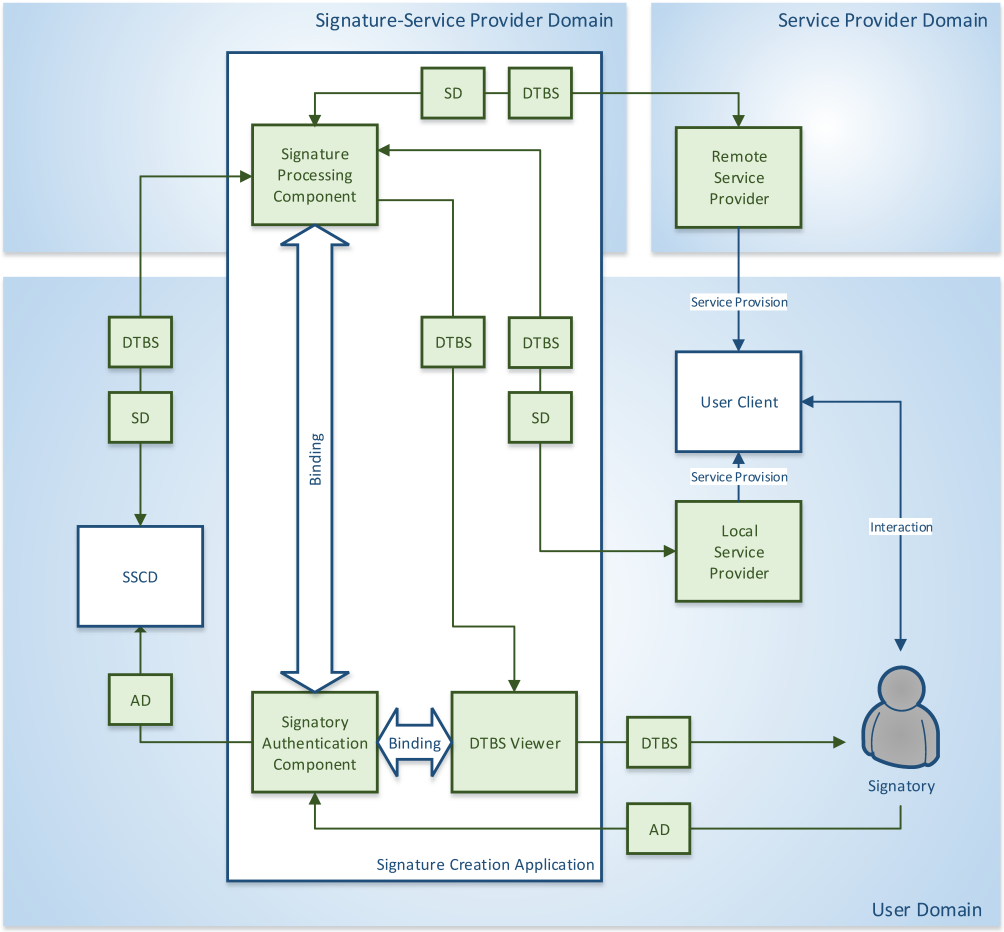

Figure 4.22: Mapping of identified assets to Implementation Variant C.

components. However, components are assigned to different domains depending on the concrete implementation variant. The capability to securely realize components in different domains is hence a crucial aspect for the conducted security assessment.

Realizing components in different domains also affects communication paths between these components. Communication paths between components of the same domain need to be realized differently than communication paths between components of different domains. As each implementation variant is unique regarding its assignment of components to different domains, each variant relies on different communication paths. This has to be considered for the conducted security assessment.

4.4.4 Identification of Relevant Components and Communication Paths

By mapping identified assets to concrete implementation variants, a preliminary list of components and communication paths, at which assets are potentially exposed to attacks, can be derived. This list can be obtained from the extended architectures of the four implementation variants simply by extracting all marked components and communication paths. In this section, this preliminary list is consolidated. The resulting consolidated list will later be used to systematically assess the security of different implementation variants for mobile signature solution.

Consolidation of the preliminary list involves two steps. First, components are removed from the preliminary list that are assumed to be secure according to predefined assumptions. Second, several

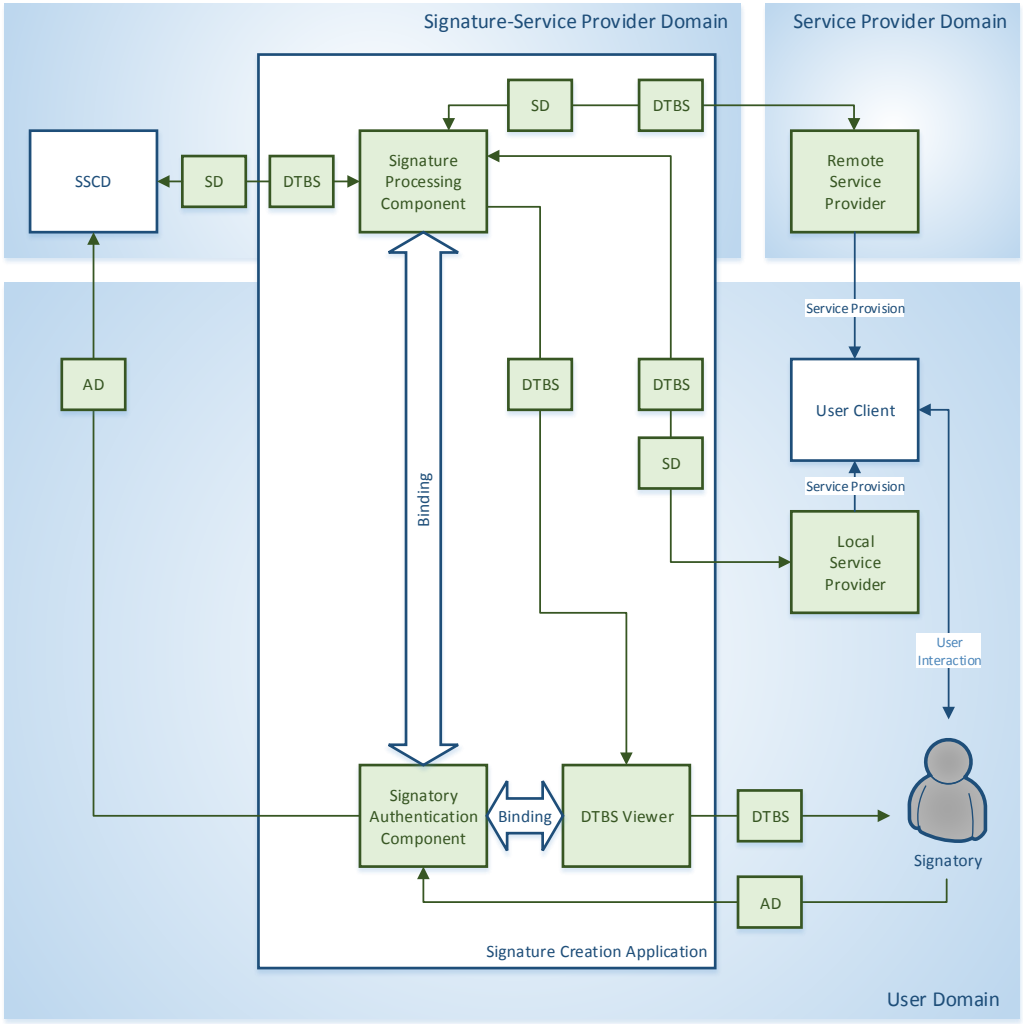

Figure 4.23: Mapping of identified assets to Implementation Variant D.

communication paths between components are combined and assigned to a few classes of communication paths. This reduces the total number of different communication paths and facilitates direct comparisons between implementation variants.

Consolidation of the preliminary list of components and communication paths finally yields four relevant components. These components need to be considered for the conducted security assessment.

- **Local Service Provider:** The Local Service Provider is the source of the DTBS and the final receiver of the SD. Thus, the Local Service Provider represents a potential target for attacks on the assets DTBS and SD. This applies irrespective of the implementation variant, as the Local Service Provider is realized locally in all four implementation variants.

- **Signature Processing Component:** The Signature Processing Component acts as intermediary between the Service Provider and the SSCD. The Signature Processing Component can be implemented locally or remotely. If implemented locally, the Signature Processing Component represents a potential target for attacks on the assets DTBS and SD.

- **Signatory Authentication Component:** The Signatory Authentication Component acts as intermediary between the Signatory and the SSCD and reads authentication data from the Signatory.

Thus, the Signatory Authentication Component represents a potential target for attacks on the asset AD. As the Signatory Authentication Component needs to be implemented locally in any case, the Signatory Authentication Component is a potential target of attacks for all four implementation variants.

- **DTBS Viewer:** The DTBS Viewer displays the DTBS to the Signatory prior to signing. Thus, this component represents a potential target for attacks on the asset DTBS. Similar to the Signatory Authentication Component, the DTBS Viewer needs to be implemented locally, as it requires a direct user interface to the Signatory. Hence, this component represents a potential target for attacks on the asset DTBS, irrespective of the underlying implementation variant.

In addition to these components, the following classes of communication paths are relevant and need to be considered for the conducted security assessment:

- **Inter-application and inter-process communication:** Communication between applications or between processes within one application need to be considered for the Local Approach and for the Remote-SSCD Approach. This is mainly due to the fact that these approaches implement the Signature Processing Component, which acts a central hub between the Signatory, the SSCD, and the Service Provider, on the local end-user device. As the local Signature Processing Component needs to communicate with the Local Service Provider to exchange the assets DTBS and SD, there is a need for inter-application communication. In case the Signature Processing Component and the DTBS Viewer are implemented by separate applications, inter-application communication is also required to transmit the DTBS from the Signature Processing Component to the DTBS Viewer. In case the Signature Processing Component and the DTBS Viewer are integrated into one single component, application-internal inter-process communication needs to be considered, as the asset DTBS is then transmitted from one application-internal process to another.

- **Communication between local and remote software:** This communication path is mainly found between the Service Provider and the Signature Processing Component. In case of a locally implemented Signature Processing Component, communication between the Remote Service Provider and the Signature Processing Component requires communication between local and remote software. Vice versa, in case of a remotely implemented Signature Processing Component, communication between the Local Service Provider and the Signature Processing Component also requires this kind of communication. In any case, the assets DTBS and SD are transmitted over this communication path.

- **Communication between local software and local SSCD:** This communication path needs to be considered for implementation variants that rely on a locally implemented SSCD. This basically applies to the Classical Approach represented by Implementation Variant A and to the SIM Approach represented by Implementation Variant C. However, there is a significant difference between these two approaches. Following the Classical Approach, all three assets are transmitted over this communication channel. As the Signature Processing Component is remotely implemented according to the SIM Approach, only the asset AD is transmitted over this communication channel in this scenario.

- **Communication between user and local software:** A user interface between the Signatory and the Signatory Authentication Component is required in each implementation variant. Hence, the security of this communication path needs to be considered irrespective of the followed approach.

- **Communication between local software and remote SSCD:** This communication path needs to be considered for all implementation variants that rely on a remote SSCD. This applies to the Remote-SSCD Approach and the Server-HSM Approach. Following the Remote-SSCD Approach,

the Signature Processing Component is implemented in the User Domain. Hence, all three assets are transmitted over this communication path in this scenario. In contrast, only the asset AD is transmitted over this communication channel when following the Server-HSM Approach.

- **Communication between remote software and local SSCD:** This communication path is relevant for the SIM Approach only, as only this implementation variant relies on a remote Signature Processing Component and a local SSCD. Over this communication channel, the assets DTBS and SD are transmitted.

Figure 4.24 summarizes identified components and communication paths. Furthermore, Figure 4.24 shows, which components and which communication paths are relevant for the four implementation variants in order to protect the three identified assets. From this table it becomes apparent that Implementation Variant C and Implementation Variant D are beneficial in terms of the total number of different components and communication paths, on which assets are potentially exposed to attacks. For these implementation variants, the lowest number of components and communication paths need to be considered. Hence, the SIM Approach and the Server-HSM Approach are advantageous in terms of security from a pure conceptual perspective.

	Implementation Variant A (Classical Approach)			Implementation Variant B (Remote-SSCD Approach)			Implementation Variant C (SIM Approach)			Implementation Variant D (Server-HSM Approach)		
	SD	DTBS	AD	SD	DTBS	AD	SD	DTBS	AD	SD	DTBS	AD
Components												
Local Service Provier	X	X		X	X		X	X		X	X	
Signature Processing Component	X	X		X	X							
Signatory Authentication Component			X			X			X			X
DTBS Viewer		X			X			X			X	
Communication Paths												
Inter-application and inter-process communication	X	X		X	X							
Communication between local and remote software	X	X		X	X		X	X		X	X	
Communication between local software and local SSCD	X	X	X						X			
Communication between user and software component			X			X			X			X
Communication between local software and remote SSCD				X	X	X						X
Communication between remote software and local SSCD							X	X				

Figure 4.24: Different components and communication paths need to be considered for different implementation variants of mobile signature solutions.

This becomes even more apparent, when considering scenarios that rely on a Remote Service Provider only and ignore the possibility of a Local Service Provider. Results of an identification of relevant components and communication paths based on such scenarios are shown in Figure 4.25. Considering a Remote Service Provider only reduces the number of components, on which assets are potentially exposed

to attacks, as one local component, i.e. the Local Service Provider, is eliminated. For the Server-HSM Approach represented by Implementation Variant D, this results in only two components and three communication paths that need to be secured in order to protect all assets. In contrast, for e.g. Implementation Variant A, which represents the Classical Approach, four components and seven communication paths need to be protected.

	Implementation Variant A (Classical Approach)			Implementation Variant B (Remote-SSCD Approach)			Implementation Variant C (SIM Approach)			Implementation Variant D (Server-HSM Approach)		
	SD	DTBS	AD	SD	DTBS	AD	SD	DTBS	AD	SD	DTBS	AD
Components												
Signature Processing Component	X	X		X	X							
Signatory Authentication Component			X			X			X			X
DTBS Viewer		X			X			X			X	
Communication Paths												
Inter-application and inter-process communication		X			X							
Communication between local and remote software	X	X		X	X			X			X	
Communication between local software and local SSCD	X	X	X						X			
Communication between user and software component			X			X			X			X
Communication between local software and remote SSCD				X	X	X						X
Communication between remote software and local SSCD							X	X				

Figure 4.25: Ignoring the possibility of a Local Service Provider reduces the number of relevant components and communication paths that need to be considered.

In summary, several differences in the number of affected components and communication paths can be observed between the different implementation variants. This is expectable to a certain extent, as the number of affected components and communication paths obviously corresponds to the number of locally implemented components. As the Server-HSM Approach implements both the SSCD and the Signature Processing Component remotely, this solution also shows the lowest number of components and communication paths, on which assets are potentially exposed to attacks.

However, the number of components and communication paths, on which assets are potentially exposed to attacks is not the only relevant measure for the security of a particular implementation variant. Opportunities to protect identified components and communication paths with currently available technologies is an even more important aspect. These opportunities are hence analyzed in the following section in more detail considering technologies available on current mobile end-user devices.

4.4.5 Component and Communication Path Specific Assessment

Figure 4.24 and Figure 4.25 compare the four approaches to create electronic signatures on mobile end-user devices from a technology-independent perspective. This is achieved by identifying components and communication paths, on which the three assets AD, DTBS, and SD are potentially exposed to attacks.

However, this comparison does not consider opportunities and limitations to secure these components and communication paths by means of currently available technologies. In this section, identified components and communication paths are hence mapped to existing technologies that are currently available on mobile end-user devices. This way, the security of these components and communication paths on current mobile end-user devices is assessed. Conducted assessments base on related work on the security of mobile end-user devices and on own contributions, which have been presented in Chapter 3 in more detail. Obtained component-specific assessment results are later used to assess the security of the four implementation variants for mobile signature solutions.

4.4.5.1 Local Service Provider

Considering the rather limited opportunities to provide services on mobile end-user devices, implementations of Local Service Providers will in most cases be based on typical mobile apps. On all major platforms, the basic security of mobile apps and of data processed by these apps is assured by means of integrated features of the mobile operating system. Such features include for instance sandboxing mechanisms, which prevent applications running on a mobile device from accessing data and resources of other applications running on the same device. Compared to classical operating systems, mobile operating systems provide enhanced means for a logical separation of applications. An overview of currently available security features of different mobile operating systems has been provided e.g. in Zefferer et al. [2013b]. In general, sandboxing and other integrated security features work reliably on current mobile end-user devices, as long as attackers do not gain root access to the mobile operating system. Otherwise, integrated security features must not be assumed to be able to protect data and resources of installed applications any longer. We have discussed implications that need to be considered for attackers with root access in Zefferer and Teufl [2013].

4.4.5.2 Signature Processing Component

For locally implemented Signature Processing Components, similar considerations apply as for Local Service Providers. Due to the system architecture of major mobile operating systems, implementation alternatives for software components are basically restricted to mobile apps. Hence, also the Signature Processing Component, if implemented locally, needs to be realized as mobile app. This implies that for the security of locally implemented Signature Processing Components basically the same security considerations apply as for the Local Service Provider. Integrated security features of mobile operating systems protect stored and processed data from access by other software components. However, these security features can theoretically be circumvented by attackers that gain root access to the mobile operating system. Hence, the security of data being stored and processed by a locally implemented Signature Processing Component must not be taken for granted.

4.4.5.3 Signatory Authentication Component

Similar to the Local Service Provider, the Signatory Authentication Component needs to be implemented locally on the mobile end-user device in any case, as this component requires a user interface to the Signatory. Being implemented locally, similar considerations apply as for the local components discussed above. Hence, also data processed by the Signatory Authentication Component is theoretically vulnerable to attack scenarios, in which attackers gain root access to the underlying operating system. In addition to other local components, two special aspects need to be considered for this component.

First, the functionality of this component is in most cases rather simple and basically limited to the reading and forwarding of authentication data. This allows for the use of alternative technologies to implement this functionality. For instance, current SIM-based signature solutions such as the Estonian

Mobiil-ID[17] usually make use of the SIM Application Toolkit to implement the functionality of the Signatory Authentication Component. The SIM Application Toolkit allows the MNO to access the SIM. Furthermore, it also allows the SIM to interact with the user, if this is supported by the operating system. This way, the use of mobile apps, which are potentially prone to malware, can be prevented. However, an attacker or malware with unlimited root access to the mobile operating system must be assumed to be able to compromise the security of solutions based on the SIM Application Toolkit.

Second, the functionality of the Signatory Authentication Component might need to be enhanced in case of remote SSCDs. In this case, the requirement for sole control over signature-creation data, i.e. cryptographic keys, might require the Signatory Authentication Component to implement additional authentication means. For instance, the Austrian Mobile Phone Signature, which relies on a remote SSCD, requires the Signatory to receive an SMS message containing an One-Time Password (OTP) and to forward this OTP to the SSCD in order to complete the authentication process. Hence, depending on the authentication scheme used by the particular implementation, the functionality of the Signatory Authentication Component might go beyond the simple reading and forwarding of static authentication data such as PINs or passwords. Implementation of required additional functionality potentially enables new attack scenarios. For instance, in case of OTP-based authentication schemes relying on SMS technology, the required processing of SMS messages on mobile end-user devices potentially raises new security issues that need to be considered.

4.4.5.4 DTBS Viewer

Similar to the Signatory Authentication Component, the DTBS Viewer needs to be implemented locally in any case, as it requires a direct user interface to the Signatory. Hence, similar considerations apply as for other locally implemented components. The security of data being processed by this component is potentially prone to attacks that rely on full root access to the mobile operating system. A key functionality of the DTBS Viewer, which needs to be especially considered for this component, is the reliable displaying of data to the Signatory. This aspect is discussed in more detail in Section 4.4.5.8.

4.4.5.5 Inter-Application and Inter-Process Communication

Communication between different mobile apps on one device and between different processes within one app have been identified as crucial, especially for implementation variants that heavily rely on locally implemented components. Support for inter-application and inter-process communication varies between different mobile operating systems. Similarly, also the security of these communication paths varies between different operating systems. Capabilities, limitations, and security issues of inter-application and inter-process communication on current mobile operating systems have for instance been discussed by Chin et al. [2011]. We have also contributed to this topic of scientific interest in Zefferer et al. [2013b]. According to these works, Apple iOS and Microsoft Windows Phone 8 provide only limited capabilities for apps and processes to communicate with each other and to exchange data. Due to these limited capabilities, potential attack vectors are also limited. Hence, these two platforms provide a sufficient level of security for data being exchanged between apps and processes. Again, the situation is sort of different when attackers with unlimited root access to the mobile operating system are considered as well. In this case, the security of data being exchanged between different apps and processes on one and the same mobile device must not be assumed to be secure any longer.

In contrast to Apple iOS or Microsoft Windows Phone 8, the mobile operating system Google Android provides a higher degree of flexibility regarding inter-application and inter-process communication. On Android, the exchange of information and data between applications and processes relies on so-called Intents. An Intent is basically a simple data object that can be exchanged between internal components.

[17] http://mobiil.id.ee/

A more detailed introduction to the concept of Intents has been provided by Chin et al. [2011]. The flexibility of the concept of Intents enables more flexible and more powerful applications. At the same time, this flexibility can also be employed by malware to intercept data being exchanged between applications and processes. We have discussed this issue in more detail in Zefferer et al. [2013b]. It is important to note that on Android attacks on the communication path between apps and between processes are also feasible without root access to the operating system.

4.4.5.6 Communication Between Local and Remote Software

Communication between local and remote software components has been common practice on classical end-user devices for decades. Time-tested technologies and protocols such an SSL/TLS [Network Working Group, 2008] exist that enable a secure data exchange between local and remote components. Most of these technologies and protocols can also be used on current mobile end-user devices. For instance, support for SSL/TLS is provided on all major mobile platforms according to publicly available documentation[18,19,20]. Thus, if implemented correctly, local components can securely access remote components and services by means of approved protocols.

4.4.5.7 Communication Between Local Software and Local SSCD

Access to a local SSCD from local software components is feasible depending on the implementation of the SSCD and on the mobile operating system. While capabilities to access SSCDs from local software components increase feasibility, these capabilities also raise additional security issues. If local software components have the technical opportunity to access local SSCDs, also malware residing on the same device has the opportunity to do so. The feasibility of attacks depends also on the concrete implementation of the SSCD. For instance, SIM-based SSCDs usually do not support access from third-party apps running on the mobile end-user device, as the SIM Application Toolkit is required to exchange data with the SIM-based SSCD. In this context, SIM-based SSCDs provide a higher level of security than other local SSCD implementations, which can be accessed by means of APIs provided by the operating system. Of course, SIM-based SSCD implementations provide a lower level of feasibility, as access to SIMs from local software components is usually impossible. In general, the security of the communication channel between local software and a local SSCD depends on the security of the mobile end-user device. Hence, this communication channel must not be assumed to be secure.

The security of this communication channel can be improved, if the SSCD implements some kind of secure-messaging protocol. In this case, all communication between the SSCD and an accessing software component can be protected by cryptographic means. These means can assure the confidentiality and integrity of data exchanged with the SSCD. Communication with SSCDs by means of secure-messaging mechanisms is common practice. For instance, several smart cards support secure messaging. An example is the Spanish national eID card[21], which requires an accessing application to establish a secure channel first.

4.4.5.8 Communication Between User and Local Software

This communication path is used by the DTBS Viewer to display data to the Signatory and by the Signatory Authentication Component to obtain required authentication data from the Signatory. For both use cases, it has to be stated that secure input and output capabilities are not available on current mobile end-user devices.

[18] http://msdn.microsoft.com
[19] https://developer.apple.com/devcenter/ios/index.action
[20] http://developer.android.com/index.html
[21] http://www.dnielectronico.es/

A potential future solution to this problem is the ARM TrustZone technology[22]. This technology makes use of integrated secure hardware components to provide a separate secure execution environment on potentially insecure platforms. This secure environment includes capabilities for secure user input and output. Hence, this technology represents a promising future solution to this problem. However, this technology is still in its infancies and not available for the majority of current mobile end-user devices.

Up to now, communication between users and software components running on mobile end-user devices must not be assumed to be secure. This basically holds true for all major mobile operating systems and available mobile devices. However, there are again certain differences between diverse operating systems. While Apple iOS and Microsoft Windows Phone 8 provide only few capabilities to modify or adapt integrated input mechanisms, Google Android e.g. enables users to install and use alternative keyboards. A large variety of alternative keyboards are offered in the official Google Play Store[23]. This flexibility raises additional security issues, as alternative keyboards can be potentially equipped with malware to spy on data entered by users. Even though Google Android is especially prone to unauthorized interception of user input, user input must not be assumed to be secure on other mobile platforms either. For instance, a recent article reports on a security vulnerability on iOS devices, which facilitates realization of key-logging functionality [Goodin, 2014]. Hence, it can be concluded that secure input and output capabilities are currently missing on all major platforms. In particular, Google Android seems especially prone to attacks on data exchanged between the users and the mobile device.

4.4.5.9 Communication Between Local Software and Remote SSCD

This communication path is relevant for implementation variants that rely on a remote SSCD and hence require cross-domain communication between local software components and this SSCD. This communication path requires the use of additional modules that extend supported communication interfaces of the SSCD and the local software component. This way, these modules enable the exchange of data using cross-domain communication protocols. The conducted feasibility assessment has shown that integration of such modules is possible both for local software components and for remote SSCDs.

Regarding security, integration of additional modules that enable cross-domain communication does only raise minor additional challenges. Local software components can be easily enhanced by such modules. Being an integral part of the local software component, these modules are protected by the same security features of the operating system as the original software component. Even though these security features do not provide full security, integration of additional modules does at least not reduce the overall security. An additional module that enables cross-domain communication is also required for the remote SSCD. This module translates the used cross-domain protocol into a protocol that is supported by the SSCD. As this module needs to be implemented remotely, this component is secure according to the made assumption that remote components are secure by definition. Hence, required modules that enable communication between local software and remote SSCDs do not reduce security of the entire solution.

In addition to the modules that enable cross-domain communication, also the communication path between these two modules needs to be secured. An adequate level of security can be achieved by choosing the right communication protocol. As both the local and the remote module represent software components, the same considerations apply as for the communication path between local and remote software components. The security of this communication path has been discussed in Section 4.4.5.6. According to this discussion, communication paths between software components can be implemented securely.

In summary it can be concluded that the communication path between local software components and remote SSCDs can be implemented securely with currently available technologies. Even though the

[22] http://www.arm.com/products/processors/technologies/trustzone/index.php
[23] https://play.google.com/store/search?q=keyboard&c=apps

need for cross-domain communication between local software and a remote SSCD raises the need for additional modules, this does not reduce the security of the overall solution.

4.4.5.10 Communication Between Remote Software and Local SSCD

Communication between remote software components and local SSCDs requires cross-domain communication as well. Hence, again additional modules are required to enhance the two communicating parties with the capability to use cross-domain communication protocols. As remote components are assumed to be secure, only the local realization of the required additional module needs to be considered in detail.

The conducted feasibility assessment has shown that this local module can either be implemented by a mobile app with access to the SSCD or by the mobile operating system. For the first case, the same security considerations apply as for the communication path between local software and local SSCDs. The security of this communication path is problematic as discussed in Section 4.4.5.7. For the latter case, the security of the additional module depends on the level of security provided by the mobile operating system. As no operating system is absolutely secure, this module must not be assumed to be secure either.

In addition to the two modules themselves, also the communication path between these two modules needs to be secured. An adequate level of security can be achieved by relying on suitable communication protocols. In case the local module is implemented as a mobile app, the same considerations apply as for the communication path between local and remote software components. This communication path can be secured with existing protocols and technologies such as SSL/TLS. In case the local module is implemented by the mobile operating system, also alternative communication channels can be employed. For instance, most current SIM-based signature solutions rely on the mobile network and on an SMS-based exchange of data between remote software and the local SSCD. If alternative communication channels and technologies are used, their security also affects the security of transmitted assets. For the special case of SIM-based signature solutions and their employed communication technology, several security issues have been identified in the past [Donohue, 2013]. In general, securing the communication path between the local and remote modules that enable cross-domain communication might be challenging.

In summary it can be concluded that a secure communication path between remote software and a local SSCD is difficult to achieve. This is mainly due to the fact that remote communication with local SSCDs potentially requires an additional local module. Depending on the implementation of this module and on the used technology for cross-domain communication, reliance on the communication path between remote software and local SSCDs hence potentially reduces the security of transmitted assets.

4.4.6 Assessment of Implementation Variants

Available technologies and the current state of the art affect the security of relevant components and communication paths of mobile signature solutions. Approaches to realize mobile signature solutions can be classified into four implementation variants. As they all rely on different components and communication paths, different security aspects need to be considered for each variant. In this section, the security of the four implementation variants is compared based on the results of the conducted component-specific security assessment.

For this purpose, Figure 4.24 on page 126 and Figure 4.25 on page 127 can be used as a starting point. These figures summarize involved components and communication paths for each implementation variant and hence enable a comparison on conceptual level. This comparison reveals that some components and communication paths are used in any case. This applies to the three local components Local Service Provider, Signatory Authentication Component, and DTBS Viewer. As shown in Figure 4.24, there are no differences between the four implementation variants with regard to assets being potentially exposed

to attacks by these three components. The same applies to two communication paths. Means for communication between local and remote software, as well as between users and software components are required in any case. For a comparative security assessment, components and communication paths, for which no differences can be identified between different implementation variants, can be omitted. Hence, focus on the remaining components and communication paths is sufficient to compare the four variants on conceptual level. Remaining relevant components and communication paths that need be considered for a comparative security assessment are highlighted in Figure 4.26.

	Implementation Variant A (Classical Approach)			Implementation Variant B (Remote-SSCD Approach)			Implementation Variant C (SIM Approach)			Implementation Variant D (Server-HSM Approach)		
	SD	DTBS	AD	SD	DTBS	AD	SD	DTBS	AD	SD	DTBS	AD
Components												
Local Service Provier	X	X		X	X		X	X		X	X	
Signature Processing Component	**X**	**X**		**X**	**X**							
Signatory Authentication Component			X			X			X			X
DTBS Viewer		X			X			X			X	
Communication Paths												
Inter-application and inter-process communication	**X**	**X**		**X**	**X**							
Communication between local and remote software	X	X		X	X		X	X		X	X	
Communication between local software and local SSCD	**X**	**X**	**X**						**X**			
Communication between user and software component			X			X			X			X
Communication between local software and remote SSCD				**X**	**X**	**X**						**X**
Communication between remote software and local SSCD							**X**	**X**				

Figure 4.26: For a comparative analysis the number of relevant components and communication paths can be reduced.

For the remaining relevant components and communication paths, several differences can be identified between the four implementation variants. Figure 4.26 shows that the Signature Processing Component is the only component that needs to be considered for direct comparisons. For signature solutions following the Classical Approach or the Remote-SSCD Approach, the security of the assets DTBS and SD depends on the security of the Signature Processing Component. Hence, these solutions are conceptually disadvantageous compared to solutions following the SIM Approach or the Server-HSM Approach, for which the security of the assets DTBS and SD are independent from the Signature Processing Component.

This finding is also backed by the conducted component-specific security assessment. According to this assessment, the security of data being processed by a locally implemented Signature Processing Component must not be taken for granted on current mobile platforms. Attackers with root access to the mobile operating system pose a serious threat to the security of a locally implemented Signature Processing Component and hence also to the assets DTBS and SD. With regard to the Signature Processing

Component, solutions following the Classical Approach or the Remote-SSCD Approach are hence also disadvantageous from a realization perspective.

Similar results can be derived with regard to inter-application and inter-process communication. Figure 4.26 shows that this communication path is required by the Classical Approach and the Remote-SSCD Approach. In contrast, for signature solutions relying on the SIM Approach or the Server-HSM Approach, no assets are exposed to threats on this communication path. Hence, the Classical Approach and the Remote-SSCD Approach are again disadvantageous from a conceptual perspective.

This result is again backed by the conducted component-specific security assessment. In Section 4.4.5.5, especially inter-application communication has been identified as potential risk on certain mobile platforms. Hence, the Classical Approach and the Remote-SSCD Approach are disadvantageous also from a realization perspective.

Comparison of the four implementation variants by means of the Signature Processing Component and by means of inter-application and inter-process communication clearly yields the SIM Approach and the Server-HSM Approach to be advantageous. The other two approaches represented by Implementation Variant A and Implementation Variant B suffer from the requirements to secure a local Signature Processing Component and to provide secure means for inter-application communication. Both requirements are difficult to satisfy on current mobile end-user devices. The remaining three communication paths, for which Figure 4.26 shows differences between the four implementation variants, are related to data exchange between software components and the SSCD. Counting the occurrences of assets on these communication paths indicates that the Server-HSM Approach is advantageous from a conceptual perspective. Following this approach, only the asset AD is transmitted once over one of these three communication channels. For all other approaches, all three assets are present at least once on one of the three communication paths.

Considering the conducted component-specific assessment also leads to useful results. Figure 4.26 shows that both the Classical Approach and the SIM Approach require communication between local software and a local SSCD. The component-specific assessment has revealed that data transmitted over this communication path is potentially prone to attacks, as attackers with access to the mobile device can compromise transmitted data. This reduces the security of signature approaches relying on this communication channel. Hence, the Classical Approach and the SIM Approach are disadvantageous to the Remote-SSCD Approach and the Server-HSM Approach, which do not transmit assets over this communication channel.

Figure 4.26 also shows that the SIM Approach is sort of special as it is the only approach that transmits assets over a communication channel between remote software and a local SSCD. The component-specific assessment of this communication path has shown that this type of cross-domain communication potentially reduces security, as an additional local software module is required. Hence, the need to transmit assets over this communication path reduces the security of the SIM Approach.

Communication between local software and remote SSCDs is required by signature solutions following the Remote-SSCD Approach and the Server-HSM Approach. While the former transmits all three identified assets over this communication path, only the asset AD is transmitted over this path when the Remote-HSM Approach is followed. The Remote-HSM Approach is hence conceptually advantageous compared to the Remote-SSCD Approach. The component-specific assessment of this communication path has shown that this communication path can be implemented securely using approved communication protocols. From a realization perspective, reliance on this communication channel does hence not reduce the achievable level of security.

From a comparison of the four implementation variants for mobile signature solutions, the following conclusions can be drawn. Comparison from a conceptual perspective yields solutions following the Server-HSM Approach to be advantageous compared to other implementation variants. This can be explained by the fact that this approach removes as many components as possible from potentially insecure local end-user devices. Taking into account the realization perspective as well, also the SIM Approach

turns out to provide an adequate level of security, if secure cross-domain communication technologies are used and additionally required local modules are realized securely. Solutions following the Classical Approach or the Remote-SSCD Approach suffer from the rather large number of components being implemented locally. Therefore, these approaches are disadvantageous in terms of security both from the conceptual and the realization perspective. Hence, it can be concluded that from a security perspective, development of a signature solution for mobile end-user devices should follow the Server-HSM Approach. Alternatively, also the SIM Approach can be followed, if several requirements regarding the secure realization of critical components are satisfied.

4.5 Usability Assessment

Usability has been identified as relevant pillar for successful m-government solutions in Part I of this thesis. According to the International Organization for Standardization (ISO), usability is *'the effectiveness, efficiency and satisfaction with which specified users achieve specified goals in particular environments'* [ISO/IEC, 2010]. According to this definition, usability contributes to user satisfaction and therefore directly to user acceptance and success. The aim to develop an accepted and successful mobile signature solution requires this solution to provide a sufficient level of usability. In many cases, usability is however not only a matter of well-designed user interfaces, but also heavily influenced by the solution's underlying concepts and architecture. It is hence reasonable to consider usability aspects when determining the best approach for mobile signature solutions. Therefore, this section assesses the usability of the four identified implementation variants for mobile signature solutions in detail. Results of this usability assessment will be used to determine the best approach to create electronic signatures on mobile end-user devices.

4.5.1 Methodology

Hornbæk [2006] has shown that measuring usability is a complex task that raises various challenges. Approved usability-measurement approaches such as thinking-aloud tests have mainly been designed for the purpose of evaluating a particular solution or product. Instead of evaluating a certain solution or implementation, this assessment aims to compare four technology-agnostic implementation variants mainly on an abstract and conceptual level. Therefore, existing approaches are not suitable to meet the requirements of this usability assessment.

To achieve a comparative usability assessment, three consecutive steps are carried out. In the first step, relevant usability aspects that apply to all implementation variants are identified. The set of relevant aspects is obtained from a survey on related work dealing with the identification of crucial aspects for usable IT systems. Additionally, results of own research on the usability of signature-creation solutions are considered. This way, usability aspects are covered that are specific to signature-creation solutions. In the second step, the usability of the four implementation variants is assessed by means of the identified usability aspects. For each aspect, a ranking of the four implementation variants is derived. Finally, obtained assessment results are compared to determine the most usable implementation variant for mobile signature solutions.

In general, the methodology resembles the one that has been followed to assess the security of the four different implementation variants for mobile signature solutions. In addition to the security assessment, the conducted usability assessment also ranks the four implementation variants according to their capability to meet relevant usability aspects. Ranking the four implementation variants has been omitted during the security assessment, as the security of the four implementation variants also heavily depends on the used mobile platform. This is not the case for the usability of the four implementation variants. As their usability is to a large extent independent from the used mobile platform, a ranking of the four implementation variants according to relevant usability aspects can be derived.

4.5.2 Identification of Relevant Usability Aspects

In this section, relevant usability aspects are identified. In general, usability aspects can be defined on different levels of abstraction. For instance, according to Frøkjær et al. [2000] usability comprises the aspects effectiveness, efficiency, and satisfaction. These rather abstract usability aspects can be split into more detailed aspects depending on the particular field of application. For instance, usability factors for mobile handsets have been identified by Ketola and Röykkee [2001] including the aspects integration of functionality, availability, utility and ease of using services, readiness for use, informativity, usefulness of support material, and interoperability. In addition, Klockar et al. [2003] consider the size of soft keys and screen, the layout of navigation keys, and the menu structure as crucial for mobile phones. This concrete example from the field of mobile phones shows that identification of relevant usability aspects is a complex task in practice and heavily depends on the concrete use case. The general problem of identifying relevant usability factors has been discussed by Heo et al. [2009] by means of the concrete case of mobile phones. Nevertheless, identification of relevant usability factors is also crucial for other use cases and defines the first relevant step of a usability assessment.

As all four implementation variants for mobile signature solutions are intended for mobile end-user devices, usability factors for mobile apps are an appropriate starting point for the identification of relevant usability aspects. Harrison et al. [2013] have provided a literature survey on the definition of relevant usability aspects for mobile applications. Based on this literature survey, Harrison et al. [2013] have proposed the PCMAD usability model, which defines the factors effectiveness, efficiency, satisfaction, learnability, memorability, errors, and cognitive load as crucial for the usability of mobile applications. These usability aspects are used as basis for the conducted usability assessment of the four implementation variants for mobile signature solutions.

Recent usability assessments of concrete signature solutions have shown that for these solutions several additional aspects need to be considered [Zefferer and Krnjic, 2012b]. Most of these aspects are related to the use of SSCDs, which potentially introduces additional hurdles for users. In order to consider these additional aspects, the basic set of usability aspects for mobile applications defined by Harrison et al. [2013] is adapted and extended accordingly. This finally leads to the following set of relevant usability aspects:

- **Effectiveness:** Effectiveness describes the *'ability of a user to complete a task in a specified context'* [Harrison et al., 2013]. In the case of mobile signature solutions, this refers to the user's ability to create a qualified electronic signature. As such, this usability factor also includes the requirement for service availability. In the context of mobile applications, this usability factor is influenced e.g. by the fact whether or not the assessed solution is also applicable without access to the mobile network, or if the solution supports roaming.

- **Efficiency:** Harrison et al. [2013] define the speed and the accuracy, with which an intended task can be completed by the user, as relevant aspects of the factor efficiency. In the context of mobile signature solutions, efficiency is hence indirectly proportional to the complexity of a signature-creation operation from the user's point of view.

- **Satisfaction:** Harrison et al. [2013] define the usability factor Satisfaction as *'the perceived level of comfort and pleasantness afforded to the user through the use of software'*. As for all applications, the provided user interface of mobile signature solutions mainly influences the perceived level of satisfaction. In addition, the following two aspects, which are rather special to signature-creation solutions, also need to be considered. We have shown and discussed the relevance of these aspects with relation to signature-creation solutions in Zefferer and Krnjic [2012b].

  - **Hardware independence:** Conducted usability assessments of existing signature solutions have shown that the requirement to use certain hardware represents a potential hurdle in terms

of usability. This applies for instance to smart card based signature solutions, which require users to acquire, maintain, and use additional hardware in the form of card-reading devices. Even if card-reading devices are available, users often refuse the use of smart cards, as they are not familiar with this technology. Hardware independence is hence a crucial usability factor for mobile signature solutions.

  - **Software independence:** For this aspect, similar considerations apply as for the usability factor hardware independence. Conducted usability assessments of existing signature solutions have shown that the requirement to install and maintain additional software often represents a usability-reducing factor. In the case of smart card based signature solutions, software being installed on the user's local device often represents a potential source of problems especially for technically inexperienced users. Software independence, i.e. the avoidance of additional software components, is hence a crucial usability factor for mobile signature solutions.

- **Learnability:** Flood et al. [2013] have shown that average users spend only five minutes or less to learn using a new mobile application. Learnability is hence an important aspect for mobile solutions that aim to attract long-term attention of users.

- **Memorability:** Memorability, i.e. *'the ability of a user to retain how to use an application effectively'* [Harrison et al., 2013] is an important aspect. This especially applies to mobile signature solutions, as such solutions must be expected to be less frequently used than other mobile applications such as web browsers or social-networking apps.

- **Error Robustness:** According to Nielsen [1993], usable systems need to have a low error rate that reduces the number of errors made by users. Furthermore, catastrophic errors must be prevented. Defining a rather abstract requirement, this usability aspect applies to nearly all technical systems and hence also includes mobile signature solutions.

- **Cognitive Load:** This usability aspect has been introduced by Harrison et al. [2013] especially for assessing the usability of mobile applications. In short, cognitive load describes the impact that using the mobile application has on the performance of other tasks the user is performing in parallel. The higher their cognitive load, the more attention mobile applications demand from the user.

4.5.3 Assessment of Implementation Variants

Related scientific work on the identification of relevant usability factors reveals that various aspects need to be considered in the course of a comprehensive usability assessment. In this section, the four identified implementation variants for mobile signature solutions are assessed by means of each relevant usability aspect. The conducted usability assessment mainly focuses on the conceptual perspective. This enables a direct comparison of different approaches to create signatures on mobile end-user devices. In addition, usability limitations imposed by currently available technologies are also considered if necessary. Obtained aspect-specific results are subsequently combined in order to determine the most usable approach.

4.5.3.1 Effectiveness

According to the definition of this usability aspect, the effectiveness of a signature solutions is closely related to its feasibility. Only if a solution is feasible, it can be effective, i.e. allow the user to complete the intended functionality. The conducted feasibility assessment has shown that all implementation variants enable the Signatory to create qualified electronic signatures. Limitations regarding feasibility are mainly imposed by required technologies that are not available on certain mobile platforms. The Classical

Approach represented by Implementation Variant A defines the highest requirements for mobile end-user devices. Hence, this approach is also disadvantageous in terms of effectiveness. From the remaining three approaches, the Server-HSM Approach represented by Implementation Variant D shows the highest degree of feasibility and hence also the highest degree of effectiveness.

Besides the pure technical feasibility, also service availability is an important factor for effectiveness. In the context of mobile signature solutions, two concrete aspects need to be considered. First, reliance on remote components potentially affects service availability. The use of remote services usually requires the mobile device to be connected to the Internet. This requirement reduces service availability in scenarios, in which the mobile device is offline. As the Classical Approach implements all relevant components locally, this solution provides the highest degree of service availability in offline scenarios. Second, limitations regarding service availability might also be caused by reliance on special communication technologies and protocols. According to the conducted feasibility assessment, this potentially applies to the communication path between remote software components and local SSCDs. For instance, several SIM-based mobile signature solutions require the Signatory's mobile network operator to establish a connection between the remote Signature Processing Component and the local SIM. Depending on the used communication protocol, this can raise the requirement for the Signatory to remain within the network of the MNO and hence reduces service availability in roaming scenarios. As the communication path between remote software components and local SSCDs is only used by the SIM Approach, this limitation applies to this approach only. All other communication paths between remote and local components are typically based on common Internet-based and web-based communication technologies.

In summary, assessment of implementation variants for mobile signature solutions regarding the usability aspect effectiveness leads to sort of ambiguous results. This also becomes apparent from Figure 4.27. In this figure, the four assessed approaches are ranked according to the two factors that contribute to the usability aspect Effectiveness. For the factor feasibility, the Remote-SSCD Approach and the Server-HSM Approach are ranked best, due to their avoidance of local SSCDs. The Classical Approach and the SIM Approach, which rely on local SSCDs, share the third place. Regarding the factor service availability, the Classical Approach is ranked best, as it does not require Internet access. The SIM Approach is ranked worst, as it partly makes use of remote components and additionally relies on special communication technologies that potentially reduce service availability in roaming scenarios.

	Implementation Variant A (Classical Approach)	Implementation Variant B (Remote-SSCD Approach)	Implementation Variant C (SIM Approach)	Implementation Variant D (Server-HSM Approach)
Effectiveness: Feasibility	3	1	3	1
Effectiveness: Service Availability	1	2	4	2

Figure 4.27: Depending on the considered factor, either Implementation Variant B and Implementation Variant D, or Implementation Variant A achieve the highest level of effectiveness.

4.5.3.2 Efficiency

In the context of mobile signature solutions, the usability factor efficiency measures the complexity of required user interactions during signature-creation processes. For mobile signature solutions, interactions with the user are limited to displaying DTBS and requesting authentication data. Displaying of DTBS is similar for all implementation variants. This does not apply in general to requesting authentication data from the user. Approaches relying on remote SSCDs potentially require additional means to assure that signature-creation data, i.e. cryptographic signing keys, remain under sole control of the Signatory. These approaches require more complex authentication schemes, which reduce the efficiency of the overall signature-creation process. Concretely, this applies to the Remote-SSCD Approach represented

by Implementation Variant B and to the Server-HSM Approach covered by Implementation Variant D. Therefore, these two approaches are disadvantageous in terms of efficiency. This is also reflected by Figure 4.28, which ranks the four approaches according to their efficiency. The Classical Approach and the SIM Approach are ranked best, as they rely on a local SSCD. The other two approaches, which make use of a remote SSCD, share the third place.

	Implementation Variant A (Classical Approach)	Implementation Variant B (Remote-SSCD Approach)	Implementation Variant C (SIM Approach)	Implementation Variant D (Server-HSM Approach)
Efficiency	1	3	1	3

Figure 4.28: The conducted assessment yields Implementation Variant A and Implement Variant C as most efficient approaches.

4.5.3.3 Satisfaction

The usability factor satisfaction is heavily influenced by the provided user interface. Considering the four assessed implementation variants, only the components User Client, DTBS Viewer, and Signatory Authentication Component provide a user interface. The User Client is regarded as out of scope, as it is not directly involved in signature-creation processes. The usability of the user interfaces of the remaining two components mainly depends on their implementation. Hence, no general conclusion can be drawn for any of the four implementation variants regarding the usability of provided user interfaces.

Still, it is worth to mention that the achievable usability of user interfaces also relies on the technologies available to realize these interfaces. Mobile apps, which represent the most common approach to implement user interfaces on mobile end-user devices, provide a high degree of flexibility to realize convenient user interfaces. Limitations regarding the usability of user interfaces might however be imposed by the need to employ other technologies than mobile apps. This can be the case for signature solutions relying on local SSCDs. As access to local SSCDs is typically limited, interaction with these components usually requires application of specific technologies. For instance, SIM-based SSCDs typically require the SIM Application Toolkit to provide a communication channel between the Signatory and the SSCD. Representing a rather old and limited technology, implementation of usable user interfaces with this technology can be difficult.

The usability of provided user interfaces is no requirement specific to signature solutions. In contrast, the factors hardware independence and software independence are especially relevant for the level of satisfaction provided by such solutions. Recent usability tests have revealed that the need to acquire and maintain additional hardware and software reduces the pertained level of usability [Zefferer and Krnjic, 2012b]. The need to acquire and maintain additional hardware or software mainly applies to solutions that rely on local SSCDs. These solutions require special hardware on the local device to implement the SSCD. Furthermore, these solutions require special software that enables access to the SSCD. Hence, the need for additional hardware and software mainly affects signature solutions following the Classical Approach or the SIM Approach.

In addition to the SSCD, also other local components require additional software. Components with direct interfaces to the Signatory need to be realized locally in any case. This applies to the DTBS Viewer, the Signatory Authentication Component, and the User Client. All these components raise the need for additional software irrespective of the underlying implementation variant. Differences between implementation variants can however be caused by the Signature Processing Component. This component can be realized either locally or remotely. Local realizations of the Signature Processing Component impose the need for additional local software and hence potentially reduce satisfaction. This applies to the Classical Approach followed by Implementation Variant A and the Remote-SSCD Approach represented by Implementation Variant B.

In summary, the Server-HSM Approach followed by Implementation Variant D is advantageous regarding the usability aspect satisfaction. In general, the Remote-SSCD Approach and the Server-HSM Approach are advantageous compared to other approaches, as they do not require special technologies that enable access to a local SSCD. These two approaches are also advantageous regarding hardware independence due to reliance on a remote SSCD. Finally, the Server-HSM Approach and the SIM Approach are beneficial in terms of software independence due to their remote implementation of the Signature Processing Component. Hence, the Server-HSM Approach represented by Implementation Variant D outperforms the other three approaches, as it is the only variant that is advantageous according to all three factors that contribute to the usability aspect satisfaction. This also becomes apparent from Figure 4.29, which ranks the four implementation variants for mobile signature solutions according to the factors that contribute to the usability aspect satisfaction.

	Implementation Variant A (Classical Approach)	Implementation Variant B (Remote-SSCD Approach)	Implementation Variant C (SIM Approach)	Implementation Variant D (Server-HSM Approach)
Satisfaction: User Interface	1	1	4	1
Satisfaction: Hardware Independence	3	1	3	1
Satisfaction: Software Independence	3	3	1	1

Figure 4.29: Implementation Variant D achieves the highest level of satisfaction when combining all relevant factors.

4.5.3.4 Learnability

The learnability of a signature-creation solution depends on its complexity. An increasing complexity reduces learnability. The complexity of a signature solution is mainly influenced by required user interactions. Required user interactions are similar for all four implementation variants of mobile signature solutions. First, the Signatory needs to check the DTBS being displayed by the DTBS Viewer. Subsequently, the Signatory needs to provide the authentication data that is required to authorize the signature-creation process. These two basic steps are equal for all four implementation variants.

However, differences can be found in the way, how authentication data need to be provided. For implementation variants relying on a local SSCD, provision of static authentication data is usually sufficient, as a second authentication factor, i.e. possession, is implicitly covered by the local SSCD. For remote SSCDs, the authentication factor possession is not implicitly implemented by the SSCD, as the SSCD is not under direct physical control of the Signatory. Hence, solutions relying on remote SSCDs potentially need to implement alternative means in order to assure sole control of the Signatory over signature-creation data, i.e. cryptographic signing keys. For instance, the mobile signature solution Austrian Mobile Phone Signature[24] implements a two-phase authentication mechanism. In the first authentication phase, the Signatory needs to provide a unique ID and a secret static password. In the second authentication phase, an OTP is sent to the Signatory's mobile phone. This OTP needs to be returned by the Signatory to prove possession of the mobile phone.

The need for alternative means to assure the Signatory's sole control over signature-creation data increases the complexity of the authentication process and reduces the learnability of the entire solution. As this applies to solutions with remote SSCDs only, solutions relying on the Classical Approach or the SIM Approach are obviously beneficial in terms of learnability. Accordingly, these two approaches are ranked best regarding the aspect learnability. This is illustrated in Figure 4.30.

[24] https://www.handy-signatur.at/Default.aspx

	Implementation Variant A (Classical Approach)	Implementation Variant B (Remote-SSCD Approach)	Implementation Variant C (SIM Approach)	Implementation Variant D (Server-HSM Approach)
Learnability	1	3	1	3

Figure 4.30: Implementation Variant A and Implementation Variant C are beneficial in terms of learnability.

4.5.3.5 Memorability

Similar to the aspect learnability, also the memorability of a mobile signature solution depends on the solution's complexity. The more complex the solution, the more difficult it is to remember its intended usage. Therefore, for the aspect memorability basically the same considerations apply as for the aspect learnability. Hence, regarding the usability aspect memorability, Implementation Variant A representing the Classical Approach and Implementation Variant C representing the SIM Approach are advantageous. This is also illustrated in Figure 4.31, which ranks all approaches according to their memorability.

	Implementation Variant A (Classical Approach)	Implementation Variant B (Remote-SSCD Approach)	Implementation Variant C (SIM Approach)	Implementation Variant D (Server-HSM Approach)
Memorability	1	3	1	3

Figure 4.31: Implementation Variant A and Implementation Variant C are beneficial in terms of memorability.

4.5.3.6 Error Robustness

Error robustness rather depends on the specific realization of a mobile signature solution than on its underlying architecture or concept. Hence, it is difficult to assess the error robustness of different implementation variants for mobile signature solutions on a pure conceptual level. Nevertheless, it can be assumed that remotely implemented components are typically less prone to errors compared to locally implemented components. This assumption is valid, as remote components are usually operated in a controlled and especially protected environment such as data-processing centers. Data-processing centers typically implement redundancy mechanisms to achieve an adequate service level. Although this does not guarantee absolute robustness against errors and failures, remotely and centrally operated components are usually more stable than components running on end-user devices.

Under this assumption, the error robustness increases with the number of components being implemented remotely. Thus, the Server-HSM Approach represented by Implementation Variant D is advantageous, as it implements both the Signature Processing Component and the SSCD remotely. Following the Remote-SSCD Approach or the SIM Approach, either the Signature Processing Component or the SSCD is implemented remotely. Hence, these approaches are still advantageous compared to the Classical Approach, which implements both the Signature Processing Component and the SSCD locally. These considerations yield the ranking of implementation variants shown in Figure 4.32.

	Implementation Variant A (Classical Approach)	Implementation Variant B (Remote-SSCD Approach)	Implementation Variant C (SIM Approach)	Implementation Variant D (Server-HSM Approach)
Error Robustness	4	2	2	1

Figure 4.32: Implementation Variant D shows the highest error robustness.

4.5.3.7 Cognitive Load

The cognitive load of a mobile signature solution increases with its complexity. Therefore, similar considerations apply for this usability aspects as for the aspects learnability and memorability. As solutions relying on a remote SSCD potentially require more complex means to assure the Signatory's sole control over signature-creation data, the Remote-SSCD Approach and the Server-HSM Approach impose a higher cognitive load than approaches relying on a local SSCD. Hence, Implementation Variant A representing the Classical Approach and Implement Variant C representing the SIM Approach are ranked best according to the usability aspect cognitive load. This is shown in Figure 4.33.

	Implementation Variant A (Classical Approach)	Implementation Variant B (Remote-SSCD Approach)	Implementation Variant C (SIM Approach)	Implementation Variant D (Server-HSM Approach)
Cognitive Load	1	3	1	3

Figure 4.33: Implementation Variant A and Implementation Variant C are beneficial in terms of cognitive load

4.5.4 Comparison of Implementation Variants

The conducted aspect-specific usability assessment compares the four implementation variants for mobile signature solutions mainly on a conceptual level. Advantages and disadvantages of concrete realizations are not considered in detail. Instead, focus is mainly put on aspects that apply to all possible realizations of a certain implementation variant. This facilitates a fair and direct comparison of the four different approaches.

Based on the obtained aspect-specific assessment results, the most usable approach is determined. For this purpose, all aspect-specific assessment results are combined. Concretely, the rankings, which have been derived for each usability aspect, are summed up. This way, the most usable solution regarding all assessed usability aspects is determined. This is illustrated in Figure 4.34. Summing up all aspect-specific results finally yields the Classical Approach represented by Implementation Variant A and the Server-HSM Approach represented by Implementation Variant D as most usable alternatives.

The significance of this kind of quantitative comparison is however limited for several reasons. First, this comparison only yields valid results, if all usability aspects are assumed to be equally important. This is usually not the case in practice. Second, the usability aspects effectiveness and satisfaction have been split into several subordinate factors. Hence, these two aspects contribute stronger to the overall result than other aspects. This needs to be considered when interpreting the results shown in Figure 4.34.

Although the ranking shown in Figure 4.34 does probably not allow for an absolute quantitative comparison of the four implementation variants, it still enables derivation of several useful findings. For instance, it clearly indicates, which approaches are beneficial regarding which usability aspects. The most interesting finding is probably the fact that the Server-HSM Approach can be regarded as the most bipolar solution. For most usability aspects, this approach is either among the best or among the worst solutions. It is also remarkable that most usability aspects, for which the Server-HSM Approach is disadvantageous according to Figure 4.34, are related to the factor efficiency. Hence, the only drawback of this approach in terms of usability is obviously its reduced efficiency. Its reduced efficiency is mainly caused by the need for additional means to assure the Signatory's sole control over remotely stored signature-creation data. If this drawback can be removed, the Server-HSM Approach will clearly represent the most usable and most user-friendly implementation variant for mobile signature solutions.

	Implementation Variant A (Classical Approach)	Implementation Variant B (Remote-SSCD Approach)	Implementation Variant C (SIM Approach)	Implementation Variant D (Server-HSM Approach)
Effectiveness: Feasibility	3	1	3	1
Effectiveness: Service Availability	1	2	4	2
Efficiency	1	3	1	3
Satisfaction: User Interface	1	1	4	1
Satisfaction: Hardware Independence	3	1	3	1
Satisfaction: Software Independence	3	3	1	1
Learnability	1	3	1	3
Memorability	1	3	1	3
Error Robustness	4	2	2	1
Cognitive Load	1	3	1	3
Sum	19	22	21	19
Overall Ranking	***1***	***4***	***3***	***1***

Figure 4.34: Implementation Variant A is beneficial in terms of usability followed by Implementation Variant D, Implementation Variant C, and Implementation Variant B.

4.6 Chapter Conclusions

In this chapter, possible approaches to create electronic signatures on mobile end-user devices have been identified. In total, four implementation variants for mobile signature solutions have been derived from an abstract model. They all have been assessed according to the factors feasibility, security, and usability. Feasibility is of fundamental importance, as only those approaches are worth to be considered that can be implemented and realized on current mobile end-user devices with currently available technologies. In addition, also security and usability need to be taken into account, as these factors are crucial pillars of successful m-government solutions.

In general, the conducted analyses have not yielded an absolute winner. Hence, no approach clearly outperforms all other approaches in all three aspects. Nevertheless, several useful findings can be derived from the obtained results. The conducted feasibility assessment has shown that the general feasibility of mobile signature solutions for mobile end-user devices increases with a decreasing number of components that need to be implemented locally. By removing as many components as possible from the mobile end-user device, it is relieved from supporting technologies that are necessary to implement these components. For these reasons, the Server-HSM Approach can be identified to be advantageous, as it implements as many components remotely as possible. Similar results have also been obtained for the aspect security. The conducted security assessment has yielded the Server-HSM Approach to be slightly advantageous, as remote components can be assumed to be more secure than local components. In terms of security, also the SIM Approach has turned out to be a possible alternative under certain circumstances.

While the Server-HSM Approach is advantageous in terms of feasibility and security, the conducted usability assessment has yielded this approach to represent the most bipolar implementation variant. While the Server-HSM Approach outperforms other approaches in various usability aspects, it has turned out to be clearly disadvantageous for some other usability aspects. Concretely, the Server-HSM Approach fails to achieve an adequate level of efficiency and is also disadvantageous regarding other usability aspects that are related to this aspect. This is mainly caused by the remote implementation of the SSCD, which raises the need for additional means to assure the Signatory's sole control over remotely stored signature-creation data.

Despite its limited usability, the Server-HSM Approach still represents the most suitable implemen-

tation variant, as it is clearly beneficial in terms of feasibility and security. Addressing the concrete problem tackled by this thesis, development of a mobile signature solution for mobile end-user devices will therefore be based on the general concept and architecture of the Server-HSM Approach represented by Implementation Variant D. Even though the Server-HSM Approach is beneficial in terms of feasibility and security, an adequate level of usability is also crucial for successful mobile solutions in general and for m-government solutions in particular. The need for additional means to assure sole control over signature-creation data has been identified as the main usability-limiting factor of the Server-HSM Approach. Development of a mobile signature solution for mobile end-user devices hence must especially focus on the development of authentication means that satisfy given security requirements while at the same time maintain an adequate level of security. Possible alternatives for the development of appropriate means to assure sole control over signature-creation data that meet both security and usability requirements are discussed in the next chapter.

Chapter 5

Authorization Mechanisms for Mobile Signature Solutions

" After climbing a great hill, one only finds that there are many more hills to climb."

[Nelson Mandela, Former President of South Africa.]

Determination of the Server-HSM Approach as the most suitable implementation variant for mobile signature solutions is an important first step. However, as phrased by Nelson Mandela, a first step is usually not enough. This also applies to the development of signature solutions for mobile end-user devices. In particular, the decision to rely on the Server-HSM Approach raises the demand for an adequate authorization mechanism that restricts the centrally provided electronic-signature functionality to legitimate users. Development of such an authorization mechanism is the main topic of this chapter.

Authorization mechanisms are crucial components of signature-creation solutions. They assure that the Signatory maintains sole control over his or her signature-creation data, i.e. his or her personal cryptographic signing keys. In other words, authorization methods guarantee that nobody else but the legitimate Signatory can create electronic signatures in the name of the Signatory. This is a fundamental requirement of advanced electronic signatures and of qualified electronic signatures according to the EU Signature Directive [The European Parliament and the Council of the European Union, 1999] and the EU eIDAS Regulation [The European Parliament and the Council of the European Union, 2014]. Hence, signature-creation solutions that aim to comply with these legislations need to implement adequate authorization mechanisms.

Implementations of authorization mechanisms heavily depend on the particular signature-creation solution. For instance, in the common case of smart card based solutions, access to signature-creation data stored on the smart card is usually protected by a secret PIN. This PIN is only known to the legitimate Signatory, who needs to enter this PIN prior to each signature-creation process. A similar approach is also followed by mobile SIM-based signature solutions, which create electronic signatures inside the Signatory's SIM. Again, the Signatory needs to enter a secret PIN, in order to authorize a signature-creation process. In contrast, server-based mobile signature solutions often follow a slightly different approach. For instance, the Austrian Mobile Phone Signature[1] implements an authorization mechanism consisting of two consecutive steps. In the first step, the Signatory needs to provide a secret password. In the second step, an OTP is sent to the Signatory's mobile phone via SMS. The Signatory needs to enter the received OTP to complete the authorization process.

These examples show that in principle different authorization mechanisms can be implemented by signature-creation solutions. Nevertheless, most implemented mechanisms share one basic commonality:

[1] http://www.handy-signatur.at/

they all rely on multiple authentication factors. For instance, authorization mechanisms of signature solutions based on smart cards or SIMs rely on the authentication factors knowledge and possession. The Signatory needs to possess the smart card or the SIM and needs to know the secret PIN. Also server-based signature solutions that implement authorization mechanisms relying on OTPs delivered by SMS make use of these two factors. Again, the Signatory needs to know a secret password. As server-based solutions use a remote SSCD, this device however cannot directly cover the authentication factor possession. Thus, a one-time password is sent to the Signatory's mobile phone. By proving reception of the one-time password, the Signatory proves to possess the mobile phone.

Differences in authorization mechanisms of existing signature solutions raise the question on minimum requirements imposed by legal or organizational frameworks. In the European Union, the Signature Directive [The European Parliament and the Council of the European Union, 1999] and the eIDAS Regulation [The European Parliament and the Council of the European Union, 2014] represent the legal basis for electronic signatures. Regarding access to signature-creation data and signature-creation functionality, these legislations state that advanced electronic signatures and hence also qualified electronic signatures must be created using means the Signatory can maintain under his sole control. However, no detailed requirements regarding assurance of this sole control are provided. Similar considerations also apply to the requirements for SSCDs, which are needed for the generation of qualified electronic signatures. For SSCDs, relevant legislations define that the signature-creation-data used for signature generation can be reliably protected by the legitimate signatory against the use of others. Again, no concrete technical requirements regarding the implementation of reliable protection mechanisms are provided.

During the past years, EU member states have implemented the EU Signature Directive by means of national laws. In general, an EU directive does not dictate the means that need to be applied by the particular member state to achieve the results defined by the relevant directive. In the special case of the EU Signature Directive, specific requirements regarding the implementation of signature-creation solutions can hence also be defined by national laws. For instance, in the EU member state Austria, the national Electronic Signature Act [Republik Österreich, 2010] and the national Signature Order [Republik Österreich, 2008] implement the EU Signature Directive on national level and define further technical requirements for the generation of qualified electronic signatures. In particular, the Austrian Signature Order defines the requirement for authorization codes to protect the signature-creation functionality of signature-creation devices. Furthermore, the Austrian Signature Order defines that entered authorization data must neither be stored by the signature solution after completion of the signature-creation process, nor by client components to facilitate subsequent signature-creation processes. Similar to the EU Signature Directive, neither the Austrian Electronic Signature Act, nor the Austrian Signature Order defines specific technologies to implement authorization mechanisms of signature-creation solutions.

Legal frameworks on European and national level define the need for appropriate authorization mechanisms for signature-creation solutions. However, these frameworks do in most cases not specify the concrete realizations of such mechanisms. This gives signature solutions the opportunity to realize these mechanisms in different ways. Existing signature solutions typically make use of authorization mechanisms relying on multiple authentication factors. This way, the required level of security is achieved. Reliance on multiple authentication factors can hence be identified as fundamental requirement for authorization mechanisms of signature-creation solutions. This also applies to mobile signature-creation solutions that are tailored to a use on mobile end-user devices. In particular, this also applies to mobile signature solutions following the Server-HSM Approach.

The Server-HSM Approach has been identified to be advantageous for mobile signature solutions regarding the aspects security, feasibility, and usability. Furthermore, a generic model of solutions following this approach has been developed. Due to its abstract nature, this model considers only the general need for an authorization method, but does not reflect the requirement for multiple authentication factors. Furthermore, it does not specify, which authentication factors need to be chosen, and how they have to be implemented. To overcome these limitations, this chapter refines the generic model of the Server-HSM Approach by further detailing aspects related to its authorization mechanism. For this purpose, suit-

able authentication factors are determined first. Subsequently, existing approaches to implement these authentication factors are assessed in order to identify the most suitable approach. This approach is incorporated into the existing model. This finally yields a complete model of signature solutions relying on the Server-HSM Approach. This model can then be used as basis for concrete signature-creation solutions for mobile end-user devices. Thus, this model represents the abstract solution proposed in this thesis.

5.1 Selection of Authentication Factors

Multi-factor authentication schemes typically involve two or three of the factors possession, knowledge, and inherence. The factor possession is usually covered by something the user has, e.g. a smart card or hardware token. Passwords or PINs are typically used to cover the factor knowledge. Finally, biometry is a frequently followed approach to implement the factor inherence. Irrespective of their concrete implementation, combination of multiple authentication factors increases security, as it raises the complexity of attacks.

The selection of authentication factors depends to a large extent on the concrete use case and deployment scenario. In the special case of mobile signature solutions, especially capabilities of mobile end-user devices need to be considered. Modern devices provide various opportunities to cover the authentication factors knowledge and possession. The factor knowledge can be implemented easily by requesting secret credentials such as PINs or passwords from the user. Implementation of the factor possession is usually more complex. Still, existing solutions show that for instance the SIM can be used to cover this authentication factor. Recently, also the factor inherence has gained relevance on mobile end-user devices. In most cases, biometric approaches are followed for this purpose. For instance, recent versions of mobile operating systems support biometry as an alternative to PIN and password-based access protection. For example, Google Android supports device-unlock mechanisms based on face recognition [Google, 2011]. Alternatively, the Apple iPhone integrates a finger-print sensor, which allows users to unlock the phone simply by touching a button. This feature has been available first on the Apple iPhone 5S, which has been introduced in 2013 [Apple, 2013].

Although biometry is obviously on the rise on mobile consumer devices, this authentication factor is not considered in this chapter for the following reasons. First, especially on mobile consumer devices, biometric approaches provide only a low level of security. For instance, the biometric device-unlock solutions of Google Android and Apple iOS are vulnerable to even simple attacks. This has for instance been shown by Diaz [2013] and Silverman [2011]. These vulnerabilities are mainly caused by missing liveness-detection mechanisms on current mobile end-user devices. Second, biometry suffers from two conceptual drawbacks. While e.g. text-based passwords can be easily kept confidential, this is much more difficult for biometric data. For instance, finger prints can be obtained from arbitrary persons rather easily, as they are leaked any time the person touches a smooth surface. Furthermore, in contrast to e.g. passwords, biometric data cannot be revoked in case these data have been compromised. A detailed analysis of biometric authentication approaches has also been provided by Bhattacharyya et al. [2009]. This analysis concludes that biometric authentication is still far from being the perfect solution. For all these reasons, biometry is not considered as potential implementation of the authentication factor inherence for mobile signature solutions. However, consideration of this authentication factor can be regarded as future work, as soon as biometric solutions reach a sufficient level of maturity and reliability on mobile end-user devices. Ruling out the factor inherence, focus is put on authentication schemes that rely on the authentication factors knowledge and possession.

Based on the decision to focus on authorization mechanisms relying on the authentication factors possession and knowledge, the generic model of signature solutions following the Server-HSM Approach can be further detailed. So far, this model considers required authorization mechanisms on a rather abstract level only. All related functionality is combined in a single local subcomponent called Signatory

Authentication Component. The Signatory Authentication Component implements an interface between the Signatory and the remote SSCD. Through this interface, the Signatory provides authentication data in order to authorize signature-creation operations. Due to its abstract nature, the model does not apply any restrictions regarding the concrete implementation of the Signatory Authentication Component. By taking into account the decision to rely on the authentication factors knowledge and possession, the abstract Signatory Authentication Component can be further detailed. This leads to the refined model shown in Figure 5.1.

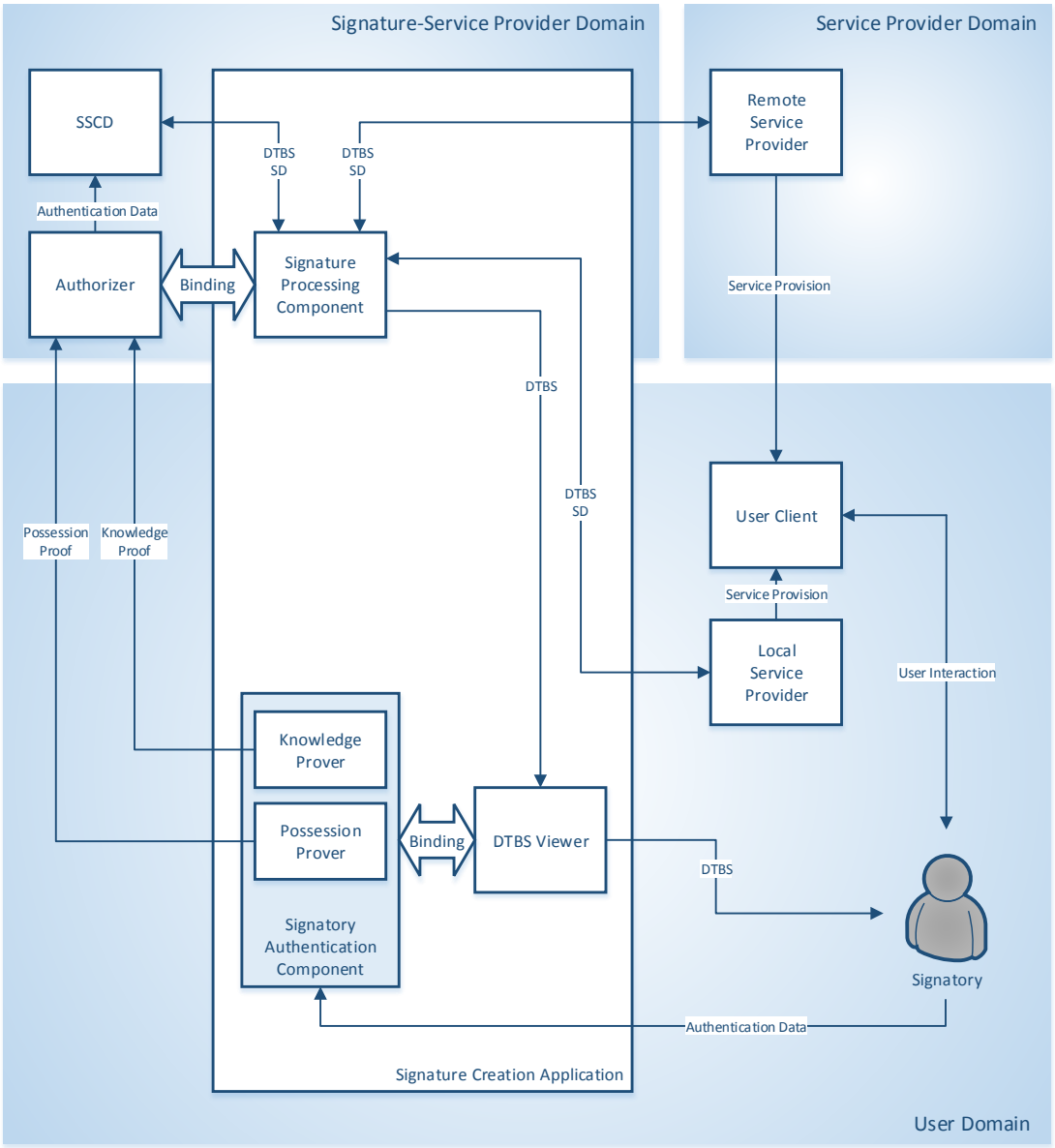

Figure 5.1: The design decision to rely on the authentication factors knowledge and possession yields a refined model of mobile signature solutions following the Server-HSM Approach.

The refined model further details the Signatory Authentication Component defined by the original model. Concretely, the Signatory Authentication Component is split into the subcomponents Knowledge Prover and Possession Prover. These subcomponents request required authentication data from the Signatory to cover the two authentication factors knowledge and possession. The functionality of the

subcomponent Knowledge Prover is rather simple. This subcomponent basically requests some kind of secret data from the Signatory. In the simplest scenario, this is a static PIN or password. By entering this secret data, the Signatory proves knowledge of this data. The Knowledge Prover forwards the requested secret data, i.e. the knowledge proof, to the remote entity, at which the Signatory wishes to authenticate.

While the functionality of the Knowledge Prover is rather simple, the subcomponent Possession Prover needs to be considered in more detail. This subcomponent provides the remote entity, at which the user attempts to authenticate, with a possession proof. This is necessary, as the SSCD is implemented remotely and hence does not implicitly cover the authentication factor possession. Hence, the Possession Prover needs to provide the remote entity with an alternative possession proof. Existing approaches to accomplish this task include for instance one-time passwords delivered by SMS. By proving reception of the one-time password, the user proves possession of a certain mobile phone, or of a certain SIM, respectively. This approach is for instance followed by various e-banking solutions. Examples are e-banking solutions provided by the Austrian banks Raiffeisen[2] or BAWAG PSK[3]. Furthermore, this approach is also followed by the productive mobile signature solution Austrian Mobile Phone Signature[4].

Due to the refinement of the Signatory Authentication Component, also the communication path between this component and the SSCD needs to be further detailed. In the original model, this communication path has been abstractly defined to transport authentication data from the Signatory Authentication Component to the SSCD. According to the design decision to rely on the authentication factors knowledge and possession, this communication path can be further concretized to transport knowledge proofs and possession proofs. These two proofs do not necessarily need to be transported over the same communication path. Hence, the communication path of the original model is also split in the refined model shown in Figure 5.1. This refinement raises the need for a new component named Authorizer, which is located in the same remote domain as the SSCD. The Authorizer receives knowledge and possession proofs from the local User Domain and combines them to authentication data. The combined authentication data is then forwarded to the SSCD. The split of the communication path between the Signatory Authentication Component and the SSCD makes the Authorizer component necessary, in order to be able to leave all SSCD interfaces unmodified. Of course, suitable means need to be implemented to assure an appropriate binding between the newly introduced Authorizer component and the Signature Processing Component. This is also illustrated in Figure 5.1.

The refined model shown in Figure 5.1 considers the design decision to rely on the authentication factors knowledge and possession. In this context, the refined model is hence more specific than the original model defined in Chapter 4. Still, also the refined model does not specify the concrete implementation and realization of authorization mechanisms. In particular, the refined model defines the need for a local Possession Prover and a local Knowledge Prover, but does not specify the implementation of these components. In general, various approaches exist that can be followed to realize these components. In order to systematically determine the best of these approaches, relevant requirements are derived in the following section.

5.2 Derivation of Requirements

Implementation of an authorization mechanism for signature solutions following the Server-HSM Approach requires realization of the two local components Knowledge Prover and Possession Prover. Realization of the Knowledge Prover is rather simple, as the functionality of this component is basically limited to requesting a secret from the Signatory and forwarding this secret to the remote Authorizer. In contrast, realization of the Possession Prover is more difficult. Due to the remote nature of the SSCD, new ground to prove possession of some kind of hardware token needs to be broken.

[2] www.raiffeisen.at
[3] https://www.bawagpsk.com
[4] http://www.handy-signatur.at/

During the past years, different approaches to realize remote possession proofs have been proposed and implemented. These approaches will be surveyed and assessed to determine the best alternative for mobile signature solutions relying on the Server-HSM Approach. The conducted assessments will be based on a set of requirements. These requirements are derived in this section from the refined model shown in Figure 5.1. This assures that all aspects that are relevant for possession-proof implementations for mobile signature solutions following the Server-HSM Approach are covered.

For the sake of clarity, relevant components of the refined model have been extracted and assembled to a reduced model. This reduced model is shown in Figure 5.2. In general, only the User Domain and the Signature-Service Provider Domain are relevant when focusing on the realization of remote possession proofs. The Service Provider Domain does not need to be considered. In addition, also the set of considered components can be reduced in the relevant domains. In the local User Domain, only the Possession Prover and the DTBS Viewer need to be considered. The Possession Prover provides possession proofs to the remote domain. The DTBS Viewer displays DTBS to the Signatory. A binding must be established between these two components, in order to enable the Signatory to check that provided possession proofs are used to sign the displayed data. In the remote Signature-Service Provider Domain, the SSCD, the Signature Processing Component, and the Authorizer need to be considered. The Signature Processing Component supplies the DTBS Viewer and the SSCD with the DTBS. The Authorizer receives possession proofs from the local Possession Prover and provides the SSCD with required authentication data. A binding must also be established between the Authorizer and the Signature Processing Component, in order to assure a correct mapping between DTBS and provided possession proofs.

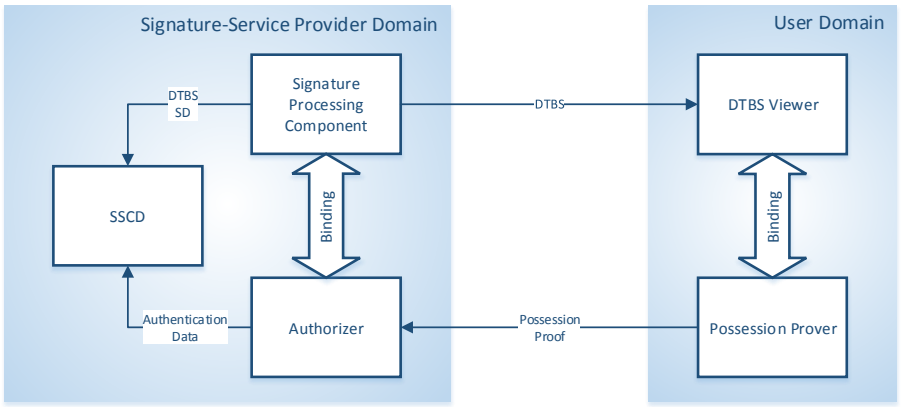

Figure 5.2: Considering only components that are relevant for the provision of possession proofs yields a reduced and simplified model, which can be used to derive requirements.

The reduced model shown in Figure 5.2 contains all components and communication paths that are relevant for the implementation of remote possession proofs. Hence, this model can be used to derive requirements for the systematic assessment of existing approaches. Concretely, the following requirements can be derived:

- **Requirement R1:** Mobile nature of possession proofs
- **Requirement R2:** Remote binding between possession proofs and DTBS
- **Requirement R3:** Local binding between possession proofs and DTBS
- **Requirement R4:** Secure transmission of possession proofs
- **Requirement R5:** Secure transmission of DTBS

Requirement R1 can be derived directly from the facts that the Possession Prover is located in the User Domain and that this component represents the source of possession proofs. As all components of the User Domain are realized on and implemented by mobile end-user devices, also possession proofs must be created on these devices. This leads to Requirement R1, which defines a mobile nature of possession proofs.

Requirement R2 is imposed by the required binding between the Signature Processing Component and the Authorizer. This binding assures that the remote signing entity is provided with opportunities to verify that a received possession proof is linked to the current signature-creation process and to the DTBS belonging to this process. Similar consideration also apply to Requirement R3. This requirement defines that in the User Domain the Signatory must be provided with means to verify that a provided possession proof is linked to the DTBS and used to sign nothing but the data displayed by the DTBS Viewer. Similar to Requirement R2, also Requirement R3 can be derived directly from the reduced model, which specifies a binding between the Possession Prover and the DTBS Viewer in the User Domain.

Finally, Requirement R4 and Requirement R5 can be derived from the fact that the reduced model comprises two locally dispersed domains. Both DTBS and possession proofs need to be exchanged between these domains. This raises the need for means to protect these data during inter-domain transmission. This finally leads to Requirement R4, which defines a secure exchange of possession proofs between the local Possession Prover and the remote Authorizer, and to Requirement R5, which defines a secure transmission of DTBS between local and remote entities.

All requirements have been derived from the abstract and implementation-independent reduced model shown in Figure 5.2. Thus, also the obtained requirements themselves are defined on a rather abstract level. Of course, these requirements need to be refined and further detailed once the abstract model is further developed towards a concrete realization. Nevertheless, the derived requirements can be used in their current form to assess different approaches to implement remote possession proofs on conceptual level. Existing approaches to implement remote possession proofs, which will be assessed by means of the derived requirements, are surveyed in the following section.

5.3 Survey of Existing Approaches

Two-factor authentication on mobile end-user devices has been a topic of interest for a long time. In the past, various solutions have been proposed that allow a user to remotely authenticate at a remote entity. Especially during the past few years, the emergence of smartphones and other powerful mobile end-user devices has facilitated the development of new approaches to implement two-factor authentication with the help of mobile end-user devices. While the authentication factor is usually implemented by means of static passwords, different alternatives to cover the authentication factor possession and to implement remote possession proofs have been proposed. Unfortunately, most of these approaches have not been designed and developed for the special use case of mobile signature solutions. Therefore, special requirements of this particular use case are usually not considered. Hence, proposed approaches can often not be directly applied to mobile signature solutions.

Still, underlying concepts of existing solutions potentially provide useful input for the development of a two-factor authentication scheme that is tailored to the special requirements of mobile signature solutions. Therefore, these solutions are surveyed, classified, and assessed against identified requirements of mobile signature solutions. The conducted survey and analysis is based on related work that has been published on this topic. Main focus is thereby put on the comprehensive survey of existing eID authentication methods in Electronic Finance (e-finance) and Electronic Payment (e-payment) services that has recently been published by the European Union Agency for Network and Information Security (ENISA) [ENISA, 2013].

5.3.1 Static Possession Proofs

A personalized mobile app is probably the most obvious and most simple approach to realize two-factor authentication schemes using current mobile end-user devices. A personalized mobile app is unambiguously linked to a certain user and mobile end-user device. The user can use the app to send static possession proofs that are unique for this particular app instance. Furthermore, the app instance is unambiguously linked to a certain mobile end-user device and hence also to a certain user. Thus, sent possession proofs theoretically cover the authentication factor possession.

Although static possession proofs work in theory, application of this approach raises several issues. For instance, the mobile app must be personalized in order to unambiguously bind it to a certain mobile end-user device and user. This, in turn, requires application of cryptographic methods. This leads to a more complex solution, which is regarded as separate approach and discussed in Section 5.3.5.1 in more detail. A personalization of mobile apps with other means than cryptography is also possible in theory. For instance, simple identifiers could be used. However, this approach is prone to cloning and hence must not be regarded as secure. Another issue arises from the static nature of used possession proofs. Even if they are unique for a particular mobile app, reliance on static possession proofs enables the application of replay attacks. Basic concepts behind replay attacks have been discussed by Syverson [1994].

To add a further layer of security, personalized mobile apps can be equipped with an additional local user-authentication step. During this step, the user authenticates at the local app. For instance, users might be required to enter a password to the app in order to authorize the app to send a static possession proof. Instead of text-based passwords, also alternative means for local authentication such as biometry are conceivable. Regardless of the applied means, the app needs to store reference data in order to be able to validate local authentication data provided by the user. This raises the issue of secure storage of reference data on the mobile client device. Furthermore, if an attacker is capable of intercepting the static possession proof, replay attacks are still feasible. This is an inherent property of approaches relying on static possession proofs, which cannot be completely prevented by an additional local user authentication.

In general, static possession proofs have several drawbacks. It is therefore unsurprising that this approach is hardly followed in practice to implement two-factor authentication by means of mobile end-user devices. Still, this approach is listed here for the sake of completeness and to illustrate relevant aspects that need to be considered.

5.3.2 Time-Based OTP Solutions

Their vulnerability against replay attacks has been identified as most severe drawback of static possession proofs. Thus, most current solutions relying on the authentication factor possession follow dynamic approaches instead. In contrast to their static pendants, dynamic possession proofs are unique for each authentication process. This prevents replay attacks, as intercepted possession proofs cannot be reused.

A common approach to implement dynamic possession proofs are time-based OTPs. Following this approach, the user needs to send a time-dependent password to the remote entity that requires authentication. The user obtains the time-dependent password from a special hardware token, which is paired with the remote entity. This pairing enables the remote entity to verify provided possession proofs. By proving knowledge of the time-dependent OTP, the user proves possession of the token that is required to generate this OTP. This way, the OTP represents a possession proof and the authentication factor possession is covered.

Concrete solutions following this approach have been available on the market for several years. Examples are the products SecurID[5], SafeWord 2008[6], and DIGIPASS[7]. All these solutions implement

[5] http://austria.emc.com/security/rsa-securid.htm

[6] http://www.safenet-inc.com/data-protection/authentication/safeword-2008/

[7] http://www.vasco.com/products/client_products/single_button_digipass/single_button_digipass.aspx

similar functionality. Users are supplied with small security tokens, which are equipped with a rudimentary display. In an initial personalization process, the token is bound to the particular user and paired with a central server. After initialization, the token generates new OTPs in predefined intervals. During authentication processes, the user is requested to enter the current OTP. As the central server is paired with the token, it is able to verify entered OTPs and to complete authentication processes. By relying on adequate cryptographic methods, it is also guaranteed that OTPs cannot be precomputed without the hardware token.

With the emergence of smartphones, vendors of time-based OTP solutions have started to offer smartphone apps as an alternative to hardware tokens. During the personalization process, the smartphone app is bound to the user and paired with the central server. During authentication processes, the time-dependent OTP is displayed by the smartphone app. Thus, the user's smartphone covers the authentication factor possession. A currently popular example of time-based OTP authentication schemes is Google Authenticator[8]. Google Authenticator is an open-source implementation of an OTP generator for different mobile platforms. OTPs are generated according to standards defined by the Initiative for Open Authentication (OATH)[9]. In general, Google Authenticator supports creation of different types of OTPs including time-based OTPs. When relying on a time-based approach, OTPs are created according to the Time-Based One-Time Password Algorithm (TOTP) specified in Request For Comments (RFC) 6238 [Internet Engineering Task Force, 2011].

5.3.3 Event-Based OTP Solutions

Time-based OTP approaches rely on OTPs that depend on the current time. In contrast, event-based OTP solutions create one-time passwords for a special event, e.g. a particular authentication process. Again, the created OTP is unique for each authentication process. This prevents the applicability of replay attacks. Similar as for time-based OTP solutions, event-based OTPs represent possession proofs and hence cover the authentication factor possession.

A common standard for event-based OTP solutions is the HMAC-Based One-Time Password Algorithm (HOTP) specified in RFC 4226 [Network Working Group, 2005]. This standard has been proposed by OATH. According to HOTP, the computation of OTPs is based on a counter, which is incremented after each usage of an OTP. Thus, this counter represents the variable component that is covered by the current time in time-based OTP solutions. To assure that the recipient is able to verify an event-based OTP, sender and recipient need to be synchronized. For instance, if relying on a counter as proposed by RFC 4226, sender and receiver need to assure to increment this counter synchronously.

Examples for current implementations of the HOTP standard are again Google Authenticator and also the Barada Project[10]. The latter provides a smartphone app for the Google Android platform[11]. This app implements HOTP functionality and hence enables authentication processes relying on event-based one-time passwords.

5.3.4 SMS-TAN Approaches

One-time passwords sent to the user via SMS during authentication processes represent a popular alternative to time-based or event-based OTP approaches. This alternative has become commonly known under the term SMS TAN and has been frequently used e.g. by several European e-banking solutions. In contrast to time-based and event-based solutions, SMS-TAN approaches require the remote party, at which the user wishes to authenticate, to create the OTP. This OTP is also denoted as TAN and sent to the user via SMS technology. The user needs to return the received TAN to the remote party usually through

[8] http://code.google.com/p/google-authenticator/
[9] http://www.openauthentication.org/
[10] http://barada.sourceforge.net/
[11] https://play.google.com/store/apps/details?id=net.sf.crypt.gort

an alternative communication channel. By returning the OTP, the user proves reception of the SMS. As a specific SIM is required to receive the SMS, proving reception of the SMS also proves possession of the SIM. This way, the authentication factor possession is covered by the SIM, while the returned OTPs represent possession proofs.

SMS TAN based approaches can also be realized in a slightly modified way. Instead of sending the OTP to the user via SMS, the OTP can also be displayed by the application that requires the user to authenticate. In case of a web-based e-banking solution, the OTP could for instance be displayed in the user's web browser. In this case, the user needs to send the OTP via SMS to a predefined number to complete the authentication process. Also in this alternative, the user's SIM covers the authentication factor possession, as this component is required to send the OTP. Mainly due to its reduced usability, this modified approach is however hardly followed in practice.

In contrast, SMS-TAN approaches that rely on OTPs sent to user's mobile phones are currently employed in different fields of application. A popular example is the e-banking sector, where this kind of user authentication is commonly used to remotely authorize financial transactions. Interestingly, the SMS-TAN approach is also followed by the server-based signature solution Austrian Mobile Phone Signature[12] to authorize signature-creation processes. This shows that the SMS-TAN approach is in principle suitable for server-based signature solutions. The applicability of this approach on mobile end-user devices is discussed in Section 5.4 in more detail.

5.3.5 Challenge-Response Approaches

From a conceptual perspective, the SMS-TAN approach shows an interesting characteristic. In contrast to solutions relying e.g. on time-based or event-based one-time passwords, this approach requires two communication steps to cover the authentication factor possession. First, an OTP, i.e. TAN, is generated by the remote entity and sent to the user via SMS. Second, the user returns the obtained TAN to the remote entity.

Challenge-response approaches follow a similar approach and also rely on two consecutive communication steps. Again, the first communication step is initiated by the remote entity, which creates a so-called challenge. This challenge is transmitted to the user. In contrast to the SMS TAN approach, the user does not return the challenge unmodified to the remote entity. Instead, the user applies some kind of cryptographic method to the challenge. This way, the user creates a so-called response from the challenge. Usually, this includes the processing of a personalized secret cryptographic key. The created response is then returned to the remote entity. As the used key is personalized, only the legitimate user is able to create correct responses from received challenges. Thus, the cryptographic key required to compute responses from challenges covers the authentication factor possession. Accordingly, the computed response represents the possession proof.

Depending on their concrete implementation, existing challenge-response approaches can be classified into two categories. Existing solutions belonging to these two categories are discussed in the following two subsections in more detail.

5.3.5.1 Cryptography-Enabled Local Software

The first category of challenge-response approaches comprises solutions that solely rely on local software. This software is deployed on the mobile end-user device and implements required cryptographic functionality to compute responses from received challenges.

Popular examples assigned to this category are solutions using Quick Response (QR) technology. Google has experimented with QR technology in a research project called Google Sesame [Bowling,

[12] http://www.handy-signatur.at/

2012]. Another solution relying on QR technology is called Secure Quick Reliable Login (SQRL)[13]. The main aim of SQRL is secure login to websites. For this purpose, the user needs to install a software component called SQRL Client on the mobile device. Usually, this software is implemented by means of a smartphone app. The SQRL Client stores a secret master key, which is encrypted with a secret password defined by the user. This key is used to derive website-specific asymmetric key pairs. When the user attempts to login to a website, a QR code is displayed. This QR code contains a transaction token. The user scans the QR code with her mobile device. The SQRL Client extracts the transaction token from the QR code and signs it with the user's private key. The public key is sent to the website together with the signed transaction token. This enables the website to authenticate the user by verifying the obtained signed transaction token.

QR-based solutions are only one possible instance of challenge-response approaches relying on cryptography-enabled local software. Still, they perfectly illustrate key characteristics of this category of challenge-response approaches. The probably most important aspect concerns the computation of responses from challenges, which requires application of cryptographic methods and secure storage of cryptographic keys. Solutions of this category implement these functionalities solely by means of software running on the mobile end-user device. Hence, security and reliability of this software are crucial factors.

5.3.5.2 Hardware-Based Approaches

Hardware-based approaches represent the second category of challenge-response approaches. Similar to solutions relying on cryptography-enabled software, obtained challenges are again cryptographically processed before being returned to the remote entity. In contrast to solutions of the first category, hardware-based approaches rely on a secure local hardware token to store required cryptographic keys and to carry out cryptographic operations. Thus, responses are created from challenges inside a secure environment.

An example for existing solutions belonging to this category is Universal Second Factor (U2F) proposed and developed by the Fast Identity Online (FIDO) Alliance[14]. From an abstract point of view, U2F proposes the use of a hardware dongle at the end-user device. This dongle enables the cryptographic processing of obtained OTPs. However, the U2F specification are not limited to a certain technology or implementation. Thus, different form factors of this hardware dongle are feasible.

In contrast to pure software-based solutions from the first category, hardware-based approaches provide a higher level of security. This is due to the fact that secure hardware is typically more difficult to compromise than software. On the other hand, hardware-based approaches are harder to implement due to the need for secure local hardware.

5.4 Assessment of Existing Approaches

The conducted survey reveals that various approaches to realize two-factor based authentication schemes on mobile end-user devices have been developed so far. All surveyed solutions have in common that they allow users to authenticate at remote entities using mobile end-user devices. Implemented means to provide this functionality range from time-tested SMS-based approaches to sophisticated solutions that make use of innovative mobile technologies. The conducted survey has also revealed advantages and disadvantages of existing solutions. However, due to their conceptual differences, direct comparisons of surveyed approaches and solutions is difficult in most cases.

In addition, mobile signature solutions following the Server-HSM Approach define various special requirements. These requirements have been derived in Section 5.2 in detail. In particular, mobile

[13] http://sqrl.pl/blog/
[14] http://fidoalliance.org/

signature solutions need to assure a mobile nature of possession proofs, a remote and local binding between possession proofs and DTBS, and a secure transmission of possession proofs and DTBS. As these requirements are specific for mobile signature solutions, they are not necessarily considered by the surveyed authentication schemes.

In order to determine the most suitable authentication scheme for mobile signature solutions, all surveyed approaches and solutions are systematically assessed in this section. This assessment is based on the requirements of mobile signature solutions that have been defined in Section 5.2. This way, relevant aspects that are specific for mobile signature solutions are considered and the best available approach is identified.

5.4.1 Static Possession Proofs

Authentication approaches relying on static possession proofs typically make use of personalized mobile apps. These apps send static possession proofs to remote entities whenever the user attempts to authenticate. The main drawback of this approach is its static nature, which enables the application of replay attacks. If a static possession proof is intercepted, it can be used for further authentication processes.

Drawbacks of static possession proofs also become apparent when assessing this approach according to the derived requirements of mobile signature solutions. Figure 5.3 summarizes the results of this assessment and shows that static possession proofs only meet three of the five relevant requirements. The two problematic requirements R2 and R3 are both related to the binding between possession proofs and DTBS. Due to the static nature of employed possession proofs, neither the remote entity nor the Signatory has the opportunity to verify if a provided proof is linked to a particular session and to the DTBS related to this session.

Requirement	*R1: Mobile nature of possession proofs*	*R2: Remote binding between possession proofs and DTBS*	*R3: Local binding between possession proofs and DTBS*	*R4: Secure transmission of possession proofs*	*R5: Secure transmission of DTBS*
Static Possession Proofs	OK	NOK	NOK	OK	OK

Figure 5.3: Authentication solutions relying on static possession proofs are unable to assure a binding between the possession proof, the current session, and the current DTBS.

While static possession proofs are obviously disadvantageous regarding the binding between DTBS and possession proofs, they are able to meet the remaining three requirements of mobile signature solutions. All required components of the local Possession Prover can be implemented on mobile end-user devices, e.g. by means of a mobile app. Thus, the mobile nature of possession proofs can be achieved as demanded by Requirement R1. Furthermore, current mobile end-user devices provide reliable means to realize secure communication channels between local software, i.e. mobile apps, and remote components. This has been elaborated in Chapter 4 in detail. Hence, also Requirement R4 and Requirement R5, which define the need for a secure cross-domain transmission of possession proofs and DTBS, can be met. Nevertheless, Figure 5.3 shows that authentication approaches relying on static possession proofs satisfy only a subset of the identified requirements. Hence, this approach needs to be excluded from further consideration.

5.4.2 Time-Based OTP Solutions

One of the main drawbacks of static possession proofs is their vulnerability against replay attacks. The obvious countermeasure to these attacks is the use of OTPs. This approach is also followed by solutions

that rely on time-based one-time passwords. According to this approach, both the user and the authenticating remote entity create a one-time password during each authentication process. This one-time password is based on the current time and on a shared secret. To successfully complete the authentication process, the user needs to prove to be able to compute the correct OTP and hence to be in possession of the device that is required for this computation. As a new one-time password is used for each transaction, replay attacks are countered.

Still, time-based OTP solutions do not provide an unambiguous binding between the DTBS and the used possession proof, i.e. the one-time password, either. This is due to the fact that OTPs are created independently by the user and the remote entity. Even though the OTP is valid for a certain period of time only, it is not bound to the transaction by any means. In particular, the remote entity has no opportunity to verify if a received OTP belongs to the current transaction. Hence, although time-based OTP solutions prevent replay attacks, they do not assure the required binding between DTBS and provided possession proofs. Hence, these solutions are not able to meet Requirement R2 and Requirement R3, which demand such a binding. This is also illustrated in Figure 5.4, which summarizes the assessment results of authentication solutions relying on time-based one-time passwords.

Requirement	*R1: Mobile nature of possession proofs*	*R2: Remote binding between possession proofs and DTBS*	*R3: Local binding between possession proofs and DTBS*	*R4: Secure transmission of possession proofs*	*R5: Secure transmission of DTBS*
Time-Based OTP Solutions	OK	NOK	NOK	OK	OK

Figure 5.4: Authentication solutions relying on time-based OTPs are unable to assure a binding between the possession proof, the current session, and the current DTBS.

Figure 5.4 also shows that all other requirements but Requirement R2 and Requirement R3 are met by time-based OTP solutions. Existing implementations such as the Google Authenticator[15] show that this approach is feasible on current mobile end-user devices and that Requirement R1 is hence satisfied. In addition, secure communication channels between local software and remote components are also feasible as discussed in Chapter 4. Hence, also secure transmission of DTBS and possession proofs as demanded by Requirement R4 and Requirement R5 is feasible for solutions based on time-based OTPs.

5.4.3 Event-Based OTP Solutions

For event-based OTP solutions, similar considerations apply as for approaches relying on time-based OTPs. The general idea behind event-based OTP solutions has been described in Section 5.3.3 in more detail. Also with this approach, OTPs are generated independently by the user and by the remote entity. Instead of the current time, a counter or similar data being synchronized between the user and the remote entity is used to generate an OTP. Still the basic concept is the same. Hence, also event-based OTPs prevent replay attacks but do not provide any means to unambiguously bind the OTP, i.e. the possession proof, to a certain transaction. Thus, solutions relying on event-based OTPs satisfy the same set of requirements as solutions relying on time-based OTPs. This is illustrated in Figure 5.5 in detail.

Event-based OTP solutions are unable to provide the required binding between possession proofs and DTBS. Hence, these solutions do not satisfy Requirement R2 and Requirement R3, which define the need for such bindings. All other requirements are satisfied. Existing implementations prove the feasibility of this approach on mobile end-user devices as demanded by Requirement R1. Furthermore, means to establish secure communication channels between local software and remote components are

[15] http://code.google.com/p/google-authenticator/

Requirement	*R1: Mobile nature of possession proofs*	*R2: Remote binding between possession proofs and DTBS*	*R3: Local binding between possession proofs and DTBS*	*R4: Secure transmission of possession proofs*	*R5: Secure transmission of DTBS*
Event-Based OTP Solutions	OK	NOK	NOK	OK	OK

Figure 5.5: Authentication solutions relying on event-based OTPs are unable to assure a binding between the possession proof, the current session, and the current DTBS.

available. Thus, possession proofs and DTBS can be exchanged in a secure manner as demanded by Requirement R4 and Requirement R5.

5.4.4 SMS-TAN Approaches

SMS-TAN approaches have a long tradition in security-critical fields of application. For all approaches discussed so far, the possession proof is created independently on the user's local device and is delivered to the remote entity. Hence, coverage of the authentication factor possession requires one communication step only. In contrast, SMS-TAN approaches rely on two consecutive communication steps. First, a TAN is created by the remote entity and delivered to the user via SMS. Second, the received TAN is returned to the remote entity, typically over a separate communication channel.

A new TAN is created by the remote entity for each transaction. Therefore, the TAN is unique for each signature-creation process and hence also for each DTBS. The remote entity is thus able to unambiguously link a received possession proof to the correct transaction. Therefore, SMS TAN based approaches meet Requirement R2, which demands a binding between possession proofs and DTBS that is verifiable by the remote entity.

In most cases, the DTBS is not displayed to the Signatory via SMS. This would be inconvenient due to the limited capabilities of this technology. Instead, the DTBS is usually displayed in the Signatory's web browser or alternatively in a dedicated app. In this case, the Signatory receives the TAN via a different communication channel than the DTBS and the link between the DTBS and the obtained TAN might not be obvious. Hence, Requirement R3 is not implicitly met, as the Signatory cannot verify the binding between displayed DTBS and the obtained TAN. This binding can be assured by sending an additional unique identifier together with the TAN, which is also displayed together with the DTBS. This way, the Signatory can establish a link between the displayed DTBS and the received TAN. It can hence be concluded that SMS-TAN approaches are able to satisfy Requirement R3, if appropriate means, e.g. additional identifiers linking TANs and DTBS, are implemented.

Similar considerations also apply to the alternative SMS TAN based authentication method that has been discussed in Section 5.3.4. Following this alternative approach, the remote entity again generates a TAN that is linked to the current transaction. In contrast to the regular SMS-based approach discussed above, this TAN is displayed to the user by other means than SMS, e.g. through a web page or an app. The user is required to send this TAN back to the remote entity via SMS. In this case, Requirement R2 is also met, as the TAN is again generated by the remote entity, which is thus able to unambiguously link it to the current transaction. In contrast to the regular SMS-TAN approach, this alternative approach also fulfills Requirement R3 implicitly. As the TAN can be displayed together with the DTBS, the Signatory can easily verify the binding between the DTBS and the received TAN. This renders the use of additional identifiers to establish a binding between the displayed DTBS and the received TAN unnecessary.

In addition to Requirement R2 and Requirement R3, both variants of SMS-TAN approaches also fulfill Requirement R1, which demands a mobile nature of employed possession proofs. Support for SMS technology is available on virtually all current mobile end-user devices. Hence, the two discussed

alternatives can be easily implemented on these devices. Furthermore, also Requirement R5 is met by SMS TAN based approaches, as secure communication channels between local software and remote components are feasible on current mobile end-user devices.

The main drawback of SMS-TAN approaches is their reliance on SMS technology. This technology is known to be vulnerable to attacks especially on modern mobile end-user devices. This has been discussed in more detail in Chapter 4. Due to its vulnerabilities, SMS technology must not be assumed to be suitable for a secure transmission of security-critical data. Therefore, SMS-TAN approaches are not able to meet Requirement R4, which demands a secure exchange of data related to possession proofs. Concretely, SMS is no suitable technology to transmit OTPs, whose confidentiality is crucial for the entire authentication process.

Obtained results of the conducted assessment are summarized in Figure 5.6. From this figure, it becomes apparent that the main problem of SMS-TAN approaches is the potential vulnerability of the SMS-based communication channel between the remote Possession Verifier and the local Possession Prover.

Requirement	*R1: Mobile nature of possession proofs*	*R2: Remote binding between possession proofs and DTBS*	*R3: Local binding between possession proofs and DTBS*	*R4: Secure transmission of possession proofs*	*R5: Secure transmission of DTBS*
SMS-TAN Approaches	OK	OK	OK	NOK	OK

Figure 5.6: SMS-TAN approaches are unable to provide a secure communication channel for the exchange of data related to possession proofs.

5.4.5 Challenge-Response Approaches

Challenge-response approaches represent an interesting opportunity to overcome problems of other authentication approaches. For many years, challenge-response approaches have hardly been feasible on mobile end-user device due to their reduced functionality. This has changed with the emergence of smartphones and related powerful mobile end-user devices. The conducted survey on existing solutions has shown that various smartphone-based challenge-response solutions already exist. Depending on their underlying concept and implementation, these solutions can be classified into software-based and hardware-based approaches. The suitability of these two approaches to be applied in mobile signature solutions following the Server-HSM Approach is assessed in the following subsections in more detail.

5.4.5.1 Cryptography-Enabled Local Software

Similar to SMS-TAN approaches, authentication schemes based on cryptography-enabled local software rely on two consecutive communication steps. However, these solutions do not make use of the local SIM to implement the authentication factor possession. Instead, a local software component is personalized with a secret cryptographic key. Hence, this key is bound to a certain instance of the software component. This way, the mobile device, on which this software instance is running, implements the factor possession. Challenges received from the remote entity are cryptographically processed by the local software instance with the help of the Signatory's personal cryptographic key. The result of this processing step represents the response that is returned to the remote entity. The response, in turn, represents a possession proof, as it proves that the Signatory's personal cryptographic key has been used. As this key is bound to a certain mobile device, provision of the response proves possession of this device.

Since authentication solutions based on cryptography-enabled local software follow a challenge-response approach, the required binding between possession proofs and DTBS is guaranteed. As the

challenge is generated by the remote entity, this entity can easily establish a binding between this challenge and the current transaction. Furthermore, the challenge is also cryptographically linked to the possession proof returned by the Signatory. Hence, the required binding between the current transaction and the possession proof is also provided. This way, solutions based on cryptography-enabled local software satisfy Requirement R2, which demands that the remote entity must be able to verify the binding between DTBS and provided possession proofs.

In addition, solutions based on cryptography-enabled local software also satisfy Requirement R3. The local software component can also be used to display the DTBS together with the received challenges. This allows the Signatory to verify the link between the DTBS, the received challenge, and the returned response, i.e. the possession proof. This way, solutions based on cryptography-enabled local software also satisfy Requirement R3, which demands that the binding between possession proofs and DTBS must be verifiable by the Signatory.

As all functionality of the local software component can be implemented by e.g. a smartphone app, solutions following this approach are feasible on current mobile end-user devices. Thus, Requirement R1 is also met by these solutions. In contrast to e.g. SMS-TAN approaches, no additional communication technologies such as SMS are required to transmit security-critical data. Required communication channels between the remote entity and the local software component can be realized and secured using approved technologies. Hence, also Requirement R4 and Requirement R5, which demand a secure exchange of DTBS and possession proofs, are met by solutions based on cryptography-enabled local software.

In contrast to all other approaches assessed so far, solutions relying on cryptography-enabled local software meet all relevant requirements of mobile signature solutions following the Server-HSM Approach. This is also illustrated in Figure 5.7.

Requirement	*R1: Mobile nature of possession proofs*	*R2: Remote binding between possession proofs and DTBS*	*R3: Local binding between possession proofs and DTBS*	*R4: Secure transmission of possession proofs*	*R5: Secure transmission of DTBS*
Cryptography-Enabled Local Software	OK	OK	OK	OK	OK

Figure 5.7: Solutions relying on cryptography-enabled local software are able to meet all predefined requirements.

5.4.5.2 Hardware-Based Approaches

Hardware-based approaches as defined and discussed in Section 5.3.5.2 are similar to approaches relying on cryptography-enabled local software. They also rely on a challenge-response approach and process received challenges by means of cryptographic methods. As a further level of security, required cryptographic functionality is however not implemented in software but in a dedicated secure hardware module.

As hardware-based approaches rely on the same concepts as solutions based on cryptography-enabled local software, these approaches meet the same requirements. Thus, also hardware-based approaches meet all identified requirements of mobile signature solutions following the Server-HSM Approach. This is also illustrated in Figure 5.8.

The main differences between hardware-based and software-based approaches are their feasibility and their provided level of security. As secret cryptographic keys are stored inside a secure hardware element, hardware-based approaches provide a higher level of security compared to solutions that accomplish this task in software only. On the other hand, reliance on secure local hardware components

Requirement	R1: Mobile nature of possession proofs	R2: Remote binding between possession proofs and DTBS	R3: Local binding between possession proofs and DTBS	R4: Secure transmission of possession proofs	R5: Secure transmission of DTBS
Hardware-Based Approaches	OK	OK	OK	OK	OK

Figure 5.8: Solutions relying on secure local hardware elements are able to meet all predefined requirements.

defines additional requirements for mobile end-user devices. This potentially reduces the feasibility of these solutions.

5.4.6 Assessment Results

From the obtained assessment results, the most suitable approach to cover the authentication factor possession can be derived. To facilitate a direct comparison, all obtained assessment results are summarized in Figure 5.9. For each assessed approach, its capability to meet the five relevant requirements of mobile signature solutions following the Server-HSM Approach is emphasized.

Requirement	R1: Mobile nature of possession proofs	R2: Remote binding between possession proofs and DTBS	R3: Local binding between possession proofs and DTBS	R4: Secure transmission of possession proofs	R5: Secure transmission of DTBS
Static Possession Proofs	OK	NOK	NOK	OK	OK
Time-Based OTP Solutions	OK	NOK	NOK	OK	OK
Event-Based OTP Solutions	OK	NOK	NOK	OK	OK
SMS-TAN Approaches	OK	OK	OK	NOK	OK
Cryptography-Enabled Local Software	OK	OK	OK	OK	OK
Hardware-Based Approaches	OK	OK	OK	OK	OK

Figure 5.9: The conducted assessment reveals capabilities and limitations of surveyed approaches to implement remote possession proofs.

Figure 5.9 shows that all approaches are able to meet Requirement R1, which demands feasibility on mobile end-user devices. This is expectable to a certain extent, as only those approaches have been considered, for which concrete implementations on mobile end-user devices are available. More interesting results can be obtained from analyzing Requirement R2 and Requirement R3, which demand a binding between possession proofs and DTBS. As shown in Figure 5.9, these two requirements clearly rule out static possession proofs. Time-based or event-based one-time passwords are not able to meet Requirement R2 and Requirement R3 either. Hence, also these approaches need to be ruled out. While SMS-TAN approaches are able to satisfy Requirement R2 and Requirement R3, they are unable to meet Requirement R4, which demands a secure exchange of security-critical data related to possession proofs. Hence, also SMS-TAN approaches need to be ruled out as a possible alternative.

From the obtained assessment results it becomes apparent that challenge response based approaches represent the only alternative that meets all predefined requirements. This applies to both identified categories of solutions following this approach. The authentication factor possession can be covered either by solutions relying on cryptography-enabled software, or by hardware-based approaches. This

finding can be used to further refine the model of mobile signature solutions following the Server-HSM Approach. The resulting refined model is introduced and discussed in the next section.

5.5 Model Refinement

At the beginning of this chapter, two models have been defined. The first one is a more detailed version of the original model for mobile signature solutions following the Server-HSM Approach. A higher level of detail has been achieved by considering the design decision to rely on a two-factor authentication scheme and in particular on the authentication factors knowledge and possession. The second model is a reduced version of the first one and contains only components that are relevant for covering the authentication factor possession.

This chapter has also assessed different approaches to cover the authentication factor possession. These assessments have yielded challenge response based approaches to be the most suitable alternative. In this section, this finding is used to further refine the two models that have been defined in this chapter so far. First, the reduced model is further refined by considering the decision to rely on challenge-response approaches. Results of this refinement are finally integrated into the detailed model of mobile signature solutions following the Server-HSM Approach. This way, a complete model of mobile signature solutions that follow the Server-HSM Approach and rely on the authentication factors knowledge and possession is derived.

5.5.1 Refined Reduced Model

Conducted assessments have identified the need to rely on a challenge-response approach in order to cover the authentication factor possession. This finding can be used to further detail the reduced model, which has been derived in Section 5.2 and identifies relevant components to cover the authentication factor possession. A graphical illustration of the reduced model has been provided in Figure 5.2 on page 150. Considering the need to implement a challenge-response approach, this model can be further refined. In particular, the local component Possession Prover and the remote component Authorizer can be modeled in more detail. This yields the refined model shown in Figure 5.10.

In the refined model, the abstract and implementation-independent local component Possession Prover is replaced by the three components Challenge Receiver, Transaction Binding Verifier, and Response Creator. In contrast to the abstract Possession Prover, these components are specific to challenge-response approaches. Accordingly, also the abstract and implementation-independent remote component Authorizer is replaced by the components Challenge Creator, Transaction Binding Verifier, Response Verifier, and Verification Result Combiner, which are specific to challenge-response approaches.

Taking the refined model as a basis, provision of a possession proof during an authentication process requires several consecutive steps. Concretely, the following steps need to be followed to cover the authentication factor possession.

1. The Signature Processing Component sends a copy of the DTBS to the local DTBS Viewer. This takes place over a secure communication channel, which protects the confidentiality and integrity of the DTBS. In the Signature-Service Provider Domain, the DTBS is unambiguously linked to the current transaction.

2. At the same time, the Challenge Generator creates a challenge, which is unique and unambiguously linked to the current transaction. This way, the challenge is also linked to the DTBS and sent to the local Challenge Receiver over a secure communication channel, in order to protect its integrity.

3. In the User Domain, the copy of the DTBS received by the DTBS Viewer and the challenge received by the Challenge Receiver is forwarded to the Transaction Binding Verifier. The Transaction

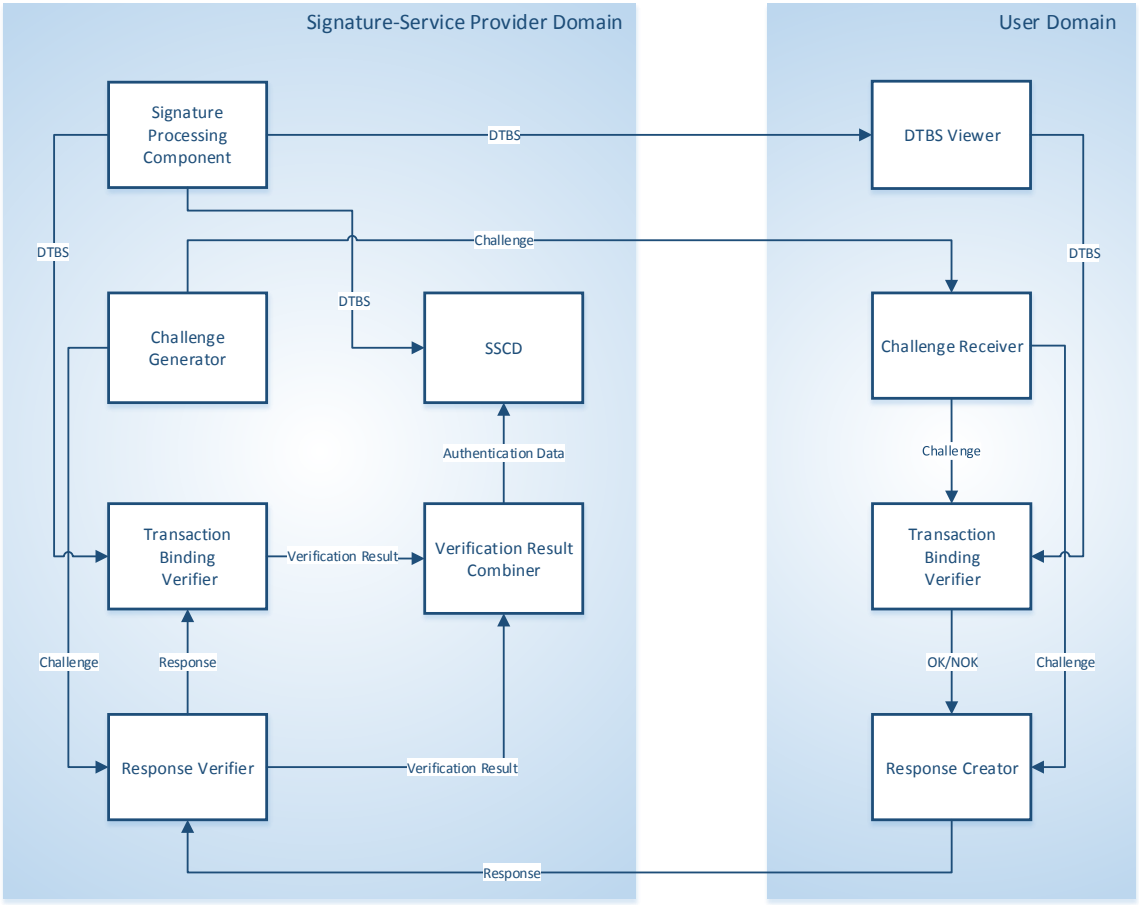

Figure 5.10: Considering the design decision to rely on challenge-response approaches, the reduced model of components relevant for the provision of remote possession proofs can be further refined.

Binding Verifier checks, whether the received challenge belongs indeed to the DTBS displayed by the DTBS Viewer. The result of this check is forwarded to the Response Creator.

4. If the Response Creator receives a positive result from the Transaction Binding Verifier, it computes a response from the received challenge. For this purpose, the Response Creator obtains the challenge from the Challenge Receiver. The locally computed response is returned to the Response Verifier, which resides in the Signature-Service Provider Domain. To protect its integrity, the response is transmitted over a secure communication channel.

5. In the Signature-Service Provider Domain, the Response Verifier checks the validity of the obtained response. For this purpose, the Response Verifier fetches the corresponding challenge from the Challenge Generator.

6. The Response Verifier forwards the obtained response to the Transaction Binding Verifier. The Transaction Binding Verifier takes this response and the sent challenge to verify if the provided response is a valid possession proof for the current transaction and hence for the current DTBS.

7. The verification results determined by the Response Verifier and by the Transaction Binding Verifier are forwarded to the Verification Result Combiner. This component combines the two results. If both results are positive, the possession proof is regarded as valid.

5.5.2 Complete Model

The refined reduced model shown in Figure 5.10 can finally be combined with the detailed version of the model for mobile signature solutions following the Server-HSM Approach, which is shown in Figure 5.1 on page 148. This way, a complete model of a mobile signature solution that follows the Server-HSM Approach and that relies on a challenge response based user authentication is derived. The resulting complete model is shown in Figure 5.11. For the sake of clarity, only the subcomponents of the SCA are shown in Figure 5.11, while the SCA itself is not depicted separately.

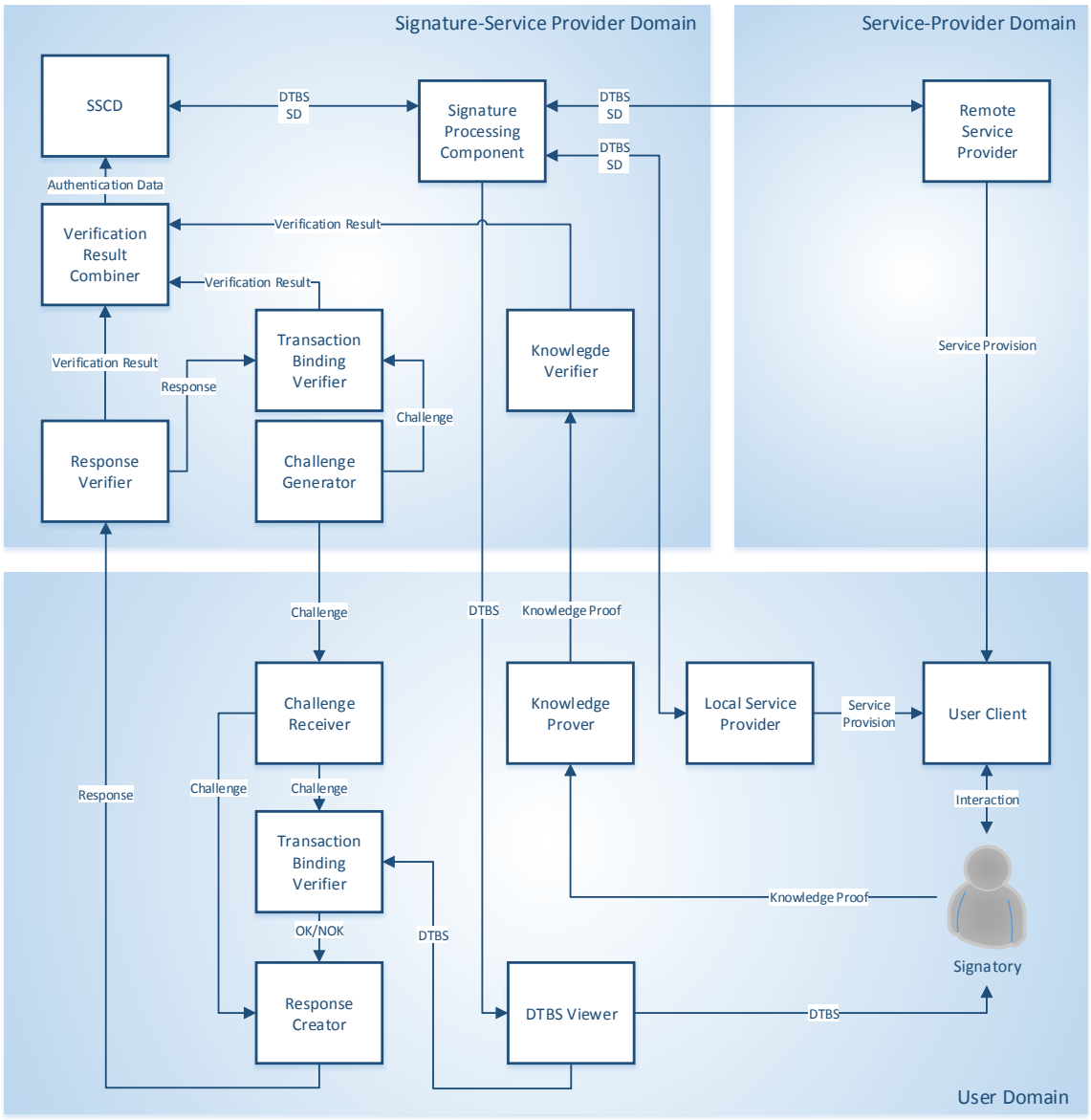

Figure 5.11: By combining the previously developed refined models, a complete model of server-based mobile signature solutions relying on challenge response based authorization methods can be derived.

The complete model identifies all relevant components and required interfaces between them. Following the general architecture of the Server-HSM Approach, also the complete model defines three domains. The User Domain contains the Signatory and all local components that need to be implemented on the Signatory's mobile end-user device. This includes the User Client, optionally a Local Service

Provider, and several components that are required during the authorization process. As defined by the Server-HSM Approach, the SSCD is located in the remote Signature-Service Provider Domain. Besides the SSCD, this domain also contains required components to supply the SSCD with the required DTBS and authentication data. In particular, the Signature-Service Provider Domain contains components to verify received knowledge and possession proofs provided by the Signatory during the authorization process. Finally, the Service-Provider Domain represents the third relevant domain. According to the Server-HSM Approach, this domain contains only the Remote Service Provider.

Based on the derived model shown in Figure 5.11, several steps need to be carried out, in order to complete a signature-creation process. Concretely, the following steps need to be followed.

1. The Signatory interacts with the User Client to access a service provided by the Local Service Provider or by the Remote Service Provider.

2. The involved Service Provider accesses the Signature Processing Component to initiate a signature-creation process. For this purpose, the involved Service Provider supplies the Signature Processing Component with the DTBS.

3. The Signatory is requested to authenticate at the signature service by providing knowledge and possession proofs.

4. The Signatory provides a knowledge proof to the local Knowledge Prover.

5. The Knowledge Prover forwards the obtained knowledge proof to the remote Knowledge Verifier.

6. The Knowledge Verifier verifies the provided knowledge proof and forwards the verification result to the Verification Result Combiner.

7. After successful and positive verification of the provided knowledge proof, the Signature Processing Component sends a copy of the DTBS to the local DTBS Viewer, which displays the DTBS to the Signatory and also forwards it to the local Transaction Binding Verifier.

8. The Challenge Generator creates a challenge. This challenge is sent to the local Challenge Receiver, which forwards it to the local Transaction Binding Verifier and to the Response Creator.

9. The local Transaction Binding Verifier checks, whether the received challenge belongs to the DTBS displayed by the DTBS Viewer. The result of this check is forwarded to the Response Creator.

10. If the challenge can be positively verified, the Response Creator computes a response. The computed response is returned to the remote Response Verifier.

11. The Response Verifier checks the validity of the obtained response. For this purpose, the Response Verifier fetches the corresponding challenge from the Challenge Generator.

12. The Response Verifier forwards the obtained response to the remote Transaction Binding Verifier. The remote Transaction Binding Verifier takes this response and the sent challenge to verify if the provided response is a valid possession proof for the current transaction and hence for the current DTBS.

13. The verification results determined by the Response Verifier and by the remote Transaction Binding Verifier are forwarded to the Verification Result Combiner. This component combines the two verification results with the previously received verification result of the Knowledge Verifier. If all verification results are positive, the signature-creation process is authorized by sending required authentication data to the SSCD.

14. The electronic signature is created inside the SSCD.
15. The SSCD returns the created SD to the Signature Processing Component.
16. The Signature Processing Component forwards the SD to the requesting Service Provider.
17. The involved Service Provider informs the Signatory via the User Client about the successful signature-creation process.

5.6 Chapter Conclusions

The complete model shown in Figure 5.11 represents the main result of Part II of this thesis and the thesis's second basic milestone. Part I of this thesis has revealed the general need for a signature solution for mobile end-user devices. Based on this general objective, possible implementation variants for such solutions have been identified. For each implementation variant, a technology-agnostic model has been defined. The defined models have been systematically assessed and compared in terms of feasibility, security, and usability. Thereby, the Server-HSM Approach has turned out to be the most suitable implementation variant. In this chapter, the general model of this approach has been further refined and developed towards a complete model.

The resulting complete model provides a higher level of detail compared to the general model of the Server-HSM Approach. This mainly applies to components related to authorization mechanisms, which are a crucial aspect of mobile signature solutions. In this chapter, the need to implement authorization mechanisms by means of multi-factor authentication schemes has been identified. The authentication factors knowledge and possession have been chosen for this purpose. Furthermore, challenge-response approaches have been identified to be best suited to cover the authentication factor possession. Taking these findings into account, the general model of the Server-HSM Approach has been further improved by refining components involved in the user-authentication process. This has finally yielded a complete model for signature solutions following the Server-HSM Approach and relying on a challenge response based authorization mechanism. This model finally represents a proposal for an abstract mobile signature solution for mobile end-user devices. As such, it defines the second milestone of this thesis according to the followed methodology.

The proposed model does not define any restrictions regarding specific technologies or other implementation details. It merely represents a common basis for arbitrary implementations of mobile signature solutions that follow the Server-HSM Approach and make use of challenge response based authentication methods. All steps followed to develop the proposed model and to reach the second milestone of this thesis have been based on well-defined methodologies, thorough assessments, and elaborate design decisions. Hence, the model can be assumed to base on the best available general architecture and to rely on the most appropriate authorization mechanism. Due to its abstract nature, it is well-suited to act as basis for concrete signature solutions. To bring this thesis down to a round figure, Part III evaluates the proposed model by further developing it towards a concrete mobile signature solution for mobile end-user devices and by realizing this solution using state-of-the-art technology.

Part III

Evaluation

Chapter 6

Towards a Concrete Mobile Signature Solution: The Smartphone Signature

" A theory must be tempered with reality."

[Jawaharlal Nehru, First Indian Prime Minister.]

Electronic signatures are a crucial building block of transactional e-government services. This especially applies to the EU, where qualified electronic signatures are legally equivalent to handwritten signatures and hence enable users to provide written consent in electronic procedures. During the past years, several signature solutions have been developed and deployed in the EU that allow users to create qualified electronic signatures and that hence enable transactional e-government. To facilitate the transition from transactional e-government to transactional m-government, users must be provided with solutions to create qualified electronic signatures also on mobile end-user devices such as smartphones or tablet computers. This has turned out to be problematic, as currently deployed solutions for the creation of qualified electronic signatures usually cannot be used on mobile end-user devices. This, in turn, renders the realization of transactional m-government services impossible and explains to some extent why most current m-government services are still purely informational.

To address this problem, a model of a mobile signature solution for mobile end-user devices has been proposed in Part II of this thesis. This model has been systematically developed by taking into account relevant legal requirements as well as special properties and characteristics of mobile end-user devices. This way, an implementation-independent and technology-agnostic model of a mobile signature solution for mobile end-user devices has been derived. The chosen abstract nature of the proposed model is actually a double-edged sword. On the one hand, maintaining a technology-agnostic level assures that the model remains valid even when available technologies change. Considering the fact that especially the mobile-computing domain is subject to frequent technological changes, the abstract nature of the proposed model is definitely an asset. On the other hand, the model's technology-agnostic nature complicates its application in practice, as it first needs to be further developed towards a concrete solution. In this regard, keeping the proposed model on a rather abstract level can be an issue.

To overcome this issue, the implementation-independent and technology-agnostic model proposed in Part II of this thesis is further developed towards a concrete mobile signature solution that relies on mobile state-of-the-art technology. In other words and picking up the statement by Jawaharlal Nehru quoted above, the theoretic solution developed in Part II of this thesis is now tempered with reality. The developed concrete solution is called Smartphone Signature and is introduced and discussed in this chapter in more detail. Its name intentionally resembles the name of the Austrian Mobile Phone Signature, as both the Smartphone Signature and the Austrian Mobile Phone Signature share several basic concepts. In addition, its name emphasizes the Smartphone Signature's capability to be used on mobile

end-user devices, which clearly distinguishes the proposed solution from the Austrian Mobile Phone Signature. As the Smartphone Signature represents a concrete implementation of the model derived in Part II of this thesis, it assesses the model's feasibility and applicability on current mobile end-user devices. This way, the Smartphone Signature evaluates the proposed model's suitability to serve as basis for concrete implementations.

Development of the Smartphone Signature has been based on a thorough methodology, which also defines the structure of this chapter. In total, the methodology followed comprises three consecutive steps, which are represented by the three sections of this chapter. In the first step, design principles, which the Smartphone Signature is based on, are defined. Subsequently, the model proposed in Part II of this thesis is used as a basis to derive a functional model of the Smartphone Signature. This derivation considers the previously defined general design principles. Finally, the concrete architecture and process flow of the Smartphone Signature are derived from this functional model. For this purpose, realization opportunities offered and limitations imposed by state-of-the-art technologies currently available on mobile end-user devices are taken into account.

6.1 Design Principles

Derivation of a concrete solution, i.e. the Smartphone Signature, from the model proposed in Part II of this thesis has been based on a set of design principles. These design principles have influenced the choice of technologies employed to implement relevant components and communication paths. Concretely, they have been used to transform the implementation-independent model into a complete functional model, from which the Smartphone Signature has finally been derived.

Design principles defined in this section are closely related to the design principles that have also been used for the systematic development of the proposed underlying model. Adopting these principles again for the derivation of a concrete solution is reasonable. It assures that positive characteristics of the proposed implementation-independent model are maintained and that relevant aspects are also considered when further developing this model towards a concrete solution. The derivation of a functional model of the Smartphone Signature is based on the following design principles:

- **Server-HSM Approach:** A systematic comparison of all possible variants to implement signature-creation solutions for mobile end-user devices has yielded the Server-HSM Approach to be advantageous regarding the factors feasibility, security, and usability. This approach implements as many components remotely and off the mobile end-user device as possible. In particular, this applies to the SSCD, which is required to securely store private cryptographic keys and to carry out required cryptographic computations. The advantage of the Server-HSM Approach compared to other implementation variants of mobile signature solutions has been directly incorporated into the proposed implementation-independent model. Hence, concrete solutions that are based on this model should implicitly follow this approach as well. Nevertheless, due to its importance and for the sake of completeness, reliance on the Server-HSM Approach is again explicitly defined as design principle for the Smartphone Signature.

- **Challenge response based authorization mechanism:** The restriction to solutions following the Server-HSM Approach raises the demand for suitable authorization mechanisms. These mechanisms are necessary to protect remotely stored signature-creation data, i.e. cryptographic signing keys. In this context, the demand for multi-factor authentication in general and for reliance on the authentication factors knowledge and possession in particular raises additional challenges. This is mainly due to the fact that the remote SSCD is not under direct physical control of the user. A thorough comparison of different approaches to overcome this issue has finally yielded challenge response based authentication and authorization mechanisms as the most suitable solution. This finding has also been incorporated into the developed implementation-independent model.

Concrete solutions adopting this model should hence implicitly feature a challenge response based authentication and authorization mechanism. For the sake of completeness, reliance on such mechanisms is also defined as general design principle for the Smartphone Signature.

- **Sufficient level of security:** A sufficient level of security is a crucial requirement for all solutions that store or process security-critical data. This especially applies to solutions that enable end users to create legally binding electronic signatures. The basic demand for a sufficient level of security has also influenced development of the implementation-independent model for mobile signature solutions. In particular, the two decisions to rely on the Server-HSM Approach in general and to rely on a challenge response based authorization mechanism have been derived by taking security considerations into account. Hence, the proposed implementation-independent model of signature solutions for mobile end-user devices can be assumed to facilitate development of concrete solutions that feature an appropriate level of security. Nevertheless, a sufficient level of security is also relevant for aspects of these solutions that are not directly covered by the underlying model. Hence, a sufficient level of security is defined as general design principle for the Smartphone Signature.

- **Support for multiple platforms:** The heterogeneity of current mobile platforms represents a significant challenge for all kinds of mobile applications. Popular mobile platforms and operating systems such as Google Android, Apple iOS, or Microsoft Windows Phone 8 differ significantly in terms of provided features and supported technologies. This complicates the development of mobile solutions applicable on and compatible to all these platforms. Nevertheless, provision of generally applicable solutions is crucial in order to not exclude certain user groups. This applies to m-government solutions in general and to mobile signature solutions in particular. Hence, support for multiple platforms can be identified as important design principle for the Smartphone Signature.

- **Best possible user experience:** Usability and user satisfaction have been identified as crucial success factors of m-government solutions. Hence, these factors are also important to be considered in the context of signature-creation solutions for mobile end-user devices. Unfortunately, there is usually a well-known trade-off between security and usability. This has for instance been discussed by Gutmann and Grigg [2005], Ben-Asher et al. [2009], or Mairiza and Zowghi [2010]. Since a sufficient level of security has been identified as crucial requirement and design principle, achieving an adequate level of usability at the same time might be challenging. Nevertheless, a best possible user experience and an appropriate level of usability need to be defined as relevant design principles for mobile signature solutions as well. Being aware of the trade-off between security and usability, concrete mobile signature solutions must aim to maximize user experience while still meeting relevant security requirements. Naturally, this also applies to the Smartphone Signature.

Considering these design principles, the proposed implementation-independent model developed in Part II of this thesis represents an ideal starting point. This model already complies with the design principles that demand reliance on the Server-HSM Approach and on a challenge response based authorization mechanism. Furthermore, the three design principles that demand a sufficient level of security, support for multiple platforms, and the best possible user experience are related to the aspects security, feasibility, and usability. These aspects have also been considered during development of the implementation-independent model. Hence, most aspects covered by the defined design principles are to a certain extent already taken into account. Nevertheless, it is important to reconsider these aspects when further developing the implementation-independent model towards a concrete solution.

6.2 Functional Model

In this section, a functional model of the Smartphone Signature is derived from the implementation-independent model for mobile signature solutions. The latter merely identifies relevant components and interfaces between them. However, it does not specify the use of certain technologies for their concrete implementation. Choosing suitable technologies and hence an appropriate implementation is left to the respective solution that builds on top of this model.

In the following subsections, components of the implementation-independent model, for which concrete implementations need to be chosen, are identified first. For all these components, possible implementations are then discussed and a concrete implementation is chosen based on the defined design principles. By combining all chosen implementations, a complete functional model of the Smartphone Signature is finally derived.

6.2.1 Identification of Relevant Components

Due to its implementation-independent and technology-agnostic nature, the model developed in Part II of this thesis can act as basis for various different implementations and solutions. The number of possible solutions increases with the number of relevant components and interfaces, for which concrete implementations need to be determined. The identification of these components and interfaces is hence an important first step towards development of a concrete solution. The identification process has been based on the following two assumptions:

- **Focus on relevant components:** The implementation-independent model contains all components that cover a certain task in a signature-creation process. This also includes the Service Provider, which acts as source of the DTBS and as final destination of the SD. Furthermore, the model also contains the User Client, which provides the Signatory access to a service provided by the Service Provider. However, these two components are not integral parts of the signature solution itself, but mainly act as intermediary between the signature solution and the Signatory. Hence, the Local Service Provider, the Remote Service Provider, and the User Client are not considered in detail when choosing concrete implementations for relevant components.

- **Focus on local components:** Following the Server-HSM Approach, the developed implementation-independent model defines both local and remote components. In most cases, these two types of components need to be aligned with each other. For instance, for a specific implementation of the local Knowledge Prover, a corresponding implementation of the remote Knowledge Verifier needs to be chosen. If the Knowledge Prover e.g. relies on alphanumeric passwords as knowledge proofs, the Knowledge Verifier needs to provide means to verify this kind of passwords. Similar considerations apply also to other local and remote components. Due to dependencies between local and remote components, it is reasonable to focus on one domain first when choosing concrete implementations. In most cases, local components are more challenging to realize. This is due to the fact that they are limited by capabilities of mobile end-user devices. In contrast, remote components are assumed to show a higher flexibility regarding their realization. For this reason, concrete implementations for local components are chosen first. Remote components are subsequently aligned accordingly.

Figure 6.1 recalls the proposed implementation-independent model for mobile signature solutions that has been developed in Part II of this thesis. In contrast to the original model, Figure 6.1 highlights those components, for which a concrete implementation needs to be chosen. Relevant components have been determined by taking into account the two assumptions discussed above. Accordingly, focus is first put on components that are integral part of the signature-creation process and that need to be realized locally. As shown in Figure 6.1, this finally yields the Knowledge Prover, the DTBS Viewer, the Challenge

Receiver, the Transaction Binding Verifier, and the Response Creator to be relevant. For them, concrete implementations need to be chosen first.

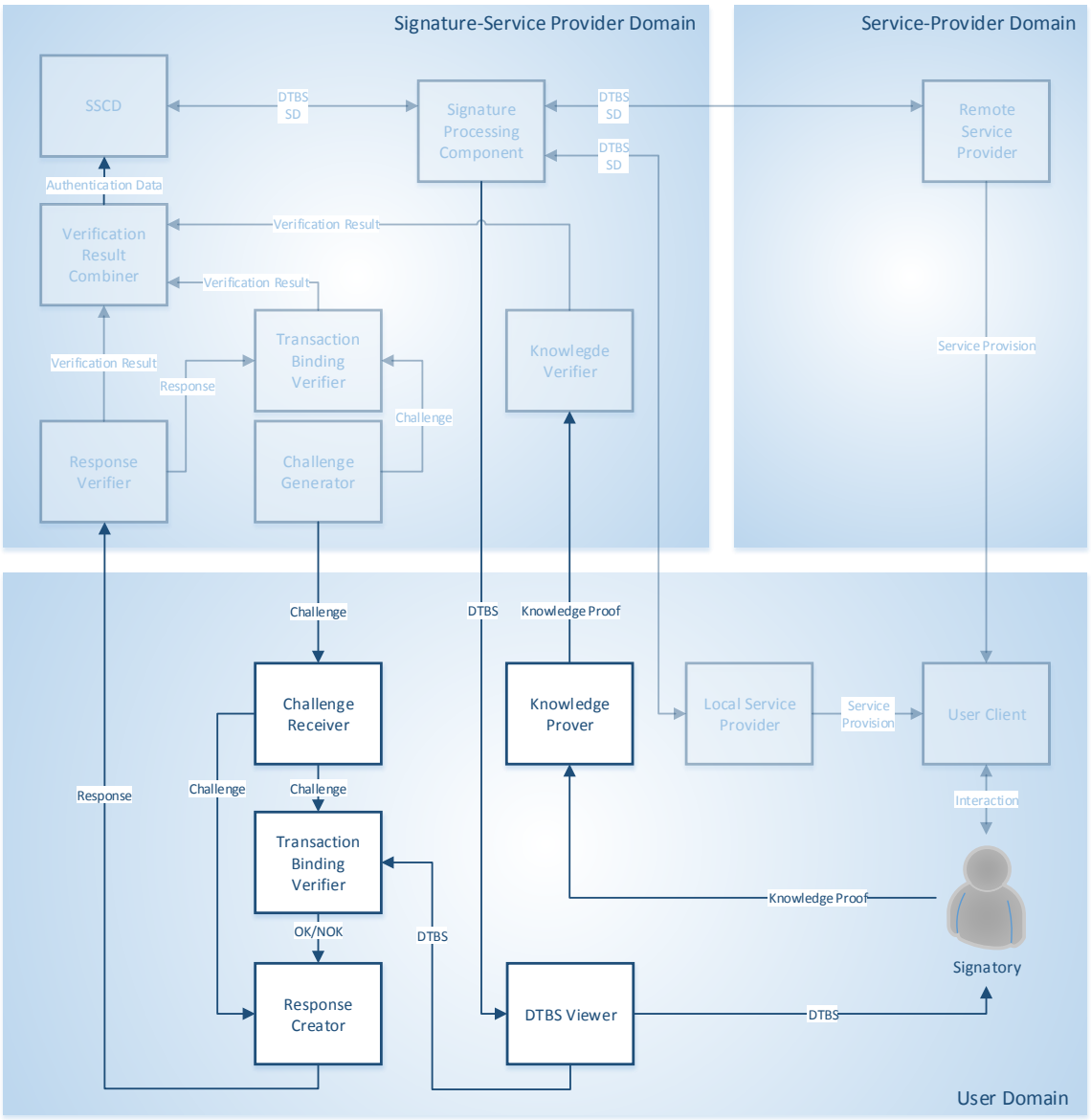

Figure 6.1: Based on the made assumptions, relevant components can be identified from the proposed abstract model.

6.2.2 Implementation of Relevant Components

Concrete implementations of the five relevant components are chosen in accordance with the design principles defined in Section 6.1. This way, the best possible implementation is determined for each relevant local component. In addition, related remote components are aligned accordingly. This is elaborated in the following subsections in detail.

6.2.2.1 Knowledge Prover

The Knowledge Prover requests knowledge proofs from the Signatory and forwards them to the remote Knowledge Verifier. Together, these two components cover the authentication factor knowledge of the multi-factor authorization mechanism. Choosing a concrete implementation for the local Knowledge Prover hence directly influences the implementation of the remote Knowledge Verifier.

As a first step towards a concrete implementation of the Knowledge Prover, the employed type of knowledge proofs needs to be determined. In principle, different approaches exist to realize knowledge proofs. The probably most popular and best known approach is the use of alphanumeric passwords. Even though these passwords suffer from several drawbacks, they are still commonly used.

Being aware of the various drawbacks of alphanumeric passwords, several alternatives have been developed in the past. A recent example are so-called picture passwords, which are for example offered as alternative authentication method for access-protection mechanisms on mobile end-user devices. Such solutions are for instance available for the mobile platforms Google Android[1] or Windows RT[2]. These solutions enable the user to define a pattern on an arbitrary picture, e.g. by drawing a circle around an object in the picture. This pattern then represents the secret picture password.

Reliance on pictures instead of text to cover the authentication factor knowledge is actually not a new idea. This approach has been a topic of scientific interest for many years. An early survey of different approaches to implement graphical passwords has been provided by Suo et al. [2005]. Since then, several new and improved authentication schemes based on graphical passwords have been introduced. Solutions proposed by Almulhem [2011] or Khan et al. [2011] are just two out of many schemes that have been introduced during the past years.

Even though several interesting authentication schemes based on graphical passwords have been proposed, hardly any of them have been broadly used in practice so far. Similar to other authentication schemes, also graphical passwords suffer from the well-known trade-off between security and usability. This has been discussed in detail by Lashkari et al. [2011]. This trade-off can be especially problematic on mobile end-user devices, which often suffer from reduced screen sizes. As a sufficient level of security and a best possible user experience have been defined as design principles, graphical passwords must be regarded as inappropriate for the implementation of the Knowledge Prover.

In default of an alternative that is able to provide high usability while maintaining a sufficient level of security, alphanumeric passwords are chosen to cover knowledge proofs. Even though alphanumeric passwords are also far from being perfect, they at least represent an established and approved method that can be easily implemented on arbitrary mobile end-user devices. By defining rules regarding the minimum complexity of chosen passwords, an adequate compromise between security and usability can be achieved.

Based on the decision to rely on alphanumeric passwords as possession proofs, also the implementation of the local component Knowledge Prover can be determined. As knowledge is proven by means of an alphanumeric password, the Knowledge Prover needs to request this password from the Signatory and forward it to the remote Knowledge Verifier. Hence, the functionality of the Knowledge Prover is basically reduced to two aspects. First, the Knowledge Prover needs to provide a user interface, through which the Signatory can enter alphanumeric passwords. Second, an interface to the remote Knowledge Verifier needs to be provided, through which entered passwords can be forwarded.

Having determined the type of employed knowledge proofs and the concrete implementation of the Knowledge Prover, also the functionality of the remote Knowledge Verifier can be aligned accordingly. The Knowledge Verifier needs to compare the obtained knowledge proof, i.e. the alphanumeric password, with some kind of reference data. This can for instance be the alphanumeric password itself or its hash value. Provision of an interface to the Knowledge Prover, storage of reference data, and comparison of

[1] https://play.google.com/store/apps/details?id=com.TwinBlade.PicturePassword&hl=en
[2] http://windows.microsoft.com/en-us/windows-8/personalize-pc-tutorial

received passwords with stored reference data are hence the main functions that need to be implemented by the remote Knowledge Verifier. All these functions can be easily provided in software.

6.2.2.2 DTBS Viewer

The functionality of the local component DTBS Viewer is rather simple and basically limited to receiving DTBS from the remote Signature Processing Component and displaying these data to the Signatory. Hence, the number of possible implementation variants of this component is limited. In any case, this component needs to be implemented by means of a viewer component that is able to reliably display data to the Signatory. A further concretion of this component is not necessary at this point.

6.2.2.3 Challenge Receiver

Together with other local and remote components, the Challenge Receiver implements the authentication factor possession. For this purpose, a challenge is generated in the Signature-Service Provider Domain and transferred to the User Domain. There, a response is created from this challenge by applying cryptographic methods. This response finally represents the possession proof and is transferred back to the Signature-Service Provider Domain for verification. The role of the local Challenge Receiver is rather simple. It receives the challenge created in the remote Signature-Service Provider Domain and forwards it to other local components involved in the authorization process. Hence, the main functionality of this component is provision of appropriate interfaces to the remote Signature-Service Provider Domain and to several local components.

The concrete implementation of the local Challenge Receiver hence mainly depends on the communication interface that is used to transmit the remotely generated challenge to the local User Domain. Considering currently available communication technologies on mobile end-user devices, two approaches to accomplish this task are feasible. First, the challenge can be transmitted over an established Internet-based connection between the mobile end-user device and the remote Signature-Service Provider Domain. Second, the mobile network can be used to transmit challenges, e.g. by means of SMS technology. Other communication technologies such as Bluetooth or NFC, which are also available on current mobile end-user devices, provide short-range communication capabilities only and are hence not suitable for the cross-domain transfer of challenges.

Having the choice between an Internet-based or a mobile network based long-range communication interface, the latter has been chosen for the transmission of challenges. Concretely, challenges are implemented as random TANs and transferred to the local User Domain via SMS technology. This might be surprising at a first glance, since pure SMS-TAN approaches have been ruled out as potential authentication and authorization approach, due to their disadvantageous security. However, there is a significant difference between pure SMS-TAN approaches and the authorization mechanism defined by the proposed implementation-independent model. Following a pure SMS-TAN approach, the authentication factor possession is covered by the Signatory's SIM and by the fact that the TAN delivered via SMS can only be received with this SIM. Hence, the confidentiality of the transmitted TAN is crucial for the overall security of SMS-TAN approaches. Due to known vulnerabilities of SMS-based communication especially on current mobile end-user devices, the confidentiality of data delivered via SMS is however not granted. In contrast, the concrete solution proposed in this chapter, i.e. the Smartphone Signature, additionally implements a challenge-response approach. Hence, the authentication factor possession is not covered by the SIM and by the capability to receive a certain SMS message, but by the capability to compute a correct response from the received TAN. Thus, confidentiality of data transmitted via SMS, i.e. the challenge, is not crucial for the overall security of the authentication process. Even if an attacker is able to intercept the challenge transmitted via SMS, he or she cannot create a valid possession proof. Thus, reliance on SMS technology to transmit challenges from the remote Signature-Service Provider Domain to the local User Domain is a legitimate alternative.

While reliance on SMS technology to transmit challenges is not disadvantageous in terms of security, this approach is even beneficial in several other aspects. First, adding an additional communication channel increases the overall security. An attacker, who attempts to compromise the entire system, potentially needs to target two different communication channels and technologies. This increases the effort for successful attacks and hence increases security. Second, reliance on SMS technology in authentication schemes is a common approach that is for instance followed by the Austrian Mobile Phone Signature[3] and also by various e-banking solutions. Hence, relying on SMS technology in mobile signature solutions enables users to make use of familiar technologies and processes. This increases usability and in turn also the user acceptance of the proposed solution. The defined design principle that demands a best possible user experience hence justifies reliance on an SMS-based approach.

Taking the design decision made, i.e. reliance on SMS technology to transmit challenges, into account, the implementation of the local Challenge Receiver can be determined. Concretely, the Challenge Receiver needs to implement SMS-receiving functionality. Furthermore, it must be able to extract data wrapped in SMS messages and forward these data to other local components. On platforms that provide mobile apps access to incoming SMS messages, this functionality can be implemented by a mobile app. On other platforms, the SMS app that comes with the mobile operating system needs to be employed.

Based on the decision to rely on SMS technology for the transmission of challenges, i.e. TANs, from the remote Signature-Service Provider Domain to the local User Domain, also the implementation of the remote component Challenge Generator can be determined. Due to the SMS-based approach, this component needs to implement SMS-sending functionality. Furthermore, this component must be able to generate random TANs and to wrap these TANs in SMS messages. Various technologies exist that enable server-based software components to implement this functionality.

6.2.2.4 Transaction Binding Verifier

During authorization processes, the binding between the current transaction, the DTBS, and data related to possession proofs needs to be assured. In the local User Domain, the binding between the displayed DTBS and the received TAN needs to be verified. According to the proposed implementation-independent model, verification of this binding is covered by the local Transaction Binding Verifier.

In total, two approaches can be followed to verify the required binding and to implement the Transaction Binding Verifier. First, the generated challenge, i.e. the TAN, can be derived in the Signature-Service Provider Domain from the DTBS. This can be accomplished e.g. by means of a cryptographic hash function. In this case, the local Transaction Binding Verifier can apply the same hash function to the received DTBS and compare the result with the obtained TAN. Second, the Signatory can cover the functionality of the local Transaction Binding Verifier and manually verify the binding between displayed DTBS and obtained TAN. This approach is for instance followed by the Austrian Mobile Phone Signature and requires an additional reference value to be delivered to the Signatory together with the TAN. The same reference value is also displayed together with the DTBS. This way, the Signatory can manually verify the link between the DTBS and the received TAN by comparing the two reference values.

Both sketched approaches provide a sufficient level of security and reliability. Relying on a manual verification requires more interaction from the Signatory. However, this approach also gives the Signatory more control over the entire signature-creation process. Therefore, this approach has been chosen as implementation of the local Transaction Binding Verifier. This way, the defined design principle that demands a best possible user experience is considered.

Since the Signatory assumes the role of the local Transaction Binding Verifier, no technical implementation of this component needs to be determined. However, other components related to this component need to be adapted accordingly in order to consider the necessity of an additional reference

[3] http://www.handy-signatur.at/

value. This applies for instance to the remote Challenge Generator, which needs to generate and forward an additional reference value for each TAN.

6.2.2.5 Response Creator

The local Response Creator is probably the most important component regarding coverage of the authentication factor possession. According to the proposed implementation-independent model, the Response Creator computes a response from the received challenge. This response finally represents the possession proof, which is transferred to the remote Signature-Service Provider Domain. There, it is verified by the remote Response Verifier. In order to cover the authentication factor possession, it is crucial that the correct response can be created by the Response Creator only. This is usually achieved by means of a Signatory-specific cryptographic key, which is kept confidential. This key is applied to the challenge by means of a cryptographic method. This distinguishes the Smartphone Signature from pure SMS-TAN approaches. Following a pure SMS-TAN approach, the obtained TAN is returned to the remote entity without modification. In contrast, the Smartphone Signature processes the obtained TAN to produce a unique response.

Support of a proper cryptographic method to create responses, i.e. possession proofs, from received challenges is a key requirement of the local Response Creator. Different cryptographic schemes can be employed to implement the required cryptographic method. For instance, possession proofs can be created by applying a symmetric cipher operation to the challenge using a secret symmetric key. In this case, the remote Response Verifier also needs to be aware of this key, in order to be able to verify received possession proofs. This raises the need for secure and reliable key-exchange mechanisms. Alternatively, also asymmetric cryptographic approaches can be followed. For instance, the local Response Creator can sign challenges using a private key and an asymmetric cryptographic algorithm such as RSA [Rivest et al., 1978] or ECDSA [ANSI, 2005]. The signed challenge represents the possession proof. In this case, the remote Response Verifier can verify the obtained possession proof by verifying the signed challenge. When relying on an asymmetric cryptographic approach, no exchange of symmetric keys between the local Response Creator and the remote Response Verifier is necessary. Instead, only the corresponding public key must be published.

Due to their advantages regarding required key-exchange mechanisms, asymmetric cryptographic approaches are chosen for the implementation of the local Response Creator. Concretely, challenges, i.e. TANs, are cryptographically signed using an adequate asymmetric cryptographic algorithm. For this purpose, the Response Creator needs to securely store and apply a cryptographic signing key. Secure storage of cryptographic signing keys and application of asymmetric cryptographic methods are hence the main functions that need to be implemented by the local component Response Creator.

In practice, various alternatives to provide required functionality of the Response Creator on mobile end-user devices exist. However, the concrete implementation of this component is not further specified at this point. Instead, the Response Creator is intentionally defined to be as flexible as possible. The main reason for this design decision is the fact that the secure provision of cryptographic functionality on mobile end-user devices is a complex and challenging task. Mobile platforms and operating systems provide different opportunities to implement such functionality. Likewise, mobile platforms are also prone to various security threats. Hence, determination of a unique implementation that is equally suitable for all current platforms and end-user devices is practically impossible. As support for multiple platforms has been defined as design principle, the Smartphone Signature needs to feature a certain degree of flexibility, in order to be able to adapt to capabilities and limitations of different platforms and end-user devices. For these reasons, determination of implementation details of the component Response Creator is limited to specifying the use of asymmetric cryptographic methods for the creation of responses from obtained challenges. This way, the Smartphone Signature can easily adapt to properties of different mobile platforms. Furthermore, upcoming trends and technologies available on mobile end-user devices can be integrated easily to improve the Smartphone Signature in general and the implementation of the

Response Creator in particular.

6.2.3 Derivation of Functional Model

The implementation-independent model developed and proposed in Part II of this thesis defines basic characteristics such as reliance on the Server-HSM Approach and application of a challenge response based authorization mechanism. Furthermore, it identifies relevant components and communication paths on a purely implementation-independent and technology-agnostic level. In order to develop a concrete solution, i.e. the Smartphone Signature, from this model, concrete implementations of identified components and communication paths have been chosen.

In this section, all chosen implementations are combined. This way, a complete functional model of the Smartphone Signature is derived. This functional model still features the basic characteristics of the underlying implementation-independent model, but additionally specifies the concrete implementation of relevant components. For the derivation of the complete functional model, the following decisions regarding the implementation of relevant components are considered:

- **Knowledge Prover:** Alphanumeric passwords are used to cover the authentication factor knowledge.
- **Challenge Receiver:** TANs delivered via SMS technology from the remote Signature-Service Provider Domain to the local User Domain are used to implement challenges.
- **Transaction Binding Verifier:** The required binding between DTBS and provided possession proofs is manually verified by the Signatory in the User Domain.
- **Response Creator:** Asymmetric cryptographic methods are employed to derive responses from obtained challenges in order to cover the authentication factor possession.

By applying all these decisions to the proposed implementation-independent model, a complete functional model can be derived, which specifies the concrete implementation of relevant components and communication paths in more detail. This functional model is shown in Figure 6.2. Affected components have been renamed, so that their names comply with the determined implementation. Furthermore, also information and data exchanged between components have been detailed where possible.

Due to the applied modifications and concretions, the functional model differs in various aspects from the underlying implementation-independent model proposed in Part II of this thesis. In particular, the following differences can be identified:

- **Differences related to knowledge proofs:** In the implementation-independent model, coverage of the authentication factor knowledge involves the local Knowledge Prover and the remote Knowledge Verifier. The Knowledge Prover requests a knowledge proof from the Signatory and forwards this knowledge proof to the remote Knowledge Verifier for verification. Based on the decision to implement knowledge proofs by means of alphanumeric passwords, the local Knowledge Prover from the implementation-independent model is replaced by the implementation-specific Password Requester. Accordingly, the generic component Knowledge Verifier is replaced by the concrete component Password Verifier. Besides the two involved components, also data exchanges between these components are further specified by the functional model. Concretely, this model specifies that alphanumeric passwords are transmitted instead of abstract knowledge proofs between the Signatory, the local Password Requester, and the remote Password Verifier.
- **Differences related to possession proofs:** The functional model shown in Figure 6.2 also specifies the concrete implementation of the authentication factor possession. In the implementation-independent model, this authentication factor is covered by the remote components Challenge

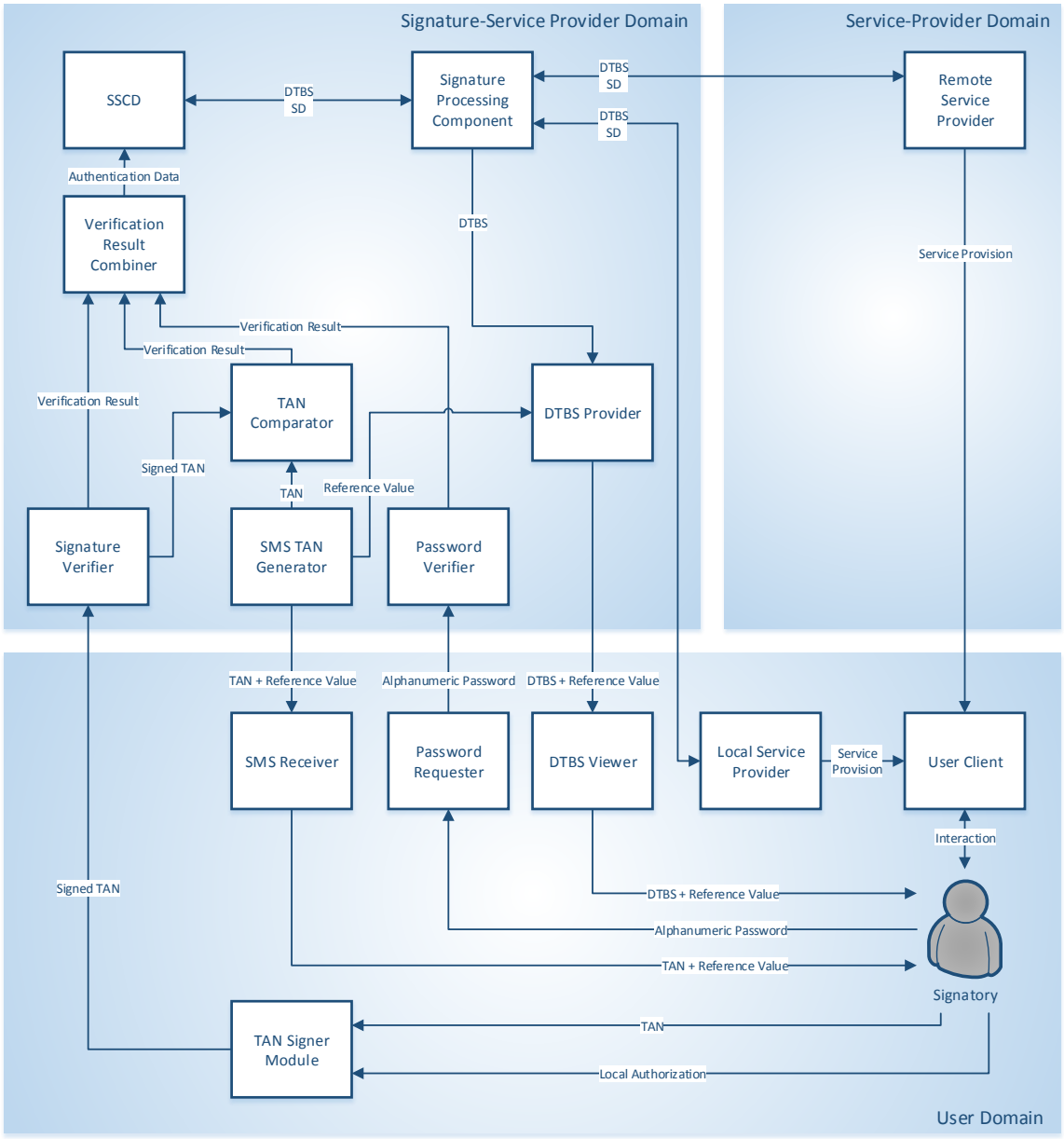

Figure 6.2: A functional model can be derived by applying the made design decisions to the developed abstract model.

Generator, Transaction Binding Verifier, and Response Verifier, as well as by the local components Challenge Receiver, Transaction Binding Verifier, and Response Creator. At the beginning of an authorization process that involves the authentication factor possession, the remote Challenge Generator creates a challenge. This challenge is sent to the local User Domain and received by the local Challenge Receiver. The Challenge Receiver forwards the challenge to the local Transaction Binding Verifier, which checks whether the obtained challenge belongs to the DTBS displayed by the DTBS Viewer. After this check, the local Response Creator computes a response from the received challenge and forwards this response to the remote Response Verifier. The Response Verifier verifies the received response and forwards it to the remote Transaction Binding Verifier. The remote Transaction Binding Verifier finally checks whether the received response belongs to the current transaction and is appropriately linked to the DTBS. In the functional model, the following

involved components and interfaces between these components have been further specified:

- **Challenges and related components:** Challenges have been determined to be implemented by means of TANs transmitted via SMS technology. Thus, the generic remote component Challenge Generator from the implementation-independent model is replaced by the concrete component SMS TAN Generator. Accordingly, the local component Challenge Receiver is replaced by the component SMS Receiver. Besides relevant components, also data exchanged between these components is further specified in the functional model. Accordingly, this model uses the implementation-specific term TAN instead of the abstract term challenge.
- **Transaction Binding Verifier:** Figure 6.2 shows that the local component Transaction Binding Verifier defined by the implementation-independent model is not part of the functional model any longer. This is due to the fact that the binding between received challenges, i.e. TANs, and DTBS is verified manually by the Signatory. This renders a technical implementation of the abstract component Transaction Binding Verifier unnecessary. However, the functionality of other involved components needs to be extended, in order to enable the Signatory to manually verify the binding between DTBS and received TANs. Concretely, the remote SMS TAN Generator needs to create an additional reference value for each generated TAN. This reference value needs to be sent together with the TAN and the DTBS in order to establish a verifiable link between these data. For this purpose, the functional model shown in Figure 6.2 introduces an additional remote component called DTBS Provider. The DTBS Provider combines DTBS obtained from the Signature Processing Component with the transaction-specific reference value created by the SMS TAN Generator and forwards these data to the local DTBS Viewer. Additionally, the created reference value is also transmitted to the local SMS Receiver together with the generated TAN. This enables the Signatory to compare the reference value displayed by the DTBS Viewer with the reference value received via SMS and to verify the link between the DTBS and the obtained TAN.
- **Response Creator:** Based on the decision to rely on asymmetric cryptographic methods for the creation of responses from obtained TANs, the implementation-independent local component Response Creator and the generic remote component Response Verifier are further specified in the functional model shown in Figure 6.2. The local component Response Creator is replaced by the implementation-specific component TAN Signer Module. The name of this implementation-specific component already describes its functionality: received TANs are signed by this component using asymmetric cryptographic methods. The signed TAN finally represents the response, which in turn acts as possession proof. This is also considered by the functional model, in which the generic term response is replaced by the implementation-specific term signed TAN. Accordingly, the implementation-independent remote component Response Verifier is replaced by the implementation-specific component Signature Verifier. Its name again indicates the main function of this component, i.e. the verification of received signed TANs.

By incorporating decisions regarding the concrete implementation of relevant components into the implementation-independent model, a complete functional model of a signature solution for mobile end-user devices has been derived. Based on this functional model, the concrete signature solution for mobile end-user devices called Smartphone Signature is developed. This solution is introduced in the following section.

6.3 Concrete Solution: The Smartphone Signature

The developed functional model refines the underlying implementation-independent model proposed in Part II of this thesis by determining the concrete implementation of relevant components. Concretely, it

defines which functionality needs to be provided by each component. Still, it does not specify how this functionality has to be realized in practice. In particular, the functional model does not take into account potential limitations that might be imposed by the special characteristics of current mobile end-user devices.

To overcome these limitations, the functional model is further developed towards a concrete solution called Smartphone Signature. For this purpose, relevant aspects that need to be taken into account when realizing mobile signature solutions on current mobile end-user devices are identified. These aspects are then mapped to the functional model. This way, the architecture and process flow of the Smartphone Signature is derived. The Smartphone Signature inherits all characteristics of the underlying functional model and additionally takes into account relevant realization aspects.

6.3.1 Realization Aspects

In order to further develop the functional model towards a concrete solution, several realization-related aspects need to be taken into account. This is necessary, as the functional model does not consider potential limitations of technologies that are needed to realize required components and communication paths. Aspects that need to be considered during the realization of a concrete signature solution for mobile end-user devices are identified and discussed in this section. Findings of this section will later be used to derive the concrete solution Smartphone Signature from the functional model.

6.3.1.1 Realization of Local Components

The functional model defines various components that need to be implemented locally in the User Domain. This includes the SMS Receiver, the Password Requester, the DTBS Viewer, and the TAN Signer Module. The functional model clearly defines the scope and the concrete functionality for each of these components. However, it does not define how these components and their functionality have to be realized in the User Domain using currently available technology.

In general, implementation and realization of local components in the User Domain are limited to mobile end-user devices. This is due to the fact that the developed mobile signature solution must not require an additional end-user device. Hence, all local components defined by the functional model must be implemented and realized on and by the Signatory's mobile end-user device.

Mobile end-user devices such as smartphones or tablet computers differ from classical end-user devices in various aspects. For development of a concrete mobile signature solution based on the functional model, especially provided features of mobile end-user devices to deploy software are relevant. In general, software deployment on mobile end-user devices differs from software deployment on classical end-user devices. Concretely, the deployment of software on mobile end-user devices is typically limited to platform-specific mobile apps. These apps are usually distributed over a central repository, which is managed and maintained by the vendor of the respective mobile platform. All developed apps that shall be deployed on mobile end-user devices have to be submitted to this central repository first. There, they are usually subject to a vendor-specific review process. Depending on the mobile platform, apps can also be obtained from alternative sources such as alternative repositories or arbitrary download locations. In any case, mobile apps represent the method of choice to deploy software on mobile end-user devices.

To overcome limitations and disadvantages of mobile apps, mobile websites are sometimes used as an alternative. Mobile websites rely on the same technologies as ordinary websites, but are tailored to the special characteristics of mobile end-user devices. For instance, mobile websites usually display reduced content in order to consider limited input and output capabilities of mobile end-user devices. In contrast to apps, mobile websites do not require the user to install software on the local device. The only requirement that needs to be satisfied in order to access a mobile website is availability of a web browser. However, web browsers are typically pre-installed on all mobile end-user devices. By incorporating enhanced web technologies such as Hypertext Markup Language (HTML) version 5, web

applications, or JavaScript frameworks, mobile websites can thus be an attractive alternative to mobile apps. However, as mobile websites are always running inside the web browser, several limitations apply to them. These limitations especially concern access to native functionality provided by the underlying mobile operating system. For solutions that require access to this native functionality, mobile websites must hence be regarded as inappropriate.

In summary, mobile apps can be identified as the most suitable approach to realize local components of mobile signature solutions on mobile end-user devices. Even though the deployment of mobile apps is more complex compared to the deployment of software on classical end-user devices, mobile apps represent the best alternative to access native functionality provided by mobile end-user devices and operating systems. Considering the functional model for mobile signature solutions, access to native functionality can be especially relevant for realizing the component TAN Signer Module. Hence, mobile apps are used to realize local components identified by the functional model.

6.3.1.2 Secure Realization of the TAN Signer Module

The derived functional model defines reliance on a challenge-response approach to authenticate the Signatory and to authorize the creation of electronic signatures in the remote SSCD. The local TAN Signer Module is a key component of the implemented approach, as it cryptographically computes responses from obtained challenges. The security of the entire authorization mechanism relies on the assumption that only the TAN Signer Module is capable to compute correct responses from obtained challenges. Therefore, the secure realization of this component is crucial for the security of the entire signature solution.

The security of locally implemented components depends to a large extent on the underlying mobile platform and mobile operating system. Similarly, also opportunities to realize cryptographic functionality on mobile end-user devices heavily depend on these aspects. A mobile signature solution based on the functional model should always realize the TAN Signer Module in the best possible, i.e. most secure, way that is feasible on the respective platform. In particular, the TAN Signer Module should be realized by means of secure hardware elements whenever possible. This way, confidential cryptographic material such as cryptographic signing keys can be protected and cryptographic operations can be carried out in a secure environment.

Unfortunately, secure hardware elements are not available and accessible on all current mobile platforms. Although such hardware elements are integrated in most mobile end-user devices, they cannot be accessed by mobile apps in most cases. So far, rudimentary access to integrated hardware elements is currently available on Google Android only. Nevertheless, concrete mobile signature solutions should at least provide the opportunity to realize the TAN Signer Module in hardware when possible and must not preclude this opportunity a priori.

6.3.1.3 Scalability

In the functional model, the Signatory represents the user, who aims to create an electronic signature with his or her mobile end-user device. While this model is suitable from a conceptual perspective, it simplifies the role of the Signatory by reducing it to a single person. In practice, mobile signature solutions need to be developed for multiple users. Depending on the concrete deployment scenario, the number of potential users can be up to several millions, e.g. when assuming deployment on a national level.

A potentially high number of users raises additional challenges for the realization of concrete signature solutions. This especially applies to solutions that follow the Server-HSM Approach and make use of one central SSCD shared by all users. According to the underlying functional model, this SSCD stores cryptographic signing keys of all users. In practice, central SSCDs are typically implemented by HSMs.

Unfortunately, HSMs are usually not able to simultaneously store a large number of cryptographic keys. Hence, HSMs seem to be inappropriate to implement central SSCDs at a first glance.

An approach to overcome this problem has been discussed by Orthacker et al. [2010] and is for instance followed by the Austrian Mobile Phone Signature, which has been in productive operation for several years. According to this approach, user-specific cryptographic signing keys are stored securely in a database outside the HSM. This way, storage-capacity limits of HSMs can be circumvented. To protect a signing key, it is wrapped inside the HSM with an internal HSM-specific key immediately after its generation. The wrapped key is additionally encrypted with an encryption key derived from the Signatory's secret password. Only the wrapped and encrypted key is stored in the database. During signature-creation processes, the Signatory is requested to provide the secret password. Using this password, the encrypted signing key is decrypted. The decrypted but still wrapped key is loaded into the HSM. By applying the correct key, the HSM unwraps the signing key, which is then ready to use for exactly one signature-creation operation. This two-step protection approach guarantees that the Signatory's signing key is bound to the Signatory and also to the HSM. The key can only be used, if the Signatory provides the correct secret password, and if it is loaded into the correct HSM.

In summary, the approach proposed and discussed by Orthacker et al. [2010] assures secure remote storage of cryptographic signing keys for signature solutions relying on central SSCDs with limited storage capacity. Thus, this approach is also applicable to solutions relying on the developed functional model. By integrating this approach, this model can be further developed towards a scalable signature solution that is capable to handle even a large number of users.

6.3.1.4 Cross-Platform Applicability

During the past years, different mobile platforms and operating systems have emerged. They all differ in terms of provided features and functionality. The functional model has been designed to be compatible to and applicable on all major platforms. This characteristic also needs to be maintained when further developing this model towards a concrete solution. Hence, when choosing realizations of relevant components and functionalities, capabilities of current mobile platforms need to be taken into account.

For instance, the functional model identifies the receiving of SMS messages as required functionality for components assigned to the User Domain. This functionality is basically available on all platforms. Depending on the particular platform, SMS-receiving functionality can however be restricted to pre-installed SMS applications. This limits possible realization variants. Hence, restrictions imposed by mobile platforms and operating systems need to be considered when choosing suitable realizations of SMS-receiving components.

Receiving SMS messages is only one example, where platform characteristics need to be considered when deriving a concrete solution from the functional model. By limiting concrete realizations to those alternatives that are feasible on all current mobile platforms, a platform-independent applicability of developed solutions can be guaranteed.

6.3.2 Architecture

By taking into account the discussed realization aspects, the architecture of the Smartphone Signature can finally be derived from the functional model. This architecture is shown in Figure 6.3 and is introduced and discussed in detail in the following.

Similar to the functional model, also the architecture of the Smartphone Signature shown in Figure 6.3 defines the three domains User Domain, Service-Provider Domain, and Signature-Service Provider Domain. To derive this architecture, all components of the underlying functional model have been mapped to corresponding building blocks. All resulting building blocks are again assigned to one of the three defined domains. Some components of the functional model such as the User Client or the

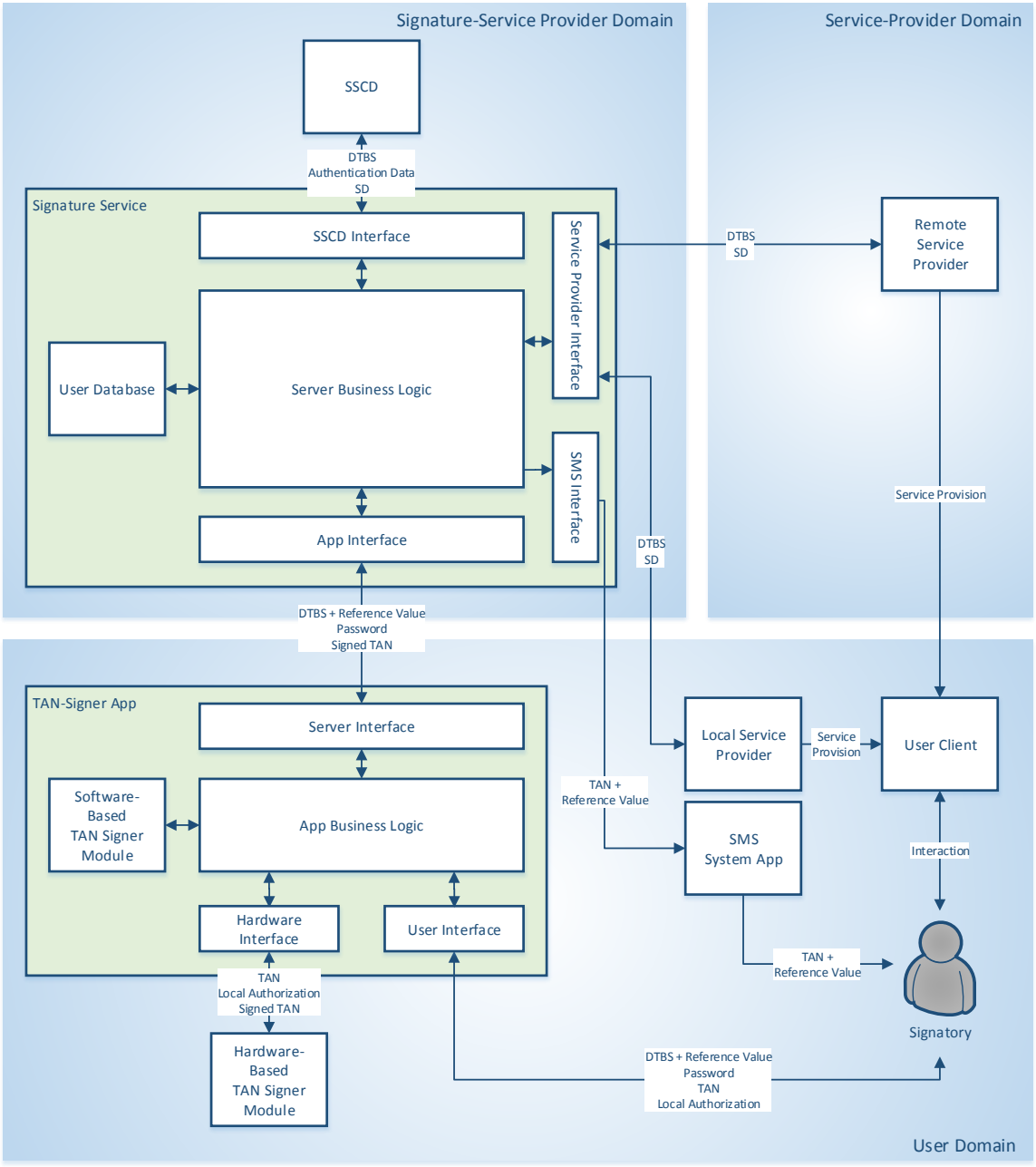

Figure 6.3: The architecture of the Smartphone Signature is based on the derived functional model and takes into account discussed realization aspects.

Local Service Provider have been directly mapped to identically named building blocks. In contrast, other components have been combined to higher-level building blocks and assigned with new names in order to avoid confusion with the functional model. All building blocks of the proposed architecture are introduced in the following subsections in more detail. This way, their scope and functionality is sketched.

6.3.2.1 Building Blocks of the User Domain

In the User Domain, the components User Client and Local Service Provider from the functional model of the Smartphone Signature are directly mapped to corresponding building blocks of the derived architecture. As these components are no integral parts of the signature-creation process, they do not need to be further detailed at this point. For the sake of clarity, the established names of these components have been reused for the resulting building blocks.

Functionalities of most other local components of the functional model have been combined and subsumed under a single local high-level building block named TAN-Signer App. As indicated by its name, this building block represents a mobile app being installed on the Signatory's mobile end-user device. By relying on a mobile app, the proposed architecture considers discussed aspects regarding the realization of local components. These discussions have yielded mobile apps as the best available approach for the realization of local components on mobile end-user devices.

The architecture shown in Figure 6.3 also identifies internal components of the local high-level building block TAN-Signer App. Its central internal component is the App Business Logic. As indicated by its name, the App Business Logic implements most functionality that is required during signature-creation processes in the local User Domain. Amongst others, it covers the functionality of the components Password Requester and DTBS Viewer from the functional model. In addition, the App Business Logic also interacts with other local and remote entities. For this purpose, it makes use of various interface building-blocks, which are also internal components of the local high-level building block TAN-Signer App. These interface building-blocks provide access and communication interfaces to external entities. For instance, the User Interface building block is used to communicate with the Signatory, while the Server Interface building block is used to exchange data with remote entities from the Signature-Service Provider Domain.

The App Business Logic covers the functionality of the components Password Requester and DTBS Viewer from the functional model. In contrast, the component TAN Signer Module defined by this model is realized as separate building block and not as part of the general App Business Logic. For security reasons, the TAN Signer Module should either be implemented in software or in hardware, depending on capabilities of the underlying mobile device. This is reflected by the architecture shown in Figure 6.3. There, the component TAN Signer Module from the functional model has been replaced by two building blocks named Software-Based TAN Signer Module and Hardware-Based TAN Signer Module. The former is an integral part of the local high-level building block TAN-Signer App and has a direct interface to the App Business Logic. In contrast, the latter is realized as separate building block outside the TAN-Signer App. The TAN-Signer App however features a Hardware Interface component, through which the App Business Logic is able to access the Hardware-Based TAN Signer Module. By mapping the TAN Signer Module to a software-based and a hardware-based building block, discussed aspects regarding the secure realization of this component are considered.

The high-level building block TAN-Signer App subsumes the functionality of most local components defined by the functional model. Exceptions are the Local Service Provider and the User Client, which have been mapped to separate building blocks, and the Hardware-Based TAN Signer Module, which also needs to be realized outside the TAN-Signer App. Additionally, also the component SMS Receiver from the functional model is realized outside the TAN-Signer App and mapped to the building block SMS System App. This building block represents the mobile device's default SMS-processing application. Even though SMS-receiving functionality could also be realized by mobile apps on certain platforms, several other platforms do not support this opportunity. In order to assure applicability on all major platforms, the component SMS Receiver from the functional model is therefore realized outside the high-level building block TAN-Signer App and relies on SMS-processing functionality provided by the mobile operating system.

6.3.2.2 Building Blocks of the Signature-Service Provider Domain

Similar to the User Domain, relevant components defined by the functional model of the Smartphone Signature have been mapped to corresponding building blocks also in the Signature-Service Provider Domain. As shown in Figure 6.3, all software-based components assigned to the Signature-Service Provider Domain have been subsumed under the high-level building block Signature Service. In addition, the component SSCD defined by the functional model has been mapped to a separate building block with the same name. Hence, the two building blocks Signature Service and the SSCD cover the entire functionality implemented in the Signature-Service Provider Domain.

For the high-level building block Signature Service, several internal components have been defined. The central component is the Server Business Logic, which covers most of the functionality needed to carry out signature-creation processes. In addition to this central component, the Signature Service also contains the User Database, which stores Signatory-related data including encrypted signing keys. This database is necessary in order to meet the requirement for scalability and can be accessed by the Server Business Logic only.

Similar to the high-level building block TAN-Signer App from the User Domain, also the Signature Service contains several interface components, which provide access to external entities. Concretely, the Signature Service contains an SSCD Interface component, providing access to the SSCD. The App Interface component is also part of the Signature Service and represents the pendant to the Server Interface component of the local TAN-Signer App. Together, they provide cross-domain communication between the TAN-Signer App and the Signature Service. For the delivery of SMS messages, the Signature Service also provides an SMS Interface component. Even though not specified in detail by the architecture shown in Figure 6.3, the SMS Interface component will in most cases access an SMS gateway to deliver SMS messages. Finally, the Signature Service also provides a Service-Provider Interface component, which enables the Server Business Logic to communicate with external Service Providers in order to receive signature-creation requests and to return results of signature-creation processes.

6.3.2.3 Building Blocks of the Service-Provider Domain

The Service Provider Domain is the third domain of the Smartphone Signature. According to the derived architecture shown in Figure 6.3, this domain contains only one building block, i.e. the Remote Service Provider. This building block has been inherited from the respective component of the functional model without modification, as it is no integral part of the signature-creation solution.

6.3.3 Process Flow

By mapping components of the functional model to concrete building blocks, a complete architecture of the Smartphone Signature has been derived. According to this architecture, a mobile app is used to implement most functionality in the User Domain. Additionally, features provided by the mobile end-user device such as the SMS System App or secure hardware elements are employed to complement the functionality of the mobile app. Required functionality of the Signature-Service Provider Domain is mainly covered by a remote Signature Service. The Signature Service provides communication interfaces to local components and represents the only building block that is capable to access the remote SSCD.

For the sake of clarity, Figure 6.3 focuses on the identification of relevant building blocks only. In addition, Figure 6.3 indicates data being exchanged between relevant building blocks. However, Figure 6.3 does not provide detailed information on the concrete processing steps that are required to complete a signature-creation process. As a complement to the architecture shown in Figure 6.3, Figure 6.4 illustrates the process flow of a typical signature-creation process.

According to the process flow illustrated in Figure 6.4, a typical signature-creation process consists of the steps depicted below. The sketched process flow represents a successful signature-creation process.

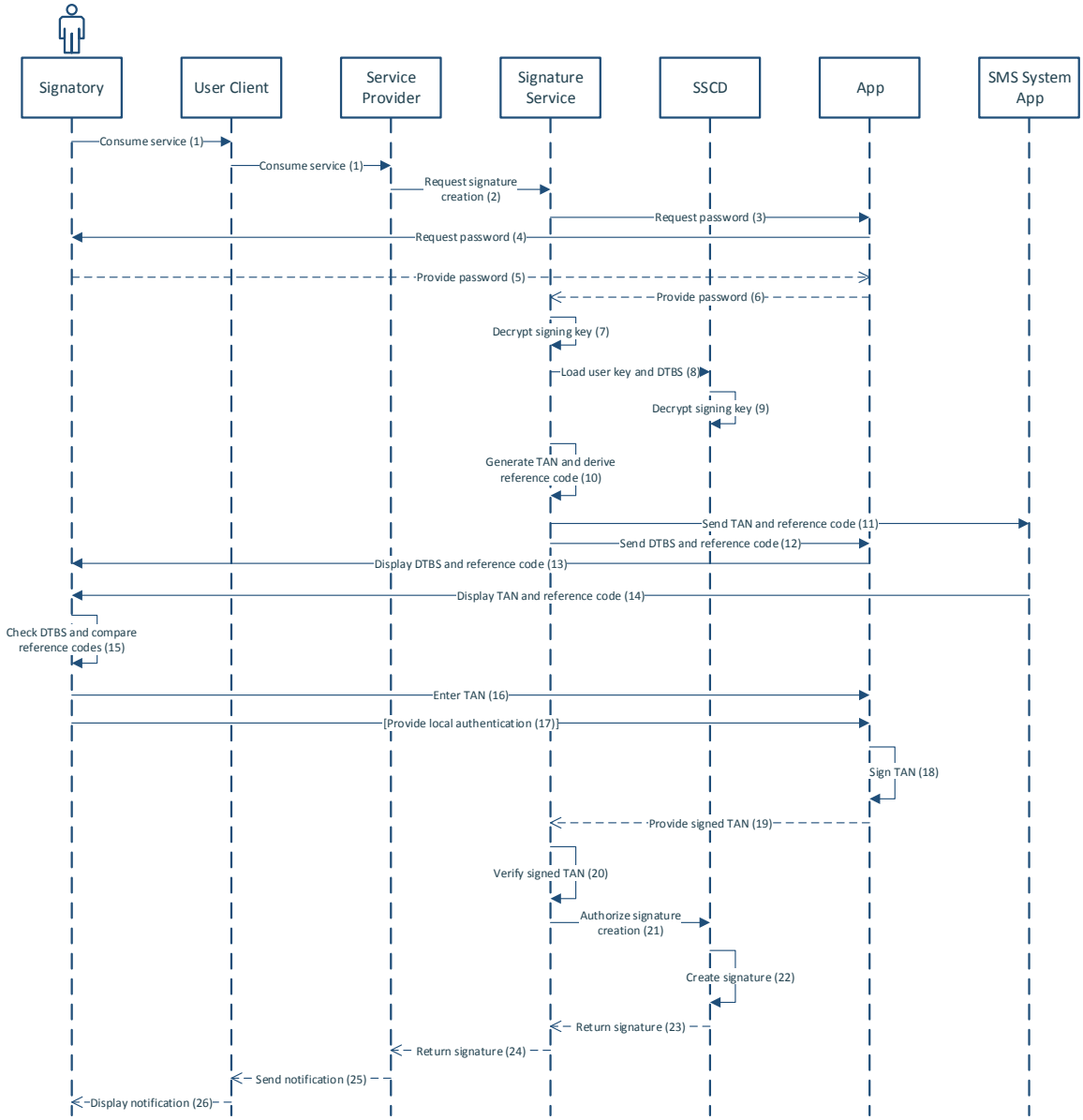

Figure 6.4: The process flow illustrates the general signature-creation process.

The handling of errors or exceptions is not considered in detail for the sake of clarity. Assuming a normal processing, the following steps are required to complete a signature-creation process using the Smartphone Signature:

1. Via the User Client, the Signatory consumes a service provided by a Service Provider. In case of a Local Service Provider, the roles of both the Service Provider and the User Client can be combined in and implemented by one single local component. In most cases, this is a mobile app being installed and running on the Signatory's mobile end-user device.

2. At some point during service provision, the Service Provider requires the Signatory to create an electronic signature. Therefore, the Service Provider sends a signature-creation request to the Signature Service. This request includes the DTBS, which are defined by the Service Provider.

3. Before signing the provided DTBS, the Signature Service needs to authenticate the Signatory. Therefore, the Signature Service starts the two factor based user-authentication process. To cover the authentication factor knowledge, the Signature Service requests the Signatory's secret password from the TAN-Signer App first.
4. The TAN-Signer App requests the Signatory to enter the secret password, for which it provides an adequate user interface.
5. Through the provided interface, the Signatory enters the secret password to the TAN-Signer App. This way, the Signatory proves knowledge of the secret password.
6. The TAN-Signer App returns the entered password to the remote Signature Service.
7. The Signature Service decrypts the Signatory's signing key with the help of the provided secret password. This is only possible, if the password provided by the Signatory is correct. Otherwise, the required decryption key cannot be derived successfully. This way, the key-derivation function implicitly verifies the provided password and hence the authentication factor knowledge.
8. The Signature Service loads the decrypted signing key and the DTBS into the SSCD.
9. The SSCD unwraps the Signatory's decrypted but still wrapped signing key.
10. The Signature Service generates a random TAN and a reference value linked to this TAN.
11. The Signature Service sends the generated TAN together with the reference value to the Signatory's mobile device, where it is received by the SMS System App.
12. At the same time, the Signature Service also sends the DTBS together with the reference value to the TAN-Signer App.
13. The TAN-Signer App displays the received DTBS together with the reference value to the Signatory.
14. The SMS System App displays the TAN together with the reference value to the Signatory.
15. The Signatory checks the displayed DTBS. In addition, the Signatory compares the two reference values, in order to verify the binding between the received TAN and the displayed DTBS.
16. If the two reference values match and the Signatory agrees to sign the displayed DTBS, he or she enters the TAN to the TAN-Signer App.
17. Optional: If required by the respective TAN Signer Module implementation, the Signatory enters local authentication data to authorize signing of the entered TAN. For instance, the signing functionality of a Hardware-Based TAN Signer Module could be protected by means of a PIN. In this case, the Signatory needs to enter this PIN in order to enable signing of the TAN.
18. The TAN-Signer App, precisely the concrete implementation of the TAN Signer Module, signs the entered TAN.
19. The TAN-Signer App returns the signed TAN to the Signature Service.
20. After reception of the signed TAN, the Signature Service verifies the signed TAN.
21. If the signed TAN can be verified positively, the Signature Service authorizes the signature-creation process in the SSCD.
22. The SSCD creates the electronic signature on the provided DTBS with the Signatory's decrypted signing key.

23. The SSCD returns the SD, i.e. the created signature, to the Signature Service.
24. The Signature Service returns the SD, i.e. the created signature, to the Service Provider.
25. The Service Provider notifies the User Client about the successful completion of the signature-creation process.
26. The User Client informs the Signatory of the successful completion of the signature-creation process.

This process flow shows, how building blocks of the architecture interact with each other in order to produce an electronic signature. This way, this process flow complements the derived architecture. Together, the architecture shown in Figure 6.3 and the process flow illustrated in Figure 6.4 describe the proposed concrete signature-creation solution Smartphone Signature.

6.4 Chapter Conclusions

In this chapter, a concrete signature-creation solution for mobile end-user devices called Smartphone Signature has been proposed. For this purpose, a set of design principles has been defined first. Based on these design principles, a functional model has been developed. From this model, the proposed solution has finally been derived by taking into account restrictions imposed by currently available mobile technologies. The proposed solution, i.e. the Smartphone Signature, has been described by means of an architecture and a process flow, which can both serve as basis for arbitrary implementations.

The Smartphone Signature is based on the implementation-independent and technology-agnostic model proposed in Part II of this thesis. This model has been systematically developed by assessing different approaches for mobile signature solutions and by determining the best possible approach to implement secure and reliable authorization mechanisms. In this chapter, this model has been further developed towards a concrete solution.

As the Smartphone Signature bases on the proposed implementation-independent model, it inherits all its characteristics. In particular, it follows the Server-HSM Approach and relies on a challenge response based authorization mechanism. Thus, the Smartphone Signature shows that the proposed model is a suitable basis for the development of concrete signature solutions for mobile end-user devices. This way, the Smartphone Signature represents a first evaluation of the proposed model.

To complete this evaluation, the Smartphone Signature has been implemented in practice. This implementation shows that the Smartphone Signature and hence also the underlying model can serve as basis for real-world applications. Details of this implementation are presented and discussed in the following chapter.

Chapter 7

Implementation

“ Well done is better than well said.”

[Benjamin Franklin, American Polymath.]

Following this thesis’s methodology, proposed and developed models and signature solutions have been kept on different levels of abstraction. This assures sustainability in the mobile-computing domain, which is subject to ephemeral trends and fast-evolving technologies. In this chapter, the above quotation by Benjamin Franklin is taken literally and the developed signature solution Smartphone Signature is finally implemented. Because of the special characteristics of mobile platforms, limited capabilities of software running on these platforms, and the steadily increasing fragmentation of mobile operating systems and end-user devices, the feasibility and applicability of mobile solutions is a crucial aspect. This especially applies to mobile solutions that require special features such as cryptographic functionality. Hence, this also applies to the Smartphone Signature, which has been proposed and introduced in Chapter 6 in detail. According to its underlying model proposed in Part II of this thesis, the Smartphone Signature requires the secure application of cryptographic functionality on the mobile end-user device. Hence, the feasibility and applicability of the Smartphone Signature on current mobile platforms needs to be assessed and evaluated. Only if the Smartphone Signature is feasible and applicable in practice and with available state-of-the-art technology, it can be regarded as appropriate.

To prove its feasibility in practice, the Smartphone Signature has been implemented for common mobile end-user devices. This implementation is presented and discussed in this chapter in detail. The presented implementation of the Smartphone Signature finally completes the evaluation of its underlying model derived and proposed in Part II of this thesis. In Chapter 6, it has been shown that this model can be further developed towards a concrete solution that builds on existing technologies. The practical implementation presented in this chapter goes one step further and shows that the developed concrete solution is indeed feasible in practice. The presented implementation covers all relevant use cases of mobile signature solutions. Furthermore, it integrates three different mobile cutting-edge technologies to realize required cryptographic functionality on the mobile end-user device. Concretely, it supports the use of Secure Elements (SEs), cryptography-enabled NFC tokens, and KeyChain implementations of current mobile platforms for this purpose. The feasibility of the presented implementation finally proves that the model proposed in Part II of this thesis is not restricted to serve as basis for concrete solutions, but can also be the basis for practical implementations.

The implementation presented in this chapter has been designed and realized such that it can be easily adapted according to future technological developments. For this purpose, it has been based on a deliberate design and architecture assuring a high level of flexibility and adaptability. Its flexibility and adaptability have also been shown by integration of different technologies for the realization of cryptographic functionality. Thus, the presented implementation does not only serve as evaluation of the

Smartphone Signature and its underlying model, it also represents a solid basis for a productive mobile signature solution. This way, the presented implementation paves the way for the application of qualified electronic signature on mobile end-user devices and represents a considerable step towards transactional m-government services.

To provide a comprehensive insight into the developed implementation, the remainder of this chapter is structured as follows. First, the mobile signature solution ServerBKU, which acts as basis for the presented implementation, is briefly introduced. Subsequently, design and architecture of the presented implementation are introduced. Special focus is put on strategies that have been pursued to assure flexibility and adaptability. Finally, the concrete realization of the presented implementation is presented and discussed in detail. Thereby, the integration of different mobile cutting-edge technologies for the realization of required cryptographic functionality on the mobile end-user device is especially emphasized.

7.1 Implementation Basis: The ServerBKU

The implementation presented in this chapter is based on an existing signature solution. This solution has been adapted and enhanced in order to comply with the architecture and process flow of the mobile signature solution Smartphone Signature proposed in Chapter 6. Basing the implementation on an existing solution is reasonable for several reasons. First, time-tested components and functionalities of this solution can be re-used. This reduces development time. Second, the re-use of existing functionality enables the bundling of resources and allows to concentrate on relevant aspects. Third, reliance on an existing solution implicitly assesses the compatibility of the Smartphone Signature with existing implementations. For all these reasons, the presented implementation has been based on an existing mobile signature solution.

Based on an analysis of available alternatives, the presented implementation has finally been based on the mobile signature solution called ServerBKU, which has been introduced by Rath et al. [2014a]. The ServerBKU relies on the same concepts as the Austrian Mobile Phone Signature. It follows the Server-HSM Approach and relies on TANs delivered by SMS to authenticate users. However, the ServerBKU is not explicitly tailored to the Austrian eID infrastructure. Therefore, it shows a higher degree of flexibility regarding different deployment scenarios. As its source code has been available, the ServerBKU has been perfectly suitable to act as basis for the implementation of the mobile signature solution proposed and developed in this thesis.

In this section, the ServerBKU is introduced briefly. We have presented and discussed details of this solution in Rath et al. [2014a]. This section recaps the most relevant aspects of this publication and focuses on those aspects that are especially relevant for the presented implementation of the Smartphone Signature. In particular, this section introduces underlying concept of the ServerBKU, discusses its design principles, and shows its general architecture. This way, a basic understanding of the ServerBKU is achieved.

7.1.1 Concept

The ServerBKU relies on the concept for qualified server-based signatures proposed and discussed by Orthacker et al. [2010]. This concept follows the Server-HSM Approach. Accordingly, electronic signatures are created inside a server-based SSCD.

Orthacker et al. [2010] identify the reliable authentication of the user and the secure central storage of SCD, i.e. cryptographic signing keys, as crucial for server-based signature solutions. Therefore, their concept mainly focuses on these two aspects. For the reliable authentication of users, they propose reliance on an SMS TAN based authentication scheme. After entering a user-specific secret password, a TAN, i.e. a one-time password, is sent to the user's mobile phone via SMS. The user has to enter the received TAN to complete the authentication process and to authorize the signature creation in the

remote SSCD. For the secure central storage of SCD, Orthacker et al. [2010] propose the application of encryption schemes. Concretely, SCD are encrypted and wrapped such that decryption and unwrapping is only possible inside the SSCD and requires interaction of the legitimate user. This way, SCD are bound to the user and to the SSCD. Unauthorized users are not able to decrypt SCD and use them to create signatures on behalf of the legitimate user.

From this concept, a basic architecture can be derived. This architecture is shown in Figure 7.1 and has been aligned with common notations used throughout this thesis.

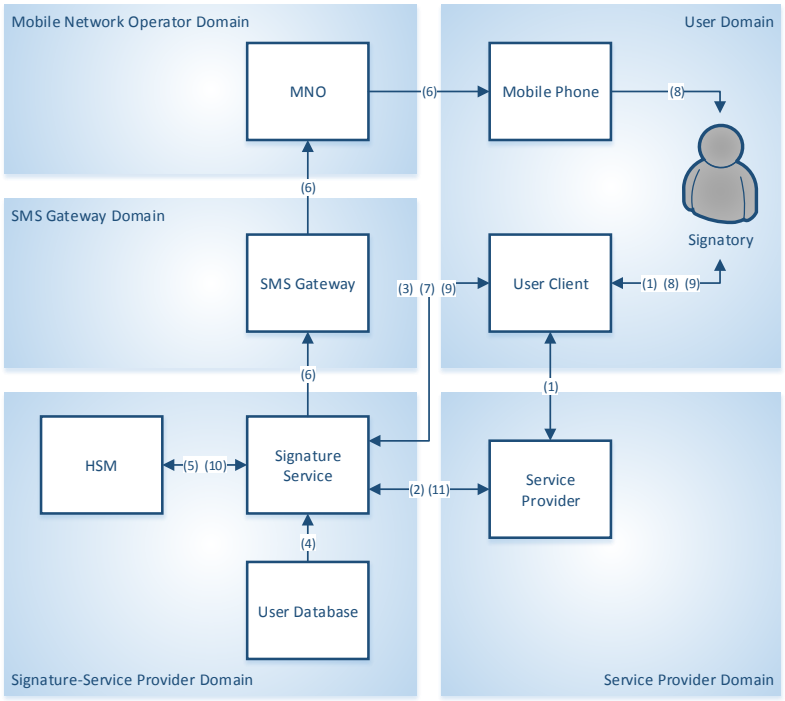

Figure 7.1: A general architecture can be derived from the concept proposed by Orthacker et al. [2010].

According to this concept, five relevant domains can be identified. Each domain contains one or more components that interact with each other during signature-creation processes. The User Domain contains the Signatory and the two components User Client and Mobile Phone. Through these components, the Signatory interacts with the signature solution. The Service Provider Domain contains only one component, i.e. the Service Provider. The Service Provider is accessed by the Signatory using the User Client. Furthermore, the Service Provider interacts with components of the Signature-Service Provider Domain to trigger signature-creation processes and to receive results of these processes. The Signature-Service Provider Domain contains all components required for signature-creation processes. This includes the HSM, which acts as SSCD, and the Signature Service, which provides interfaces to the Service Provider Domain and to the User Domain. Finally, the architecture shown in Figure 7.1 defines also two domains related to the delivery of SMS-based TANs. The SMS Gateway Domain contains the SMS Gateway, which facilitates the delivery of SMS messages. The Mobile Network Operator Domain contains the MNO, which receives SMS messages from the SMS Gateway and forwards them to the Signatory's Mobile Phone.

From the concept proposed by Orthacker et al. [2010] and from the derived general architecture shown in Figure 7.1, the process flow of a typical signature-creation process can be derived. This process flow comprises the following steps, which are also sketched in Figure 7.1.

1. The Signatory consumes a service provided by the Service Provider by means of the User Client.
2. The Service Provider requests the Signatory to create an electronic signature. For this purpose, the Service Provider sends a signature-creation request to the Signature Service.
3. The Signature Service requests the Signatory to enter the mobile-phone number and the Signatory-specific secret password.
4. The Signature Service identifies the Signatory by means of the provided mobile-phone number and loads the Signatory's encrypted SCD from the User Database.
5. The encrypted SCD is decrypted with the help of the provided secret password.
6. The decrypted SCD is loaded into the HSM, where it is unwrapped.
7. After successful decryption and unwrapping of the SCD, the Signature Service generates a random TAN and a reference value. The TAN and the reference value are sent to the Signatory's Mobile Phone via the SMS Gateway and the MNO.
8. At the same time, the DTBS are displayed by the Signature Service in the User Client together with the reference value.
9. The Signatory compares the reference value that has been received together with the TAN with the reference value displayed together with the DTBS.
10. If the reference values match, the Signatory enters the received TAN into the User Client and sends the TAN to the Signature Service.
11. The Signature Service verifies the received TAN. If the TAN is correct, it authorizes the signature creation in the HSM.
12. The Signature Service returns the result of the signature-creation process to the Service Provider.

The architecture and process flow that have been derived from the general concept resemble in several aspects the proposed mobile signature solution Smartphone Signature. This is comprehensible, as both the concept introduced by Orthacker et al. [2010] and the Smartphone Signature follow the Server-HSM Approach. However, there are two significant differences between them. First, the concept proposed by Orthacker et al. [2010] does not rely on a challenge-response approach to authenticate the Signatory. Second, it specifies the use of two separate end-user devices represented by the components User Client and Mobile Phone. The security of the concept proposed by Orthacker et al. [2010] relies on the assumption that these components are realized on separate devices.

7.1.2 Design Principles

The concept for server-based signature solutions proposed by Orthacker et al. [2010] mainly focuses on the secure remote storage of SCD and on suitable authentications schemes. Other relevant aspects of signature-creation solutions are not considered in detail. These limitations are also reflected by the general architecture and process flow that have been derived from the proposed concept. Concretely, the following limitations can be identified:

- **Focus on signature creation:** Orthacker et al. [2010] mainly focus on the signature-creation process. However, signature solutions also need to consider other processes related to the creation of electronic signatures. For instance, most solutions require a preceding personalization process, in which SCD are created for the Signatory and authentication data are defined and linked

to the created SCD. Signature-creation processes can only be carried out after a successful personalization process. The concept proposed by Orthacker et al. [2010] does not explicitly cover personalization-related aspects or other required processes.

- **Focus on specific deployment scenario:** To a certain extent, the concept proposed by Orthacker et al. [2010] has been customized for a use in the EU member state Austria. For instance, Orthacker et al. [2010] state that their concept is aligned with Austrian law. On the one hand, this is beneficial as it shows that the proposed concept can be successfully applied in a concrete use case. On the other hand, customization to a specific deployment scenario renders application of the proposed concept in other environments difficult.

Concrete signature solutions need to handle these limitations and complement aspects not covered by the proposed concept. Only if all relevant aspects are covered, the signature solution is able to provide all required functionality. Therefore, development of the ServerBKU has been based on a set of design principles. These design principles complement the underlying concept where necessary. Together with the general concept proposed by Orthacker et al. [2010], the design principles represent the basis for the ServerBKU. All specified design principles are defined and discussed in the following subsections.

7.1.2.1 Support of Relevant Processes

While the underlying concept mainly focuses on the signature-creation process, three relevant processes can be defined for the ServerBKU. These processes cover different use cases related to the creation of electronic signatures. Concretely, the ServerBKU needs to implement the three processes Registration, Activation, and Signature Creation.

- **Registration:** The registration process has to be carried out once by each Signatory prior to all other processes. During this process, the identity of the Signatory is verified and a user account is created. This user account is a prerequisite for the other two specified processes.

- **Activation:** The activation process can only be carried out after a successful registration. To run this process, the Signatory needs to log in to the user account that has been created during registration. During the activation process, a new virtual signature token is created. This includes the generation of a new key pair for the Signatory and the issuing of a corresponding certificate. Furthermore, the Signatory defines authentication data to protect the created keys, i.e. the SCD. This includes binding the Signatory's mobile phone to the created key pair. For this purpose, the Signatory needs to enter his or her mobile phone number. To verify the entered number, a one-time password is sent to the mobile phone via SMS. The Signatory needs to enter this one-time password to prove possession of the mobile phone. The activation process needs to be run at least once, before the Signatory is able to create electronic signatures. However, the activation process can also be run multiple times by each user. In each run, a new virtual signature token is generated and bound to the Signatory's mobile phone. This allows the Signatory to maintain multiple tokens. Each token is uniquely identified by its associated phone number and password, which enables the Signatory to select the preferred one for signature creation.

- **Signature Creation:** As soon as registration and activation have been run at least once, the signature-creation process can be carried out as often as desired. During this process, the Signatory selects the preferred virtual signature token and provides the chosen authentication data, i.e. the secret password, for this token. A TAN is then created and sent to the Signatory's mobile phone. By entering this TAN, the Signatory proves possession of the mobile phone, completes the authentication process, and authorizes the remote signature creation.

By clearly assigning relevant use cases to the three processes Registration, Activation, and Signature Creation, all relevant aspects of mobile signature solutions are covered. Implementing these three processes assures that the ServerBKU provides all functionality required for a mobile signature solution.

7.1.2.2 Flexibility

The concept proposed by Orthacker et al. [2010] has been tailored to the special use case of Austria. Therefore, it is optimized for this particular deployment scenario, but does not explicitly consider potentially varying requirements of other scenarios. In order to assure applicability in arbitrary deployment scenarios, an adequate level of flexibility needs to be provided by concrete implementations such as the ServerBKU that build on top of this concept.

This applies for instance to the interaction with external components. Such components are certification authorities, which issue certificates during the activation process, or the MNO, which is required to transmit TANs to the Signatory. In order to assure that the ServerBKU is applicable in different scenarios, it must not be limited to certain external components. Instead, the ServerBKU must provide a sufficient level of flexibility to enable interaction with various external components. This way, the ServerBKU can overcome limitations of the underlying concept and can be prepared for arbitrary deployment scenarios.

Another aspect, where a sufficient level of flexibility is mandatory, is the deployment and operation of mobile signature solutions. Different deployment scenarios usually define different requirements regarding the operation of signature solutions. In order to achieve compatibility with arbitrary scenarios, the ServerBKU shall hence provide a sufficient level of flexibility that facilitates its deployment and operation in different environments.

7.1.2.3 Security

Security is a central requirement of mobile signature solutions. The concept proposed by Orthacker et al. [2010] considers this requirement by specifying a complex user authentication and authorization mechanism and by defining the storage of SCD in encrypted form. Nevertheless, security also needs to be considered when developing and deploying concrete signature solutions based on this general concept. In this context, also other aspects than user authentication and secure storage of SCD need to be taken into account, even if these aspects are not covered in detail by the underlying concept.

7.1.3 Architecture

From the general concept for server-based signature solutions proposed by Orthacker et al. [2010] and from the defined set of design principles, the architecture of the ServerBKU has been derived. As the concept proposed by Orthacker et al. [2010] represents the underlying basis, the general architecture derived from this concept and shown in Figure 7.1 on pare 193 is a suitable starting point. This general architecture shows that most functionality of the signature solution is implemented in the Signature-Service Provider Domain. The local User Domain only contains the Signatory and the two components User Client and Mobile Phone. These components are mainly used to interact with the signature-creation solution and to provide required authentication data. However, these components are not directly involved in the signature-creation process itself. Signatures are created solely in the Signature-Service Provider Domain. Hence, focus has been put on this domain when deriving a suitable architecture for the ServerBKU.

The Signature-Service Provider Domain contains the three components HSM, Signature Service, and User Database. The role and the implementation of the HSM is rather clear. This component is implemented by a secure hardware element that is able to securely store and process data and to create electronic signatures on behalf of the Signatory. Products that cover this functionality already exist. Hence, this component does not need to be detailed further.

In contrast, the functionality of the Signature Service and the User Database is more complex. Especially the Signature Service acts as central component in the Signature-Service Provider Domain and needs to provide various functionalities. Furthermore, this component needs to implement interfaces to various internal and external components. Hence, this component needs to be refined when further developing the abstract architecture shown in Figure 7.1 on page 193 towards a concrete architecture. This refinement yields the ServerBKU architecture shown in Figure 7.2. Even though applied refinements concern components of the Signature-Service Provider Domain only, the ServerBKU's architecture shown in Figure 7.2 again outlines all involved domains for the sake of completeness.

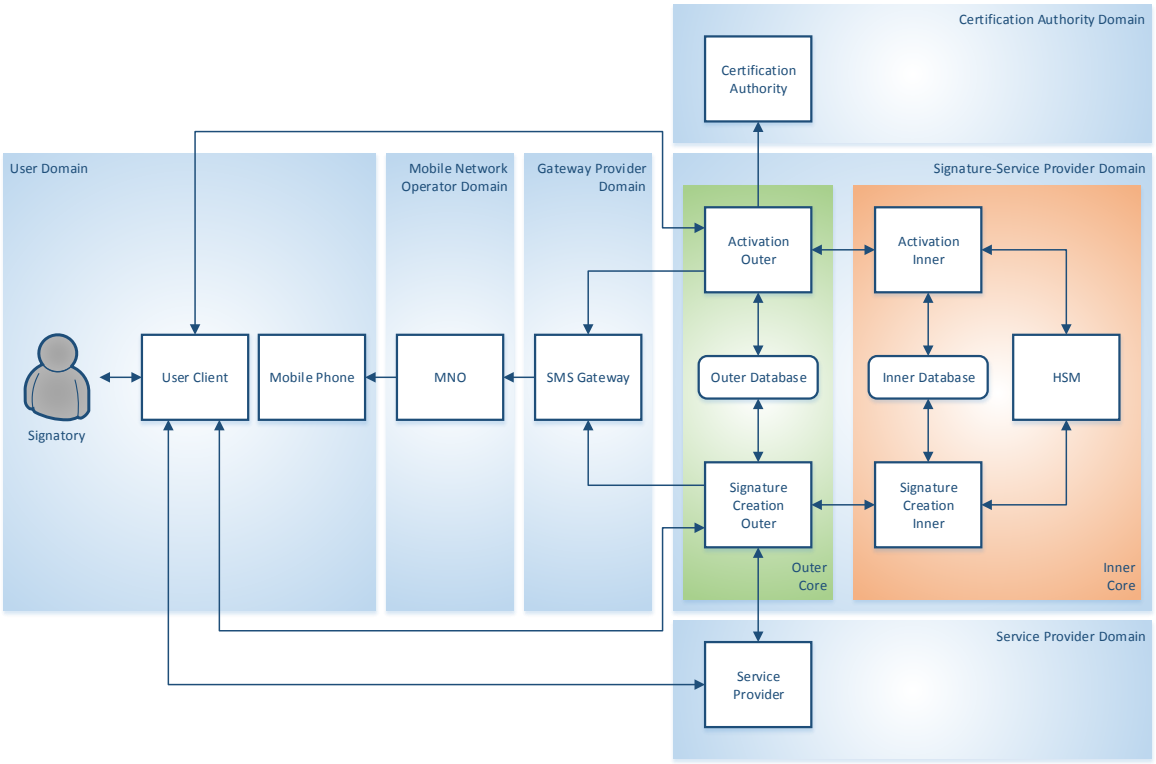

Figure 7.2: The architecture of the ServerBKU is derived from the concept proposed by Orthacker et al. [2010] and the defined design principles.

Figure 7.2 shows that the Signature-Service Provider Domain itself has been split into an Outer Core and an Inner Core. All relevant components are implemented in one of these two cores. The Inner Core comprises the HSM and components with direct interfaces to the HSM. The Outer Core comprises those components that have interfaces to entities from other domains. Components of the Inner Core and the Outer Core communicate with each other via well-defined internal interfaces. The conceptual split of the Signature-Service Provider Domain into an Inner Core and an Outer Core enhances the ServerBKU's flexibility regarding deployment and operation. Security-critical components such as the HSM can be deployed in a different environment than components that require external interfaces. This way, security-critical components can be operated in especially protected environments. This, in turn, improves the security of the entire solution. By relying on an Inner Core and an Outer Core, the ServerBKU hence complies with the defined design principles that demand flexibility and security.

Similar to the abstract architecture derived from the underlying concept, also the architecture shown in Figure 7.2 contains the component HSM. The HSM is assigned to the Inner Core. The other two components from the Signature-Service Provider Domain defined by the abstract architecture are further refined. Concretely, this concerns the components Signature Service and User Database. These two components are split into corresponding inner and outer components, in order to meet the general

architectural design. The User Database is split into an Inner Database assigned to the Inner Core and an Outer Database assigned to the Outer Core. Accordingly, the Signature Service is split into an inner and an outer component too. Additionally, this component is also split into an activation and a signature creation component. This finally yields the following four components that together cover the functionality of the abstract component Signature Service: Activation Outer, Activation Inner, Signature Creation Outer, Signature Creation Inner. The components Activation Outer and Signature Creation Outer are implemented in the Outer Core, whereas the components Activation Inner and Signature Creation Inner are assigned to the Inner Core.

Splitting the abstract components Signature Service and User Database into multiple components has several advantages. All functionality that involves the processing or storage of security-critical data can be assigned to and implemented by components of the Inner Core. As components of the Inner Core do not have any interfaces to external components, they can be deployed and operated in an especially protected environment. This way, the ServerBKU fulfills the design principles that demand a sufficient level of security and flexibility regarding deployment. Flexibility regarding deployment and operation is also achieved by splitting components according to supported processes. This way, components involved in the activation process can be clearly separated from components involved in the signature-creation process. If desired, relevant components can thus be deployed and operated in different environments depending on their purpose. This again maintains the flexibility of the ServerBKU while fulfilling the design principle regarding support of all relevant processes.

The architecture shown in Figure 7.2 considers the two processes Activation and Signature Creation. For each process, the architecture defines two components that cover the respective process's functionality. However, the architecture does intentionally not define components to cover the process Registration. There are two reasons for that. First, the registration process needs to be run only once by each Signatory. Hence, this process can also be covered by non-technical means. For instance, registration can be carried out by an authorized registration officer, who manually verifies the identity of the Signatory. Second, the registration process can also be implemented by various different technical means. Covering all possible means would significantly increase the complexity of the architecture shown in Figure 7.2 and reduce its clarity. For these reasons, the process registration is not explicitly considered. We have discussed all registration opportunities supported by the ServerBKU in detail in Rath et al. [2014a]. For the scope of this thesis, it is however sufficient to focus on the two processes Activation and Signature Creation and to assume that the process Registration is carried out by non-technical means.

Following the derived architecture, the ServerBKU complies with all defined design principles while still relying on the general concept proposed by Orthacker et al. [2010]. The ServerBKU supports all relevant processes including Activation and Signature Creation. By splitting abstract components into several subordinate building blocks, the ServerBKU also achieves the demanded flexibility regarding deployment and operation. Finally, also the demand for security is fulfilled by implementing security-critical components in the Inner Core, which can be sufficiently protected. This way, all design principles are already considered on architectural level. The next section shows how this architecture has been further developed towards a concrete solution and introduces the implementation of the ServerBKU in detail.

7.1.4 Implementation

In this section, the concrete implementation of the ServerBKU is introduced. This implementation is based on the architecture shown in Figure 7.2. In order to map this architecture to a concrete implementation, an adequate technology has been chosen first. Then, suitable development frameworks have been selected for this technology. The selected frameworks have been used to implement the ServerBKU and all its processes. The following subsections elaborate on the basic steps that have been followed to implement the ServerBKU.

7.1.4.1 Choice of Technology

The architecture shown in Figure 7.2 identifies components and building blocks of the ServerBKU but does not specify their concrete realization. To further develop this rather abstract architecture towards a concrete solution, an adequate underlying technology needs to be chosen. Based on this choice, concrete implementations of identified components can be determined.

For the ServerBKU, web technologies have been selected as technology of choice. Accordingly, the ServerBKU has been implemented as web application. Reliance on web technologies and realization of the ServerBKU in the form of a web application has several advantages. First, web technologies also build the basis of the Austrian Mobile Phone Signature. This shows that these technologies are suitable for the realization of signature solutions following the Server-HSM Approach. Second, web technologies have proven their capabilities in various security-critical fields of applications such as e-banking or e-commerce. Existing solutions from these fields show that web technologies provide mechanisms to implement and provide functionality in a secure and reliable way. Finally, web technologies release the Signatory from the need for special local software. Instead, arbitrary web browsers can be used to cover the functionality of the local component User Client and to interact with remote web-based components. For all these reasons, selecting web technologies as fundamental basis and implementation of the ServerBKU as web application is appropriate.

Taking into account the decision to implement the ServerBKU as web application, its architecture can be further refined. In particular, the implementation of relevant components can be determined. Due to the chosen web-based nature of the ServerBKU, the four components of the Signature-Service Provider Domain, i.e. the components Activation Outer, Activation Inner, Signature Creation Outer, and Signature Creation Inner are implemented as separate web modules. These modules have been named Public Activator, Private Activator, Public Signature Creator, and Private Signature Creator. Public modules are implemented in the Outer Core and feature web-based interfaces to the Signatory and to external components. In contrast, the two private modules are implemented in the Inner Core and do not provide public interfaces for external access. In general, each web module represents a subordinate web application. Together, they build the superordinate ServerBKU web application. The local component User Client has direct interfaces to two of these web modules. Hence, its implementation needs to be aligned with those of these web modules. Considering their implementation as web application, the User Client can be determined to be implemented as web browser. The two decisions to implement components of the Signature-Service Provider Domain as web modules and to realize the local User Client as web browser can be incorporated into the architecture of the ServerBKU. This yields the refined architecture shown in Figure 7.3. Components that are affected by the made decisions are highlighted.

7.1.4.2 Employed Development Frameworks

Implementation of the ServerBKU according to the refined architecture shown in Figure 7.3 has been based on approved development frameworks. Due to the popularity of web technologies, numerous frameworks that facilitate the development of web-based solutions have been introduced during the past few decades. Reliance on these frameworks is reasonable for several reasons. For instance, these frameworks improve the entire development process by enforcing reliance on approved design patterns. Furthermore, they typically provide libraries that cover common functionality. This releases developers from implementing recurring functionality on their own and hence reduces the probability of implementation errors. To exploit their various advantages, the ServerBKU has been based on the following development frameworks for web applications:

- **Spring:** All web applications that compose the ServerBKU have been based on the Spring framework[1]. This framework facilitates the development of Java-based applications and enforces re-

[1] http://spring.io/

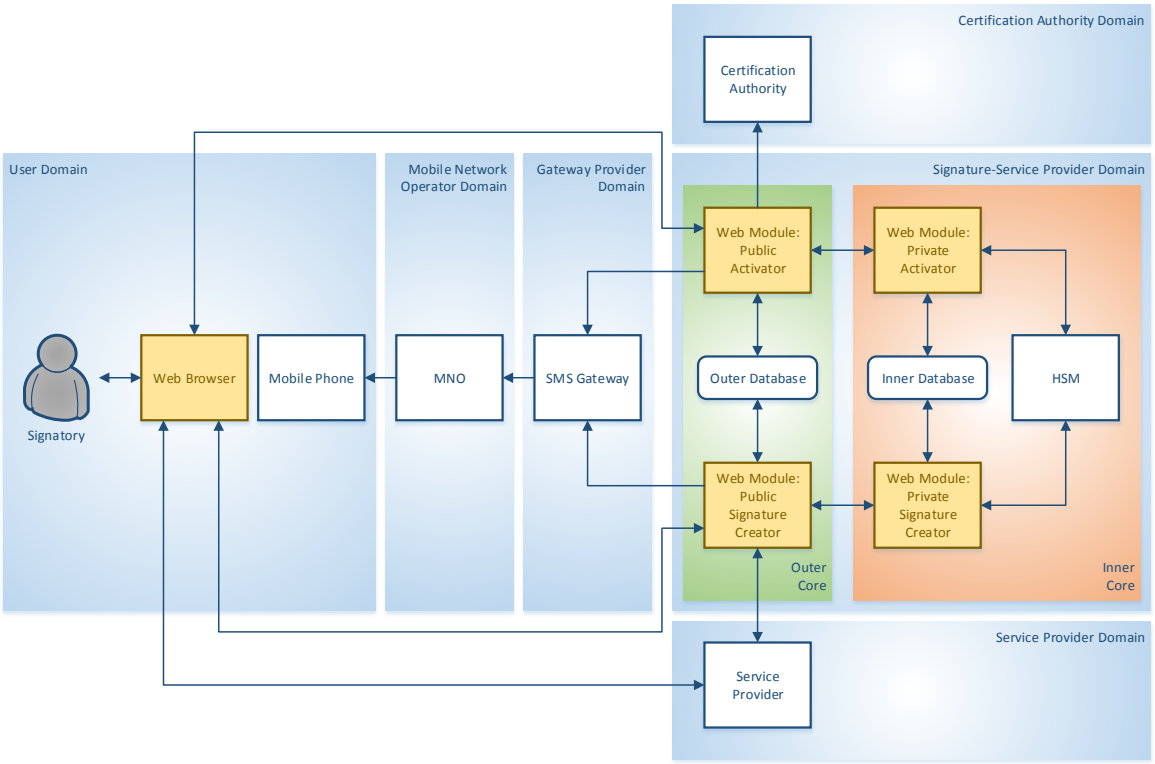

Figure 7.3: The architecture of the ServerBKU can be refined by taking into account the decision to rely on web technologies.

liance on several programming principles such as Dependency Injection (DI) and Aspect-Oriented Programming (AOP). The Spring framework is frequently used for the development of professional web applications.

- **Java Server Faces (JSF) and Primefaces:** JSF[2] and Primefaces[3] have been used to implement required user interfaces. These frameworks provide a rich set of user-interface components and hence speed up the development process.

- **Hibernate:** The Hibernate framework[4] has been used to access databases from web applications. Hibernate represents an object-relational mapping library and implements an abstraction layer between a Java-based web application and a relational database. As such, Hibernate facilitates the storage, retrieval, and modification of persistent data in web applications.

- **IAIK Security Libraries:** The ServerBKU needs to implement various cryptographic operations. These operations have been implemented with the help of the IAIK Java Cryptography Extension (JCE) and iSaSiLk libraries[5].

- **Apache ActiveMQ:** The split of components into several subordinate modules raises the need for means to exchange messages between these modules. In particular, these means are required to exchange messages between the Inner Core and the Outer Core of the ServerBKU. The implementation of this cross-core exchange of messages has been based on the Apache ActiveMQ

[2] https://javaserverfaces.java.net/
[3] http://primefaces.org/
[4] http://www.hibernate.org/
[5] http://jce.iaik.tugraz.at/

framework[6].

All functionality of the ServerBKU has been implemented based on these libraries and frameworks. The concrete implementation of the ServerBKU's relevant processes is discussed in the next section.

7.1.4.3 Implementation of Processes

The functionality of the ServerBKU is mainly covered by the four web modules implemented in the Signature-Service Provider Domain. Each module covers a well-defined set of functionality. By combining the functionality of all four modules, all required processes are implemented. In particular, the ServerBKU supports the registration of Signatories, the activation of virtual signature tokens, and signature-creation processes with these tokens. In this section, the functionality of the ServerBKU is illustrated by means of its implemented processes. As the registration of Signatories is regarded as out of scope for this thesis, main focus is put on the processes Activation and Signature Creation.

Functionality related to the process Activation is covered by the web modules Public Activator and Private Activator. By going through a typical activation process, functionality provided by these two modules is illustrated. In particular, the activation process comprises the following steps:

- **Log in to user account:** To start the activation process, the registered Signatory has to log in to the ServerBKU in order to access his or her personal user account, which has been created during the registration process. For this purpose, the Public Activator provides a web-based user interface. This way, the Signatory can access functionality provided by the Public Activator via a web browser. After log-in, the Signatory activates a new virtual signature token. This token can subsequently be used for the creation of electronic signatures. In principle, each Signatory can create an arbitrary number of virtual tokens.

- **Specify required data:** For the activation of a new token, the Public Activator provides a simple web form. Through this web form, the Signatory enters and specifies required data. This includes the Signatory's mobile-phone number secret password assigned to the token. These data are sent to the Public Activator via a Hypertext Transfer Protocol (HTTP) Post request and are stored in the Outer Database being part of the Outer Core.

- **Verify mobile-phone number:** To verify the provided mobile-phone number, the Public Activator sends an activation code to the Signatory's mobile phone via SMS with the help of the SMS Gateway. The SMS Gateway is provided by an external entity and is not integral part of the ServerBKU. The user needs to enter the received activation code into a web form provided by the Public Activator in order to prove possession of and control over the mobile phone. The entered activation code is sent again via HTTP Post to the Public Activator for verification.

- **Generate key pair:** If the mobile-phone number can be verified successfully, a new key pair is created for the Signatory and stored wrapped and encrypted in the Inner Database. This functionality is mainly covered by the Private Activator. The key-pair generation is triggered by the Public Activator, which is the only component outside the Inner Core that is able to communicate with the Private Activator. The Private Activator in turn relies on the HSM to carry out the required cryptographic functionality. Concretely, the HSM creates the key pair and wraps the private key according to the concept proposed by Orthacker et al. [2010]. The wrapped private key is finally exported to the Private Activator, encrypted, and stored in the Inner Database. This way, the private key never leaves the Inner Core.

[6] http://activemq.apache.org/

- **Issue certificate:** The public key of the key pair created in the HSM is also exported to the Private Activator, which forwards it to the Public Activator. The Public Activator connects to the external Certification Authority, which issues a certificate for the user's public key. The Certification Authority is provided by an external entity and is no integral part of the ServerBKU. The issued certificate is stored together with other data related to the newly activated virtual signature token.

As soon as these steps have been completed successfully, the newly activated virtual signature token can be used for subsequent signature-creation processes. Required functionality of the signature-creation process is mainly covered by the two web modules Public Signature Creator and Private Signature Creator. In the following, a typical signature-creation process is sketched in order to illustrate the functionality provided by these modules.

- **Send signature-creation request:** The process Signature Creation is initially triggered by a Service Provider, which requires the Signatory to create an electronic signature. For this purpose, the Public Signature Creator provides a well-defined public interface. Through this interface, Service Providers can send XML-based signature-creation requests via an HTTP Post messages. Thus, the architecture shown in Figure 7.3 is inaccurate to a certain extent. The Service Provider does not directly send signature-creation requests to the Public Signature Creator as shown in Figure 7.3. Instead, these requests are sent via the Web Browser by means of an HTTP Post request. As the Web Browser has an interface to the Service Provider anyways, this does however not impose additional issues from a functional point of view.

- **Request identification data:** As the signature-creation request is sent via the Signatory's Web Browser, the Public Signature Creator already has an established session with this local end-user component. Through this session, the Public Signature Creator identifies the Signatory prior to proceeding with the requested signature creation. To identify the Signatory, the Public Signature Creator requests him or her to enter the phone number associated with the desired virtual signature token. Together with the phone number, the Signatory is requested to enter the assigned secret password that has been defined during activation. For this purpose, the Public Signature Creator provides a simple web form that is displayed in the Signatory's Web Browser. This web form can be integrated by the Service Provider smoothly e.g. by means of an HTML Inline Frame (IFRAME) element. Data entered into the provided web form is sent to the Public Signature Creator via an HTTP Post message.

- **Verify identification data:** By means of these data, the Public Signature Creator identifies the Signatory and the desired virtual signature token. To proceed with the signature creation, the Private Signature Creator loads the encrypted and wrapped private key assigned to the identified signature token from the Inner Database. This action is triggered by the Public Signature Creator, which is the only component outside the Inner Core with direct access to the Private Signature Creator. The encrypted and wrapped key is decrypted by the Private Signature Creator with the help of a decryption key derived from the Signatory's secret password. This way, the correctness of the secret password entered by the user is verified implicitly. Only if the provided data is correct, the Signatory's key can be decrypted successfully. The decrypted but still wrapped key is loaded into the HSM, where it is unwrapped using an HSM-specific internal key.

- **Request authorization data:** If the Signatory's key can be decrypted successfully, the Public Signature Creator completes the authentication process. For this purpose, the Public Signature Creator generates a random TAN and sends this TAN via SMS to the mobile-phone number that is assigned to the selected virtual signature token. The external SMS Gateway and the Signatory's MNO are employed for that. The Public Signature Creator again provides a simple web form in the Web Browser, through which the Signatory can enter the received TAN. Together with this

web form, also the DTBS is displayed to the Signatory. The entered TAN is then sent to the Public Signature Creator by means of an HTTP Post request.

- **Verify authorization data:** The Public Signature Creator compares the entered TAN with the generated one. If this comparison is successful, the Private Signature Creator triggers the signature-creation process in the HSM.
- **Return signature-creation response:** The result of this process, i.e. the created signature value, and the Signatory's certificate are assembled to a well-defined XML-based signature-creation response. This response is finally returned to the Service Provider as answer to the initial signature-creation request.

The two processes Activation and Signature Creation cover most of the functionality provided by the ServerBKU. Even though it shows a higher degree of flexibility regarding different deployment scenarios, the ServerBKU basically resembles the Austrian Mobile Phone Signature. This especially applies to the implemented user-authentication mechanism, which relies on a simple SMS-TAN approach. This approach has been shown to be inappropriate for signature solutions that rely on one single mobile end-user device only. In the remainder of this chapter, we show how the ServerBKU can be improved by concepts and solutions proposed in this thesis. Concretely, the ServerBKU's user-authentication and authorization mechanism is replaced by an enhanced challenge-response approach. This way, the ServerBKU is further developed towards an implementation of the mobile signature solution Smartphone Signature proposed in Chapter 6.

7.2 Implementation Design

The ServerBKU represents a ready-to-use signature solution that enables users to activate an arbitrary number of virtual signature tokens and to use these tokens for the creation of electronic signatures. Due to its reliance on the SMS-TAN approach, the ServerBKU is however restricted to a use with two separate end-user devices. Thus, the ServerBKU is perfectly suitable to act as implementation basis for the Smartphone Signature, which has been proposed in Chapter 6. By enhancing the ServerBKU's user-authentications scheme with concepts of the Smartphone Signature, the ServerBKU can be made ready for a use on a single mobile end-user device. This way, the capabilities of the proposed signature solution Smartphone Signature can be evaluated in practice by means of a concrete implementation.

In the following, the architecture of this implementation is derived by combining the existing signature solution ServerBKU with the proposed solution Smartphone Signature. From the resulting architecture, process flows for the activation of new virtual signature tokens and for the creation of electronic signatures are derived. The two covered processes, i.e. activation of virtual signature tokens and creation of electronic signatures, are discussed in the following two subsections. As the proposed Smartphone Signature mainly focuses on the signature-creation process, this process is discussed first.

7.2.1 Signature Creation

Figure 7.4 shows the architecture of the presented implementation that combines the ServerBKU with the Smartphone Signature. This architecture identifies domains, entities, and components that are involved in a typical signature-creation process. For the sake of clarity, components related to and needed only for the activation process are not considered by the architecture shown in Figure 7.4. In general, the shown architecture resembles the architecture of the ServerBKU, which is illustrated in Figure 7.3. This is reasonable, as the presented implementation bases on the ServerBKU.

Although the architecture of the presented implementation resembles the architecture of the ServerBKU, several differences can be identified. These differences are caused by the integration of concepts

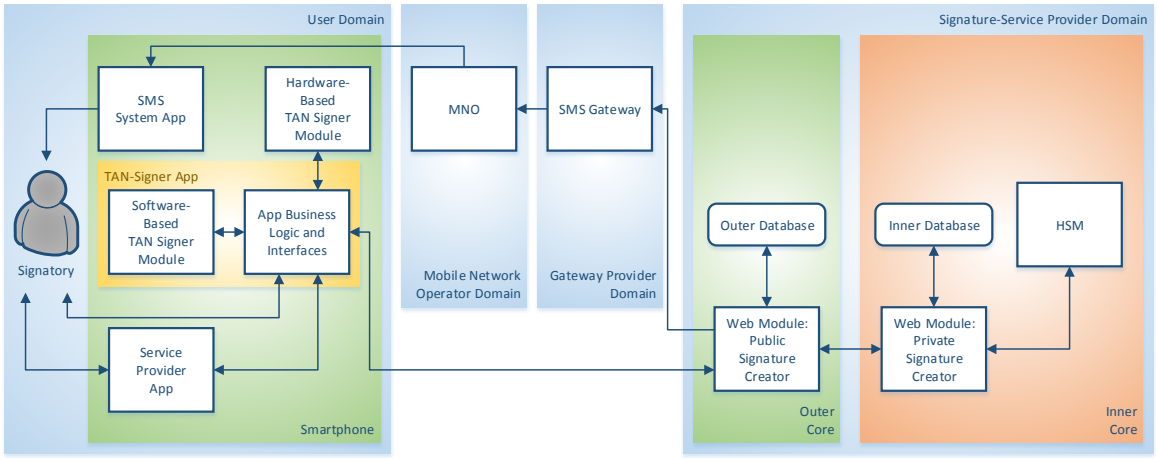

Figure 7.4: The signature creation related architecture is obtained by incorporating concepts of the Smartphone Signature into the architecture of the ServerBKU.

of the Smartphone Signature into the ServerBKU. While the ServerBKU defines six domains in total, the architecture shown in Figure 7.4 comprises four domains only. The Certification Authority Domain has been omitted, as this domain and its components are only required during the activation of new virtual signature tokens. Similarly, the Service Provider Domain has also been omitted by the architecture shown in Figure 7.4, because the Service Provider is moved to the User Domain. In general, the Smartphone Signature considers both a Local Service Provider implemented in the User Domain and a Remote Service Provider implemented in the Service-Provider Domain. For the presented implementation, the case of a Local Service Provider implemented on the user's mobile end-user device is considered initially. Realizing the Service Provider on the mobile end-user device is the more interesting use case, as this variant is not possible with the ServerBKU. Hence, this approach is followed by the presented implementation. Accordingly, the Service Provider is implemented on the mobile end-user device by means of a mobile app. Figure 7.4 shows that this mobile app is represented by the local component Service Provider App. As the Service Provider is implemented locally in the User Domain, the Service-Provider Domain is not needed and can be omitted.

Another basic difference between the ServerBKU's architecture and the architecture of the presented implementation concerns the components of the Signature-Service Provider Domain. As the architecture shown in Figure 7.4 focuses on the signature-creation process only, components that are not required for this process are omitted. Thus, the Signature-Service Provider Domain contains only those components that are directly involved in a signature-creation process. These are the Public Signature Creator, the Outer Database, the Private Signature Creator, the Inner Database, and the HSM. These components have been inherited from the architecture of the ServerBKU, as they already cover most of the required server-based functionality. Minor enhancements have only been applied to the components Public Signature Creator and Outer Database, in order to integrate required user-authentication functionality defined by the underlying concepts of the Smartphone Signature. Applied enhancements will be discussed in detail in Section 7.3.

The probably most significant differences between the architectures of the ServerBKU and the presented implementation concern the User Domain. According to the original architecture of the ServerBKU, the User Domain contains the Signatory and two separate end-user devices represented by the two components Web Browser and Mobile Phone. These components are sufficient to implement the ServerBKU's SMS TAN based user-authentication mechanism. As the presented implementation replaces this mechanism by the challenge response based authentication mechanism of the Smartphone Signature, components of the User Domain need to be adapted accordingly. As the Smartphone Signature has been

designed to be used on one single mobile end-user device, all required components of the User Domain are implemented in one component. In Figure 7.4, this component is denoted as Smartphone. To cover the functionality of the Smartphone Signature, the Smartphone implements the TAN-Signer App, which in turn comprises the App Business Logic, the Software-Based TAN Signer Module, and the Hardware-Based TAN Signer Module. Furthermore, the Smartphone implements the SMS System App and the Service Provider App, which assumes the role of a Local Service Provider.

The architecture shown in Figure 7.4 identifies relevant domains and components that are involved in a signature-creation process. From this architecture, the process flow of a typical signature-creation process can be derived. For the sake of clarity, the various processing steps that are required to complete such a process have not been illustrated in Figure 7.4. Instead, the entire process flow of a typical signature-creation process is shown in Figure 7.5 by means of a separate sequence diagram.

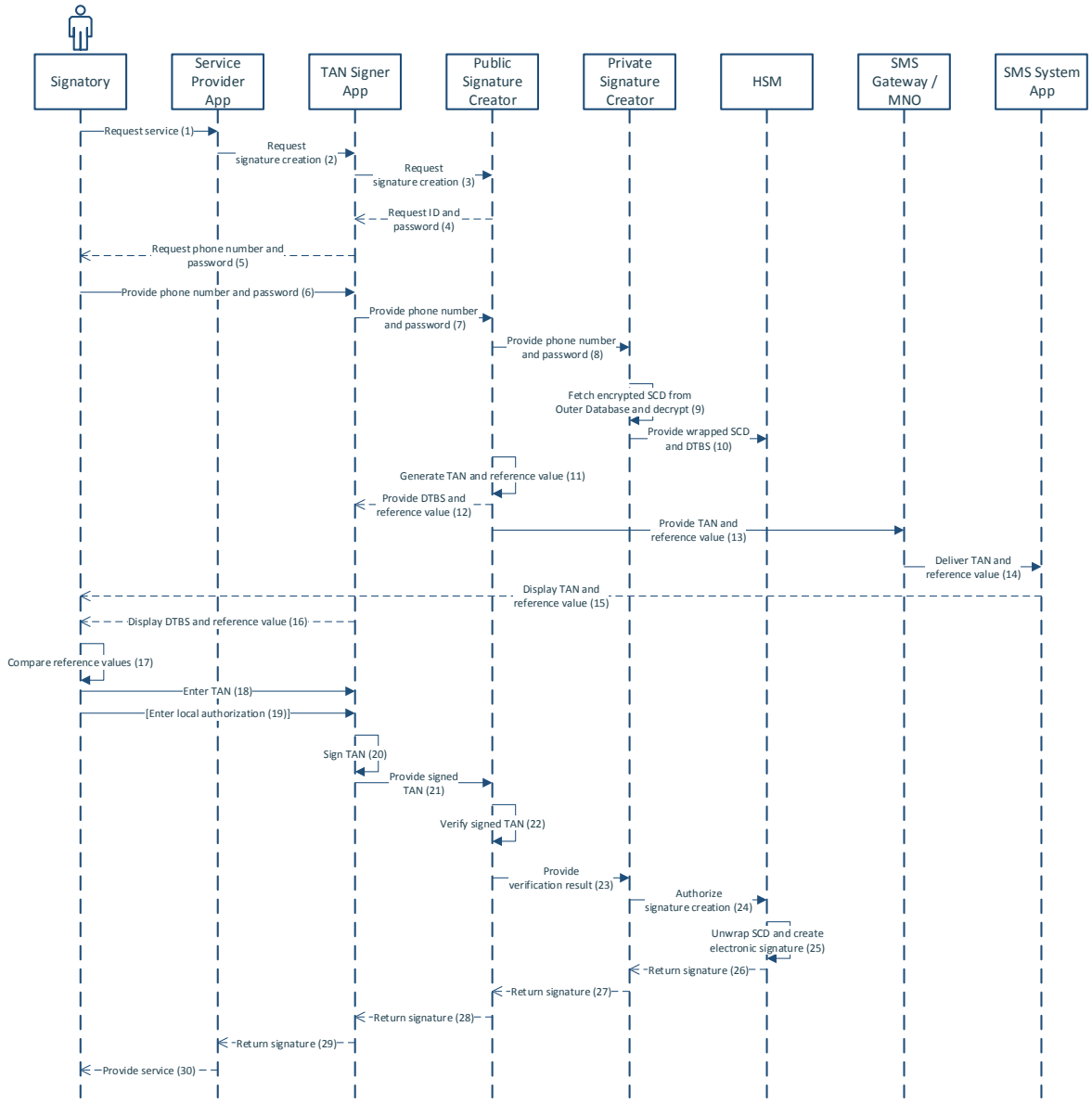

Figure 7.5: The creation of an electronic signature requires the interaction of various components.

The sequence diagram considers all relevant components identified by the architecture shown in

Figure 7.4. For the sake of clarity, several simplifications have been applied. First, the components MNO and SMS Gateway are represented by one entity in the sequence diagram. Second, the two databases of the Signature-Service Provider Domain are not modeled as separate entities in the sequence diagram. Finally, in the User Domain, subcomponents of the TAN-Signer App are not modeled separately by the sequence diagram and the Hardware-Based TAN-Signer Module is assumed to be part of the TAN-Signer App. Consideration of these simplifications finally yields eight entities that are modeled in the sequence diagram shown in Figure 7.5. According to this diagram, the following processing steps are required to complete a typical signature-creation process.

1. The Signatory requests a service from the Service Provider App.
2. To provide the requested service, the Service Provider App requires an electronic signature from the Signatory. To start the signature-creation process, the Service Provider App sends a signature-creation request to the TAN Signer App.
3. The TAN-Signer App forwards the received signature-creation request to the Public Signature Creator.
4. Prior to creating the requested signature, the Signatory needs to be authenticated. To start the authentication process, the Public Signature Creator requests the TAN-Signer App to provide the phone number and the password associated with the virtual signature token that shall be used for creating the signature.
5. The TAN-Signer App requests the Signatory to enter phone number and password.
6. The Signatory enters the phone number and password associated with the preferred signature token.
7. The TAN-Signer App forwards phone number and password to the Public Signature Creator.
8. The Public Signature Creator forwards these data to the Private Signature Creator.
9. The Private Signature Creator fetches the decrypted and wrapped signature key, i.e. SCD, of the virtual signature token that is referenced by the provided phone number and password from the Inner Database. It decrypts the fetched SCD using the provided password. Decryption is only successful, if the password is correct. This way, the password is verified implicitly.
10. The decrypted but still wrapped password is provided to the HSM.
11. To start the challenge response based user authentication, the Public Signature Creator generates a TAN and a reference value.
12. The Public Signature Creator sends the DTBS together with the reference value to the TAN-Signer App.
13. At the same time, the Public Signature Creator sends the generated TAN together with the reference value to the SMS Gateway.
14. With the help of the Signatory's MNO, the SMS Gateway delivers the TAN to the SMS System App on the Signatory's smartphone.
15. The SMS System App displays the TAN together with the reference value to the Signatory.
16. At the same time, the TAN-Signer App displays the DTBS together with the reference value to the Signatory.
17. The Signatory compares the two displayed reference values.

18. If the two reference values match, the Signatory enters the TAN to the TAN-Signer App.
19. Optional: If required by the employed TAN-Signer Module, the Signatory enters local authentication data such as a secret PIN.
20. The TAN-Signer App signs the entered TAN with the help of an available TAN-Signer Module. If required by the employed TAN-Signer Module, the TAN-signing functionality is enabled first with the help of the provided local authentication data.
21. The TAN-Signer App sends the signed TAN to the Public Signature Creator.
22. The Public Signature Creator verifies the signed TAN. For this purpose, it fetches the required public key from the Outer Database.
23. The Public Signature Creator provides the Private Signature Creator with the verification result.
24. If this result is positive, the Private Signature Creator authorizes the signature-creation operation in the HSM.
25. The HSM unwraps the still wrapped SCD and creates the electronic signature on behalf of the Signatory. Afterwards, the unwrapped key is discarded immediately.
26. The created signature is returned to the Private Signature Creator.
27. The Private Signature Creator forwards the created signature to the Public Signature Creator.
28. The Public Signature Creator returns the created signature to the TAN-Signer App.
29. The TAN-Signer App returns the signature to the Service Provider App.
30. The Service Provider App provides the service that has been requested in the first step by the Signatory.

This process flow illustrates the enhancements that have been achieved by integrating concepts of the Smartphone Signature into the ServerBKU. In general, two basic enhancements can be identified. First, by incorporating concepts of the Smartphone Signature, the SMS TAN based user-authentication mechanism is replaced by a more secure challenge-response approach. Second, reliance on concepts of the Smartphone Signature and hence reliance on a challenge response based user authentication enables the use of a Local Service Provider such as a mobile app. These two enhancements are achieved by means of an additional mobile app, which is implemented on the mobile end-user device. This app provides required client functionality of the challenge response based authentication scheme and acts as intermediary between the Local Service Provider and remote components of the server-based signature-creation solution. Due to its reliance on an additional mobile app, the presented implementation, i.e. the enhanced ServerBKU, is especially tailored to a use on modern mobile end-user devices such as smartphones.

7.2.2 Activation

Besides signature creation, activation represents the second basic process supported by the ServerBKU. This process has to be carried out prior to signature-creation processes. During activation, a virtual signature token is created that can later be used for an arbitrary number of signature-creation processes. When integrating concepts of the proposed Smartphone Signature into the ServerBKU, also the ServerBKU's activation process has to be adapted accordingly. In particular, the additional TAN-Signer App on the Signatory's mobile end-user device needs to be paired with the newly activated virtual signature token. During this paring process, the TAN-Signer App needs to generate an asymmetric key pair. The private part of this key pair is later used to sign TANs. The public part of the key pair must be remotely stored

together with the activated virtual signature token. This enables remote components of the server-based signature solution to verify signed TANs during subsequent signature-creation processes.

In order to achieve the required pairing between activated virtual signature tokens and local instances of the TAN-Signer App, the ServerBKU's activation process needs to be extended. Components that are required for this extended activation process are shown in Figure 7.6. This figure again illustrates the architecture of the presented implementation and has been derived from the architecture of the ServerBKU shown in Figure 7.3 on page 200 by incorporating the proposed solution Smartphone Signature. Thus, Figure 7.6 basically resembles Figure 7.4 on page 204, but focuses only on those components that are needed during an activation process.

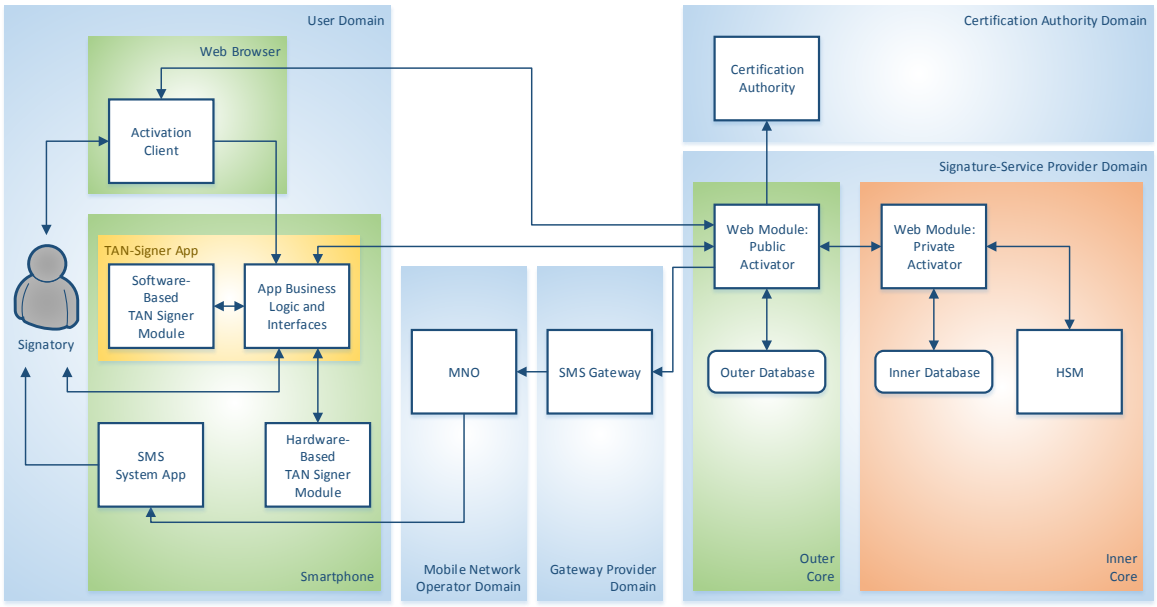

Figure 7.6: The activation-related architecture is obtained by incorporating concepts of the Smartphone Signature into the architecture of the ServerBKU.

Due to its focus on the activation process, Figure 7.6 differs in some aspects from Figure 7.4, which has mainly focused on the signature-creation process. First, the architecture shown in Figure 7.6 defines five domains in total. In addition to the four domains that have already been considered in Figure 7.4, Figure 7.6 also defines the Certification Authority Domain. This domain contains the Certification Authority, which is required during the activation process to issue the certificate for the Signatory. Second, in the Signature-Service Provider Domain, only those components are considered that are involved in the activation process. These are the Public Activator, the Private Activator, the Outer Database, the Inner Database, and the HSM. Components of the Signature-Service Provider Domain that have been defined by the architecture of the ServerBKU but are only required during the signature-creation process, are not considered in Figure 7.6. In particular, this applies to the Public Signature Creator and the Private Signature Creator. Finally, relevant differences can also be identified for the User Domain. In contrast to the signature creation related architecture shown in Figure 7.4, the activation-related architecture shown in Figure 7.6 defines two components, i.e. two end-user devices, for the User Domain. In addition to the Smartphone, which implements the TAN-Signer App and other required components, also the component Web Browser is defined. In the Web Browser, the component Activation Client is implemented.

From the architecture shown in Figure 7.6, the process flow of a typical signature-creation process can be derived. For the sake of clarity, the processing steps required to complete an activation process are again illustrated in a separate sequence diagram. This sequence diagram is shown in Figure 7.7 and Figure 7.8. Figure 7.7 recaps the typical activation process of the ServerBKU, in which a new virtual

signature token is activated. This activation process is not affected by the integration of concepts of the Smartphone Signature at all. In contrast, Figure 7.8 shows those processing steps that are required to establish the pairing between the newly activated virtual signature token and the local instance of the TAN-Signer App. Figure 7.7 and Figure 7.8 show that the entire activation process can be clearly separated into two parts. The first part is equal to the original activation process of the ServerBKU. The second part bases on the first one and is required to pair the TAN-Signer App with the newly activated virtual signature token. Both parts are described in the following by means of a typical activation process.

Figure 7.7 shows the processing steps of the activation process that is already known from the original ServerBKU. For this purpose, all relevant components of the underlying architecture are also modeled by the sequence diagram shown in Figure 7.7. For the sake of clarity, a few simplifications have been applied. For instance, the two databases of the Signature-Service Provider Domain are not modeled separately by the sequence diagram. Furthermore, the SMS Gateway and the MNO have been combined in one component in the sequence diagram. Finally, the Hardware-Based TAN-Signer Module, which is implemented by the local Smartphone component is assumed to be part of the TAN-Signer App and not modeled separately. It is important to note that the TAN-Signer App is not involved in the original activation process of the ServerBKU, which is shown in Figure 7.7. Still, this component is part of the illustrated sequence diagram for the sake of completeness. However, it has been greyed out in order to emphasize the fact that this component is not involved in the illustrated process flow.

According to Figure 7.7, the first part of the registration process comprises the processing steps described in the following. It is assumed that the Signatory has already completed the registration process and has successfully logged in to the user account that has been created during this process. Thus, the Signatory has an authenticated browser session with the ServerBKU.

1. The Signatory starts the activation process in the Activation Client, i.e. in a web-based front-end provided by the ServerBKU.
2. The Activation Client sends a request to start a new activation process to the remote Public Activator.
3. The Public Activator provides a web form to the Activation Client, through which the Signatory can specify and enter required data.
4. The Activation Client displays this web form to the Signatory.
5. The Signatory enters the required data. This includes the Signatory's phone number and a secret password associated with the new virtual signature token to be activated.
6. The Activation Client transmits the entered data to the Public Activator.
7. The Public Activator stores the received data in the Outer Database.
8. The Public Activator generates a random activation code.
9. The Public Activator sends this activation code to the SMS Gateway.
10. By means of the Signatory's MNO, the SMS Gateway delivers the activation code to the SMS System App on the Signatory's Smartphone via SMS.
11. The SMS System App displays the received activation code to the Signatory.
12. The Signatory enters the activation code to the Activation Client.
13. The Activation Client transmits the activation code to the Public Activator.
14. The Public Activator verifies the received activation code by comparing it with the generated one.

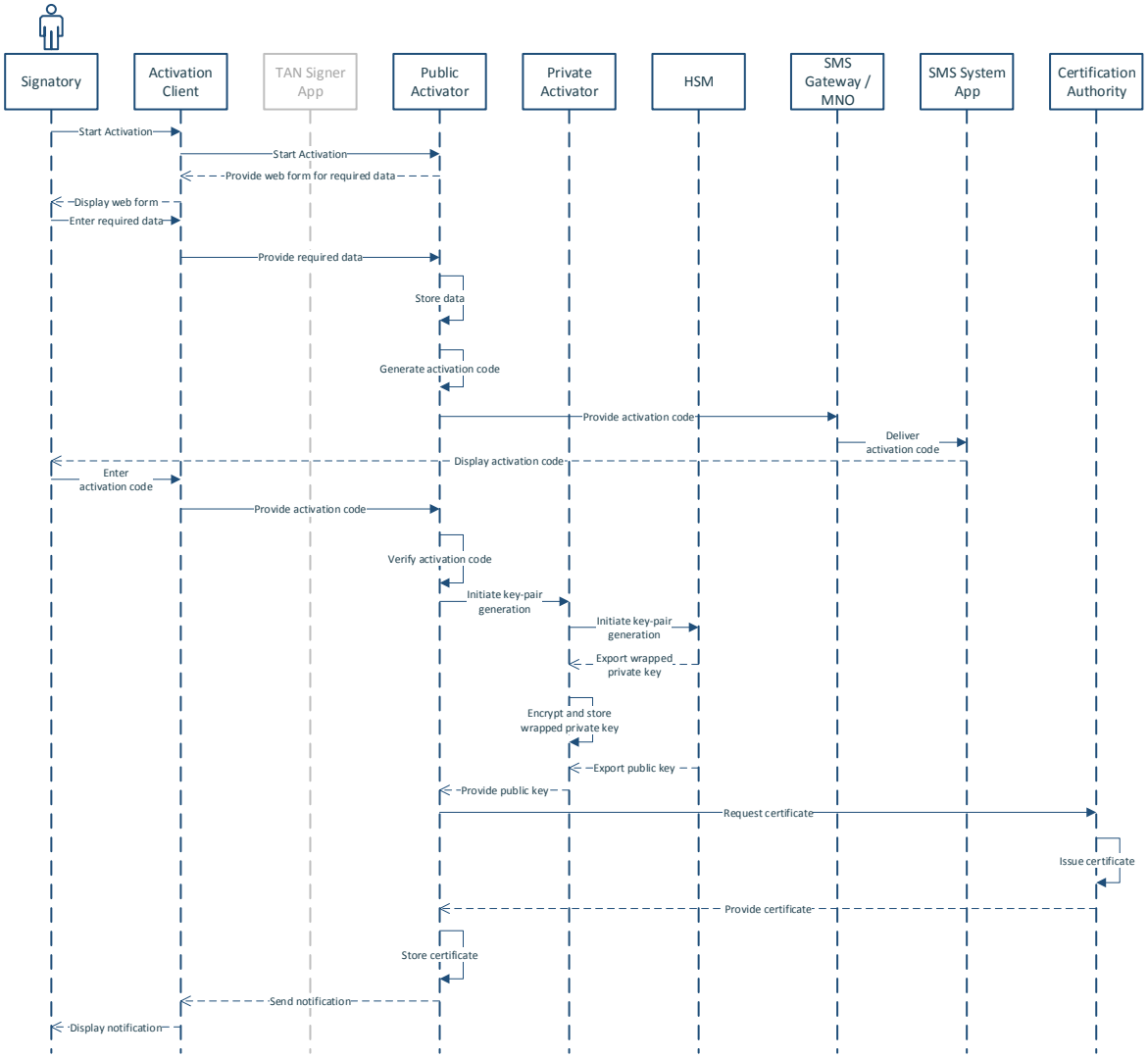

Figure 7.7: The first part of the activation process is equal to a typical activation process of the ServerBKU.

15. The Public Activator contacts the Private Activator to initiate the generation of a new key pair.
16. The Private Activator initiates the key-pair generation in the HSM.
17. The HSM generates the key pair.
18. The HSM wraps the private key of the generated key pair and exports it to the Private Activator.
19. The Private Activator encrypts the wrapped key and stores it in the Inner Database.
20. The HSM exports the public key of the generated key pair to the Private Activator.
21. The Private Activator provides the exported public key to the Public Activator.
22. The Public Activator requests a certificate for the public key from the Certification Authority.
23. The Certification Authority issues a certificate.

24. The Certification authority returns the issued certificate to the Public Activator.
25. The Public Activator stores the certificate in the Outer Database.
26. The Public Activator sends a notification to the Activation Client.
27. The Activation Client displays the notification to the Signatory.

After successful completion of these processing steps, a new virtual signature token has been created for the Signatory. According to the original ServerBKU implementation, the token can be used during signature-creation processes, whenever the Signatory has been successfully authenticated at the ServerBKU following the SMS-TAN approach. The presented implementation replaces the SMS TAN based authentication scheme by an improved challenge-response approach. As this approach requires an additional local mobile app, the activation process described above has to be extended in order to pair this app with the newly created virtual signature token.

The required pairing process is illustrated in Figure 7.8 and described in the following. It is assumed that a virtual signature token has already been successfully created, i.e. that the first part of the activation process illustrated in Figure 7.7 has been completed successfully. For the sake of clarity, the sequence diagram shown in Figure 7.8 models the same components as the sequence diagram that has been used to illustrate the first part of the activation process. Again, components not directly involved in the illustrated process are greyed out, but are still displayed for the sake of completeness.

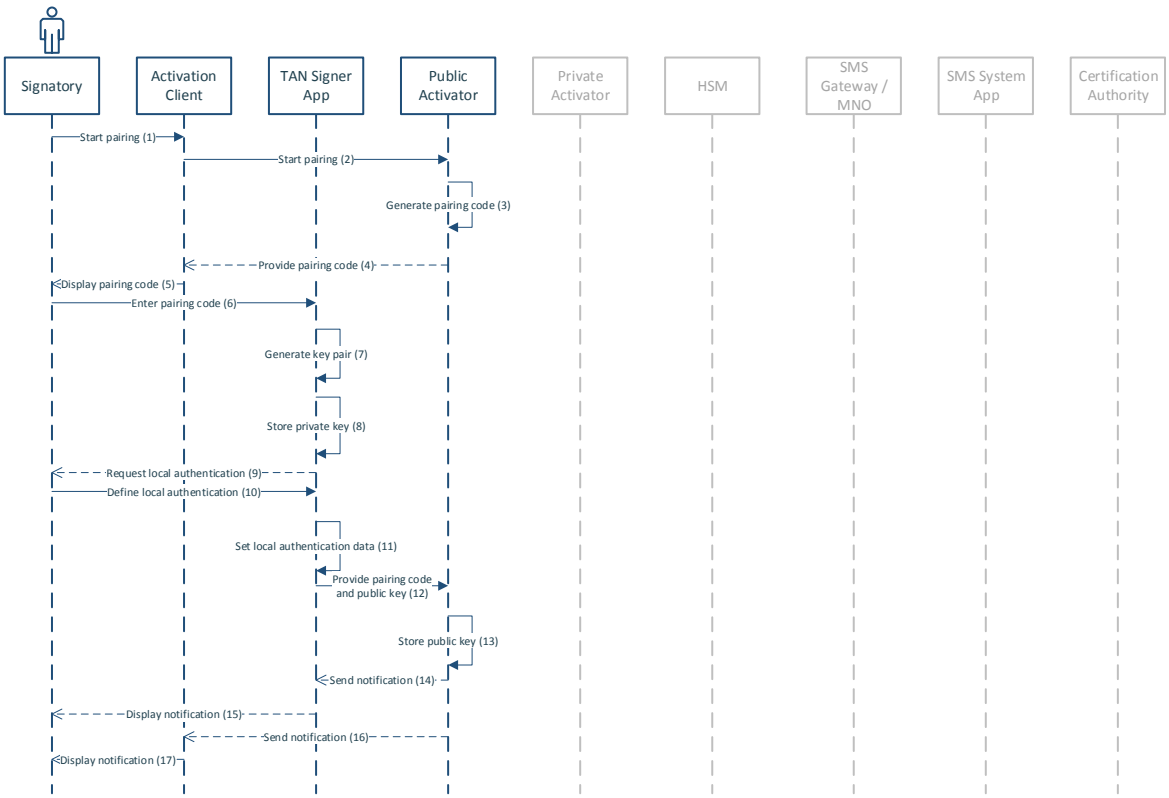

Figure 7.8: The second part of the activation process covers the pairing of the TAN-Signer App.

1. In the established authenticated browser session, i.e. in the Activation Client, the Signatory starts the pairing process for the newly created virtual signature token.

2. The Activation Client starts the pairing process in the Public Activator.
3. The Public Activator generates a pairing code.
4. The generated pairing code is sent to the Activation Client.
5. The Activation Client displays the pairing code to the Signatory.
6. The Signatory enters the activation code to the TAN-Signer App.
7. The TAN-Signer App generates a key pair for the signing of TANs.
8. The TAN-Signer App securely stores the private key of the generated key pair.
9. Optional: The TAN-Signer App can request the Signatory to define local authentication data such as a PIN that is used to protect the private key.
10. Optional: If requested to do so, the Signatory defines local authentication data in the TAN-Signer App.
11. Optional: If local authentication data have been defined, these data are set by the TAN-Signer App.
12. The TAN-Signer App sends the public key of the generated key pair together with the pairing code to the Public Activator.
13. The Public Activator identifies the relevant virtual signature token by means of the received pairing code and stores the provided public key for this token.
14. The Public Activator sends a notification to the TAN-Signer App.
15. The TAN-Signer App displays the notification to the Signatory.
16. The Public Activator sends a notification to the Activation Client.
17. The Activation Client displays the notification to the Signatory.

Upon successful completion of these processing steps, the activated virtual signature token is paired with the Signatory's TAN-Signer App and can be used for subsequent signature-creation processes as described and discussed in Section 7.2.1. The applicability of the architectures and process flows derived in this section have been evaluated by means of a concrete realization. Details of this realization are presented in the next section.

7.3 Realization

The implementation design presented and discussed in Section 7.2 has been realized using state-of-the-art technology. Essentially, this realization extends the ServerBKU such that it complies with the derived architecture and process flows and hence implements the concepts of the proposed Smartphone Signature. This is elaborated in this section in more detail.

7.3.1 Realization Details

In general, two basic tasks had to be accomplished to realize the implementation design presented in Section 7.2. First, a mobile app had to be implemented that assumes the role of the local component TAN-Signer App. Second, components of the existing signature solution ServerBKU had to be extended, in order to cover required server functionality. In particular, the server components Public Activator, Public Signature Creator, and Outer Database had to be extended.

Accomplishment of the latter task, i.e. extension of existing ServerBKU components, was rather simple, as only minor extensions were necessary. Essentially, the ServerBKU's SMS TAN based user-authentication scheme has been replaced by an improved challenge response based method that relies on signed TANs. For this purpose, two basic modifications of the ServerBKU had to be applied. First, the ServerBKU's Public Activator has been extended, in order to adapt the activation process such that key material required for the verification of signed TANs can be reliably exchanged with the Signatory's TAN-Signer App and stored in the Outer Database. Second, TAN-verification functionality of the Server-BKU's Public Signature Creator has been modified. Instead of simply comparing the received TAN with the one sent to the Signatory, the Public Signature Creator has been extended to cryptographically verify received signed TANs using the key material exchanged during the activation process. Both required changes have been implemented using existing libraries such as the IAIK JCE[7].

In addition to the extension of components of the ServerBKU, development of a mobile app that implements functionality of the TAN-Signer App represented the second mandatory realization task. As inclusion of a mobile app has not been part of the original ServerBKU concept, a suitable app had to be developed from scratch. Implementation details of the developed app are discussed in the following in more detail.

The required app has been developed for the Google Android[8] platform. This platform has been chosen mainly because of three reasons. First, Google Android is the current market leader for mobile operating systems [MobiThinking, 2014]. Hence, reliance on this platform assures that a high percentage of potential users can be reached. Second, own conducted assessments of relevant platforms have revealed that Google Android currently provides developers the highest degree of flexibility and functionality [Zefferer et al., 2013b]. This makes this platform perfectly suitable for the implementation and realization of innovative solutions. Third, from all platforms, Android is most affected by malware [Kelly, 2014]. Hence, a solution that satisfies security requirements on Android should also be able to satisfy the same requirements on any other platform.

Required functionality of the developed app has been identified from the implementation design, in particular from the derived activation and signature-creation processes. During activation, the TAN-Signer App needs to create a cryptographic key pair, store the private part of the key pair, request required data such as the pairing code from the Signatory, and forward the pairing code together with the public part of the key pair to the ServerBKU. During signature-creation, the TAN-Signer App acts as intermediary between a Local Service Provider and the ServerBKU, requests data such as the secure password and the TAN from the Signatory, signs the TAN, and forwards the signed TAN to the ServerBKU. Hence, provision of user interfaces, communication with the ServerBKU, communication with the Local Service Provider, generation and storage of cryptographic key material, and the signing of TANs are the basic features that need to be covered by the developed TAN-Signer App.

The required functionality of the TAN-Signer App also defines its internal structure, which is shown in Figure 7.9. Naturally, this structure resembles the architecture of the TAN-Signer App that has been defined in Chapter 6 as part of the Smartphone Signature. However, the internal structure shown in Figure 7.9 is more specific and also takes into account the decision to rely on the Android platform.

Figure 7.9 shows that the App Business Logic is the central building block of the TAN-Signer App

[7] https://jce.iaik.tugraz.at/
[8] http://www.android.com

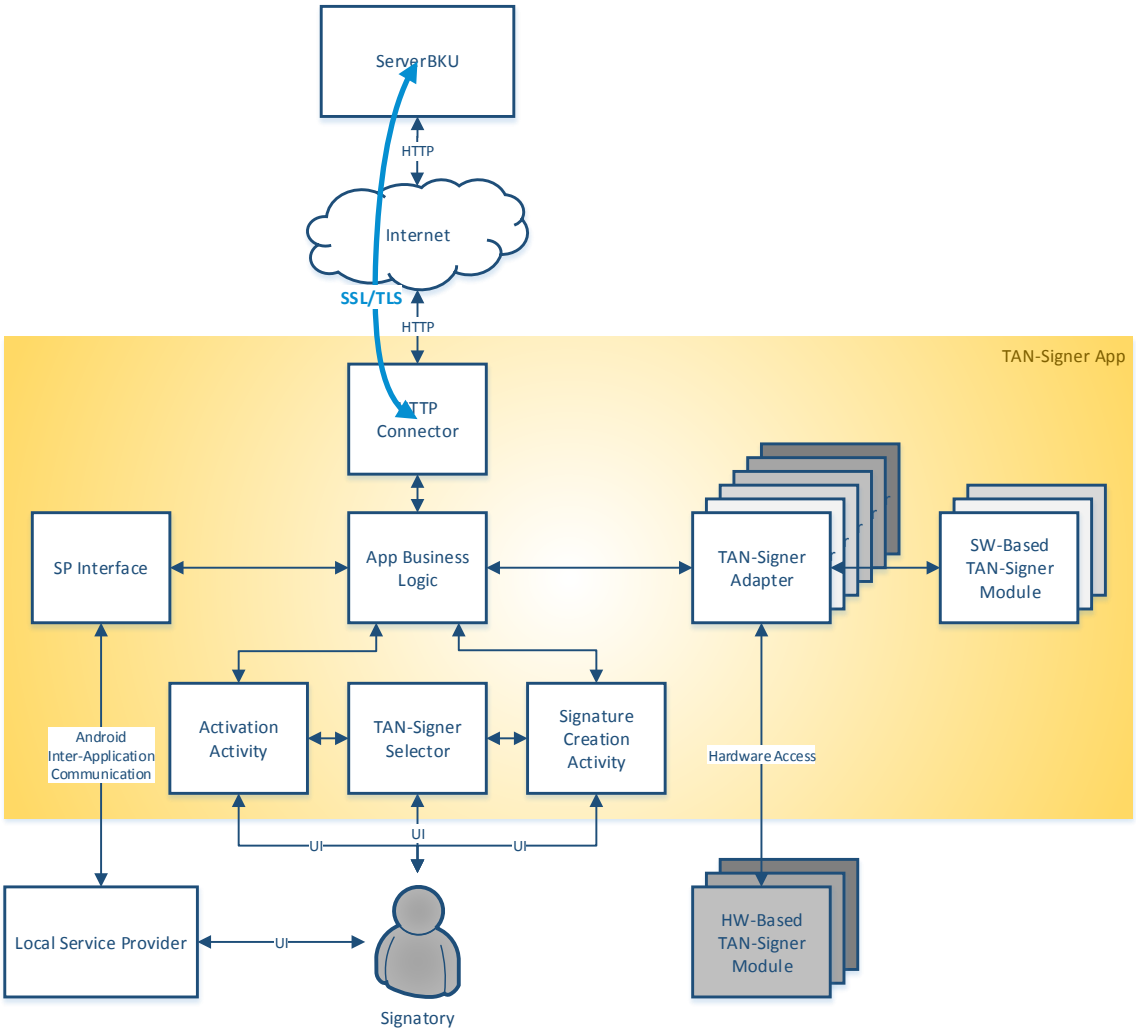

Figure 7.9: The TAN-Signer App is composed of several internal building blocks and interfaces.

implementation. This building block acts as central control unit and implements required process flows. For this purpose, the App Business Logic relies on and connects to other internal building blocks, which provide specific functionalities.

To interact with the Signatory, the App Business Logic makes use of the two interface components Activation Activity and Signature Creation Activity. The names of these building blocks reflect the Android-based nature of the TAN-Signer App. Under Android, user-interface components are implemented by so-called Activities[9], which interact with other app components by means of inter-process communication mechanisms. In addition to the two implemented user-interface building blocks, the TAN-Signer App also implements interfaces to the remote ServerBKU and to Local Service Providers. To access the ServerBKU through its HTTP-based interface, an HTTP Connector has been implemented for the TAN-Signer App. The HTTP Connector provides support for SSL/TLS in order to protect data exchanged with the ServerBKU. To interact with Local Service Providers, a simple SP Interface building block has been implemented. This building block enables a Local Service Provider to invoke the TAN-Signer App in order to initiate a signature-creation process and to retrieve created signatures.

In addition to interface building-blocks that enable communication with external entities, the TAN-

[9] http://developer.android.com/reference/android/app/Activity.html

Signer App also needs to implement cryptographic functionality to generate asymmetric key pairs during the activation process, and to sign TANs during subsequent signature-creation processes. The secure and reliable implementation of required cryptographic functionality on current mobile end-user devices is a challenging task. This has already been considered during development of the proposed signature solution Smartphone Signature in Chapter 6. In order to cope with varying threats and opportunities on different mobile end-user devices, the Smartphone Signature has left the concrete implementation of the TAN-Signer Module, which covers required cryptographic functionality, open. This generic approach has also been pursued by the concrete realization of the proposed solution. Concretely, the TAN-Signer App supports the use of different TAN-Signer Modules. This is also illustrated in Figure 7.9. Different hardware-based and software-based implementation variants of the TAN-Signer Module are connected to the App Business Logic by means of so-called TAN-Signer Adapters. These adapters assure that the actual implementation of the TAN-Signer Module is to a large extent transparent to the App Business Logic. This way, different implementations of the TAN-Signer Module can be easily integrated into the TAN-Signer App without the need to adapt other internal building blocks. In order to enable the Signatory to select the preferred TAN-Signer Module at runtime, the TAN-Signer App features an additional user-interface component called TAN-Signer Selector. This component is used by the Activation Activity and the Signature Creation Activity to determine the TAN-Signer Module to be used.

Flexibility with regard to the concrete realization of the TAN-Signer Module is a key feature of the TAN-Signer App. Maintaining a high degree of flexibility is important, in order to cope with varying capabilities and limitations of different mobile end-user devices. To demonstrate the flexibility of the developed TAN-Signer App, three variants of the TAN-Signer Module have been realized that all rely on a different enabling technology. These three realizations are introduced in the following three sections. Concretely, TAN-Signer Module implementations relying on SEs, cryptography-enabled NFC tokens, and the Android KeyChain are presented.

7.3.2 Realization Variant 1: Secure Elements

SEs are probably the most obvious alternative to implement a hardware-based TAN-Signer Module on a mobile end-user device. Capabilities, limitations, and realization opportunities of SEs have already been briefly sketched in Chapter 4. There, SEs have been considered as potential enabler for signature solutions that rely on a local SSCD. Essentially, SEs are especially protected hardware modules that support the secure and reliable storage of security-critical data and the secure and reliable processing of cryptographic operations. Due to their hardware-based nature, SEs are to a large extent immune to malware. From a functional point of view, SEs are hence an ideal choice for implementing the local TAN-Signer Module. This section introduces the technology behind SEs, motivates their use, and shows how they can be employed to cover required TAN-signing functionality in the scope of the implemented Smartphone Signature.

7.3.2.1 Technology

In theory, SEs are perfectly suited to implement the functionality of a hardware-based TAN-Signer Module. Unfortunately, the availability of SEs on current mobile end-user devices is limited in practice. Even though most devices are supplied with modules that could assume the role of an SE, these modules often cannot be accessed by third-party apps that are not deeply rooted in the mobile operating system. So far, Google Android is the only platform that provides third-party apps basic access to SEs by means of the seek-for-android framework[10]. Capabilities of this framework have been analyzed by Oberauer [2012]. This analysis has revealed that access to hardware-based SEs is feasible under Android.

Despite their limited applicability on other platforms than Google Android, SEs have still been employed to implement the TAN-Signer Module. There are several reasons for that. First, there is evidence

[10] https://code.google.com/p/seek-for-android/

that SEs will continue to play a major role on mobile end-user devices. By now, SEs are integral parts of most smartphones and e.g. build the backbone of secure hardware-based credential stores provided by the mobile operating system. So far, access to these SEs is in most cases limited to the mobile operating system. However, it can be expected that future mobile end-user devices will provide further possibilities to implement SEs and that upcoming releases of mobile operating systems will provide access to SEs also for third-party apps. Second, the emergence of new security-critical applications such as mobile payment solutions will continuously raise the need for SEs on mobile end-user devices. For instance, the mobile payment system Google Wallet[11] has relied on secure elements on mobile end-user devices early. Another example for a mobile payment system that relies on a local SE is Softcard[12]. Similar to SIM-based signature solutions, Softcard makes use of an enhanced SIM, which assumes the role of the SE. These examples show that SEs can be expected to play a relevant role on mobile end-user devices in future, even though their applicability is currently still limited.

Finally, SEs additionally provide an interesting conceptual advantage that justifies their use. Despite their hardware-based nature, the functionality of SEs is dynamically adaptable. For this purpose, SEs usually rely on Java Card technology[13]. Using this technology, the functionality of an SE is defined by so-called Java Card applets. These are reduced Java-based programs that can be installed and executed in the SE. Installed Java Card applets have access to hardware-based cryptography features of the SE and can be accessed from external applications by means of the PC/SC protocol [PC/SC Workgroup, 2014]. Typically, multiple Java Card applets can be installed on one SE. As they define the functionality of the SE, secure means to install and remove Java Card applets are crucial. Hence, SEs are usually shipped with a secret cryptographic key that is securely stored in the SE. With the help of this key, a secure communication channel can be established between the SE and an authorized external entity that is also aware of this key. Over this secure communication channel, Java Card applets can be securely installed on SEs in the field. The possibility to securely install arbitrary Java Card applets on SEs offers an interesting opportunity. Applets to be installed can be equipped with a secret symmetric master key prior to their installation on the SE. As the installation of the applet takes place over a secure communication channel, this master key cannot be compromised during the installation process. Once the applet is installed on the SE, the secret master key is protected by the SE itself. This master key can subsequently be used to protect the communication between the Java Card applet and an external application. This way, all data exchanged with the SE can be reliably protected. As SEs provide means to establish a secure channel to external entities and hence enable secure messaging, they represent an interesting and promising technology for a highly secure realization of required TAN-signing functionality on mobile end-user devices.

7.3.2.2 Technology-Specific Internal Structure of the TAN-Signer App

Taking into account the decisions to rely on SEs and on their capabilities to establish secure communication channels with external entities, the internal structure of the TAN-Signer App shown in Figure 7.9 can be further refined. The resulting SE-specific internal structure is shown in Figure 7.10. For realization of the proposed solution, a Secure Element Emulator has been developed, which emulates the functionality of an SE. The Secure Element Emulator implements required cryptographic functionality and provides a PC/SC-based interface. In addition to the Secure Element Emulator, a Secure Element Adapter has been implemented, which acts as intermediary between the SE and the App Business Logic. The Secure Element Adapter basically hides implementation specifics of the SE and provides the App Business Logic a common interface to SE implementations. This way, the Secure Element Emulator can be easily replaced by arbitrary hardware-based SEs.

Figure 7.9 also shows that there is a secure channel between the SE and the ServerBKU. Hence,

[11] https://wallet.google.com
[12] https://www.gosoftcard.com
[13] http://www.oracle.com/technetwork/java/embedded/javacard/overview/index.html

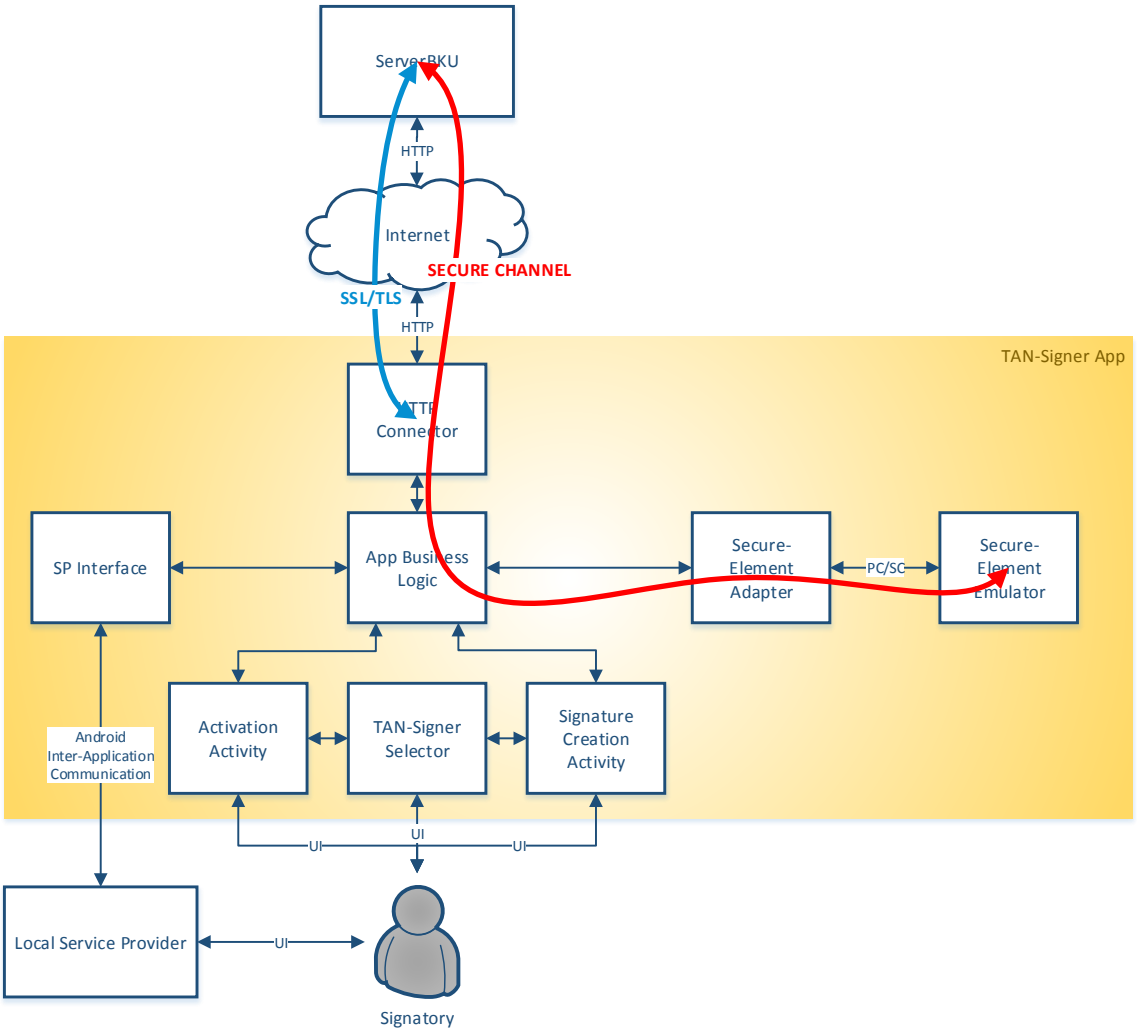

Figure 7.10: The architecture of the SE-based TAN-Signer App shows relevant internal building blocks and interfaces.

even if the mobile end-user device is compromised by malware, data exchanged between the SE and the ServerBKU remain secure and protected. This even holds true, if components of the TAN-Signer App are compromised, as exchanged data is encrypted and decrypted already in the SE.

Reliance on a secure channel between the SE and the ServerBKU, and the application of secure-messaging mechanisms also affects implemented activation and signature-creation processes. The integration of secure messaging into these processes and their SE-specific implementation is presented and discussed in the following.

7.3.2.3 Realization of the Activation Process

Following the implementation design derived in Section 7.2, the activation process consists of two parts. The first part reflects the activation process of the original ServerBKU solution: the Signatory creates a new virtual signature token through a web-based user interface provided by the ServerBKU. In the second part, the newly created virtual signature token is paired with the Signatory's personal instance of the TAN-Signer App, in order to enable the creation and verification of signed TANs during signature-creation processes.

As the first part of the activation process does not require interaction with the TAN-Signer App, required interaction with the Signatory mainly takes place over a web browser. This is illustrated by the following screenshots. Figure 7.11 illustrates the web-based user interface provided by the ServerBKU. This user interface gives the Signatory the opportunity to create new virtual signature tokens and to manage existing ones. Creation, i.e. activation, of a new token can be initiated by clicking the respective icon.

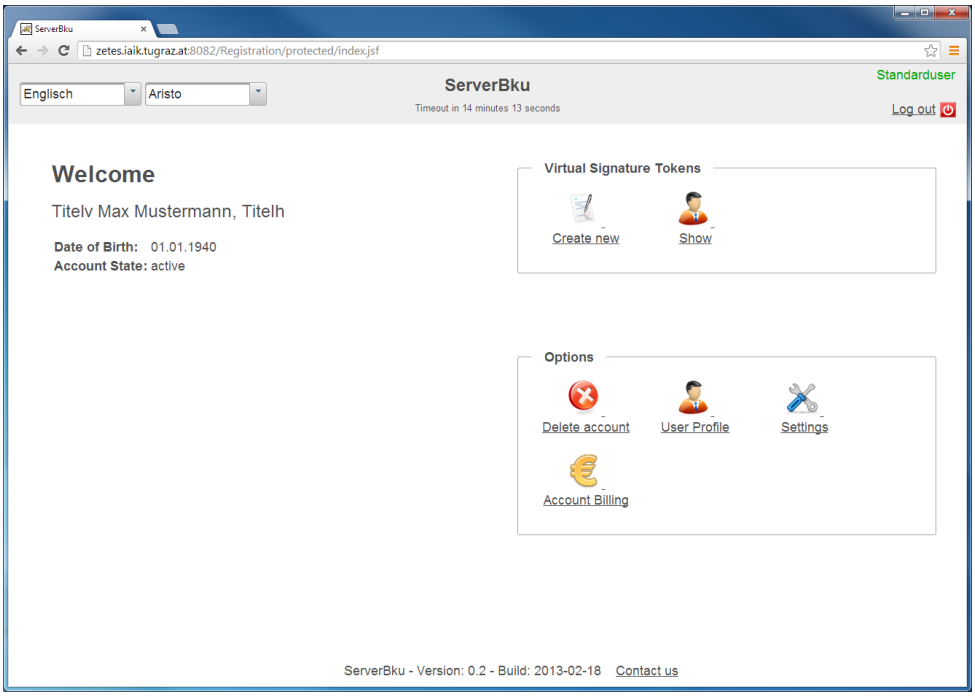

Figure 7.11: The web-based user interface provided by the ServerBKU provides means to create new virtual signature tokens and to manage existing ones.

Upon initiation of the activation process, an activation wizard is displayed, which guides the Signatory through the entire activation process. This wizard requests required data from the Signatory and displays relevant information. The first form of this wizard is shown in Figure 7.12. Through this form, the Signatory provides required data. This includes a unique identifier for the virtual signature token to be created, as well as the Signatory's phone number and e-mail address. Furthermore, the Signatory is required to define a secret password that will later be used to protect access to and use of the created token.

All entered data are transferred to the central ServerBKU. There, the phone number provided by the Signatory is verified. For this purpose, a random code is sent to the provided phone number by SMS. The Signatory has to enter this code into the activation wizard. This is illustrated in Figure 7.13.

By entering the received code, the Signatory proves possession and control of his or her mobile phone. In the next step, the Signatory is asked to enter further person-related and certificate-related data. This is illustrated in Figure 7.14 and Figure 7.15.

As soon as the ServerBKU has received the entered data, it creates a new cryptographic key pair for the Signatory in the central HSM. The private key of this key pair is wrapped and encrypted using the chosen token-specific password. The wrapped and encrypted password is finally stored in the Inner Database. The public key is sent to the Certification Authority together with certificate-related information provided by the Signatory. The Certification Authority issues a certificate in order to bind the public key to the Signatory's identity. The issued certificate is stored by the ServerBKU as part of the created virtual signature token. As all activation steps described so far have already been part of the existing ServerBKU implementation, they are not discussed here in more detail. Details of the original

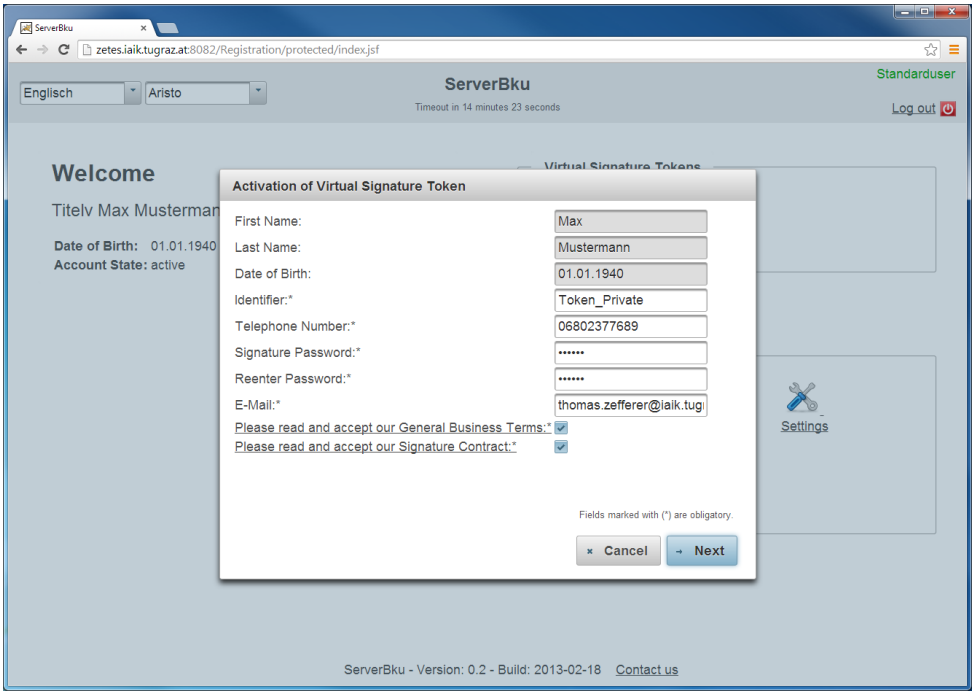

Figure 7.12: In the first step of the activation process, the Signatory is asked to enter required data.

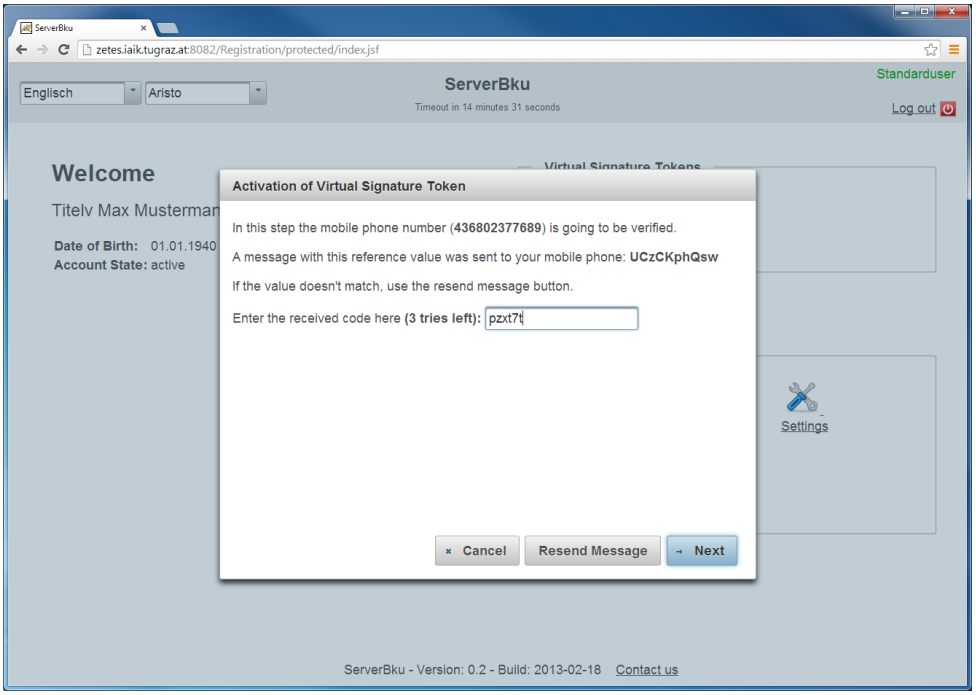

Figure 7.13: To verify the provided phone number, the Signatory is asked to enter a code delivered by SMS.

ServerBKU implementation have been presented and discussed by Rath et al. [2014a].

The second part of the activation process is actually the more interesting one. During this part, the Signatory's instance of the TAN-Signer App is paired with the newly created virtual signature token. During this pairing process, the TAN-Signer App, concretely the TAN-Signer Module, generates a new key pair that is subsequently used to sign TANs. The public key of this key pair is transmitted to the ServerBKU, in order to enable a remote verification of signed TANs during signature-creation processes.

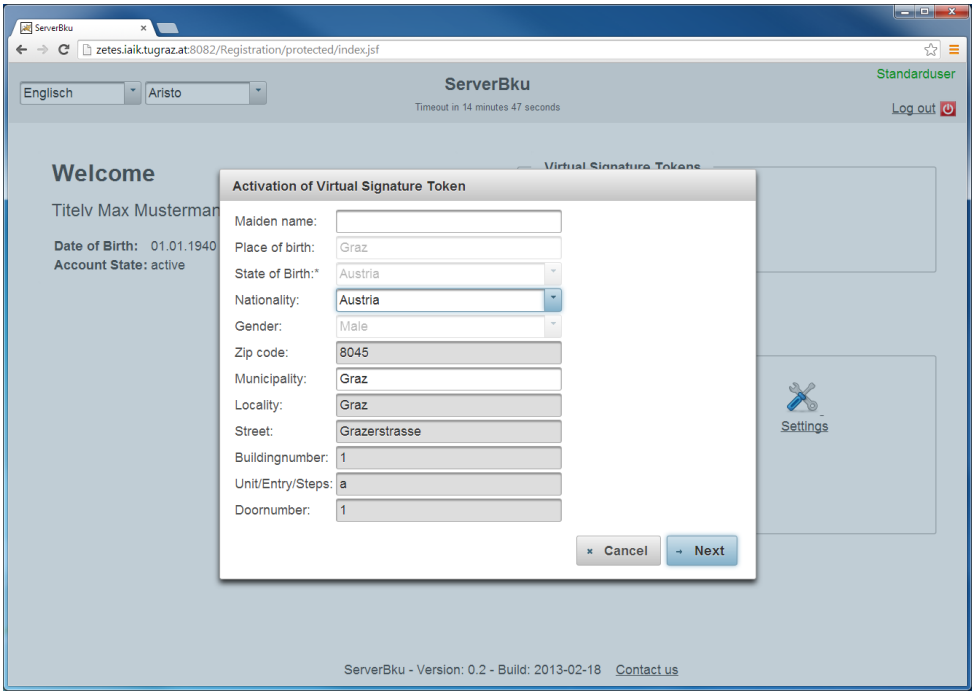

Figure 7.14: The TAN-Signer App prompts the Signatory to enter required person-related data.

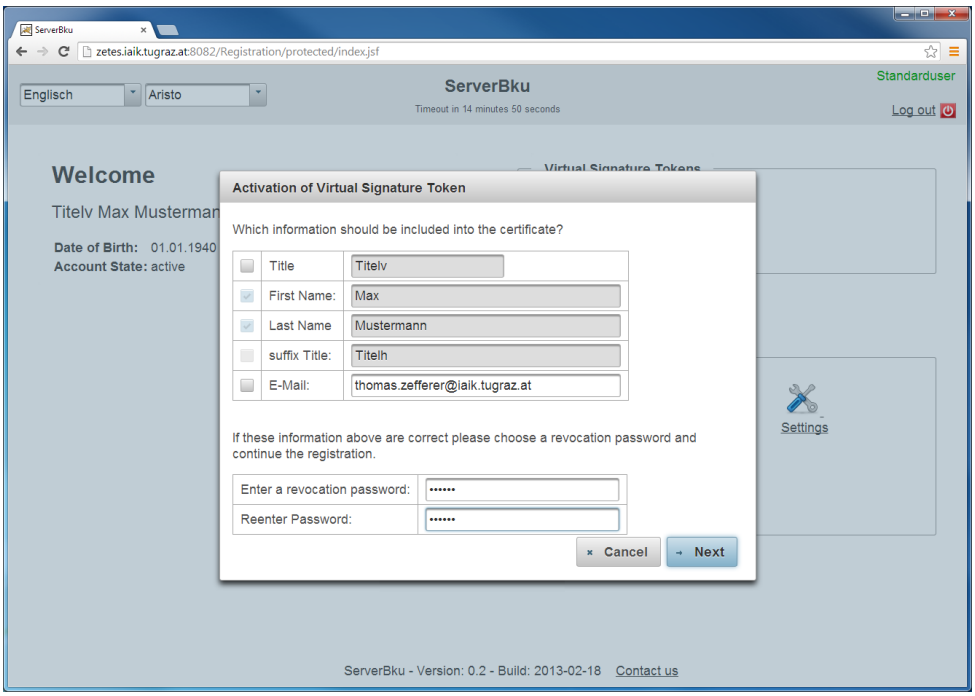

Figure 7.15: The Signatory is also asked to enter required certificate-related data.

To implement the pairing process, the ServerBKU's web-based activation wizard has been extended by an additional form. This form is displayed immediately after successful completion of the first part of the activation process, i.e. as soon as a new virtual signature token has been created. The displayed form is illustrated in Figure 7.16. Essentially, it displays a random activation code. By entering this code in the TAN-Signer App, an unambiguous link between the current browser session and the TAN-Signer App can be established. The displayed activation code is also provided in the form of a QR code, which can be scanned with mobile end-user devices featuring a camera and a QR scanner.

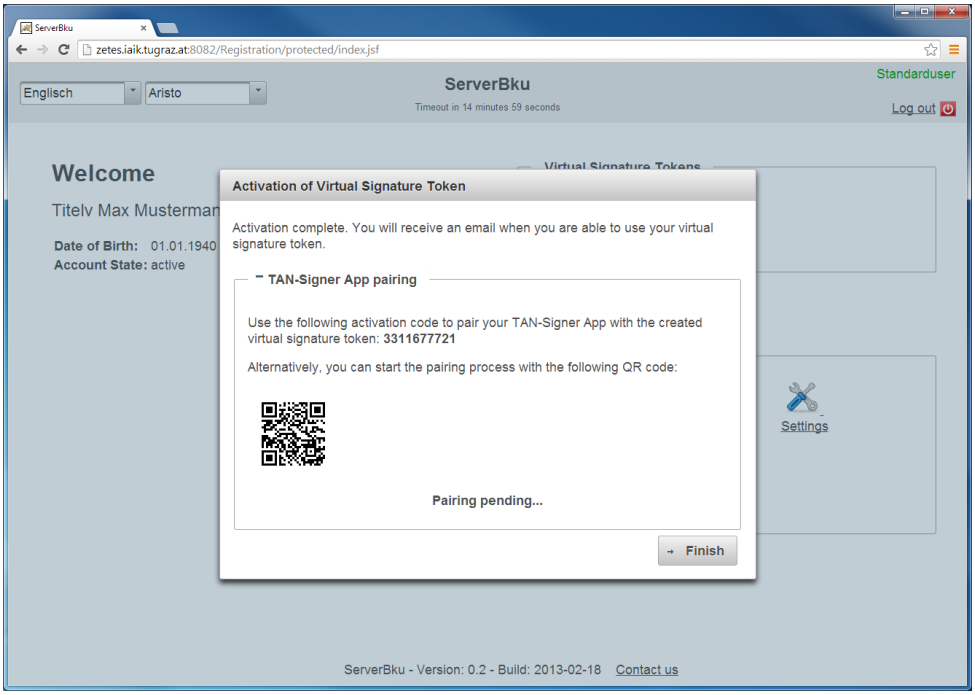

Figure 7.16: The pairing form is displayed immediately after successful creation of a new virtual signature token.

In order to initiate the app-pairing process, the Signatory needs to start the TAN-Signer App. This can be done manually by touching the app icon, or automatically by scanning the QR code displayed in the web browser. Upon initiation of the paring process, the TAN-Signer App first requests the Signatory to choose the preferred TAN-Signer Module, i.e. the technology that will later be used to sign received TANs. For this purpose, the TAN-Signer App displays the dialog shown in Figure 7.17.

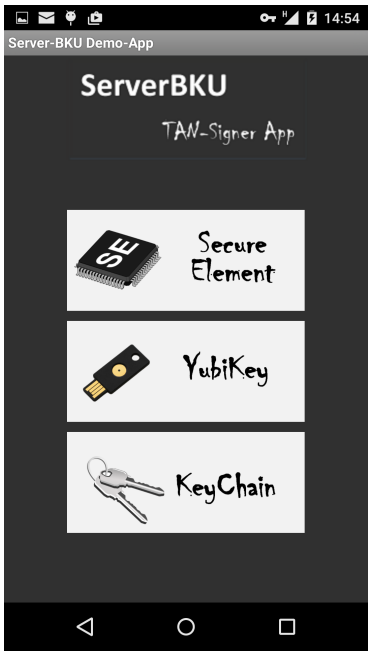

Figure 7.17: The TAN-Signer App provides a simple user interface to select the preferred technology used to sign TANs.

Assuming that the Signatory selects SE technology to sign TANs during authentication processes, the

TAN-Signer App continues the app-pairing process with the dialog shown in Figure 7.18. This dialog requires the Signatory to enter the phone number and the activation code. If supported by the mobile operating system, the phone number is automatically read out from the device. Otherwise, the phone number has to be entered manually by the Signatory. The app also supports the Signatory in entering the activation code. In case the TAN-Signer App has been started by means of the displayed QR code, the activation code is automatically pre-populated. Only if the app has been started manually, the Signatory needs to type in the activation code manually.

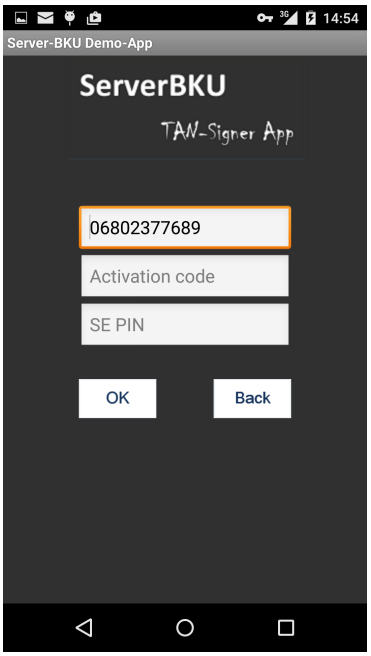

Figure 7.18: The TAN-Signer App provides a simple user interface to enter data required for the paring process.

In addition to the phone number and the activation code, the TAN-Signer App also requires the Signatory to define a secret SE PIN. This SE PIN is used to protect the private key stored in the SE and used to sign TANs during signature-creation processes. To increase the security of the PIN-entering process, a custom PIN pad has been implemented for the TAN-Signer App. This PIN pad, which is shown in Figure 7.19, circumvents the use of the mobile device's keyboard. This way, it counters potential threats caused by compromised soft keyboards.

When all required data have been entered to the TAN-Signer App, the pairing process can be started by pressing the OK button. The various communication steps of the pairing process, in which data is mainly exchanged between the remote ServerBKU and the local SE, are completely transparent to the Signatory. During execution of these steps, a progress bar is displayed and the Signatory is requested to stand by. This is illustrated in Figure 7.20. Due to the establishment of a secure channel between ServerBKU and local SE, which requires the execution of several cryptographic operations, the activation process can take a couple of seconds.

Figure 7.21 offers a more detailed look at the pairing process and shows in detail all communication steps that take place between the ServerBKU and the TAN-Signer App including the employed SE. Thus, Figure 7.21 essentially further details processing steps 6 – 15 of the second part of the pairing process that has been discussed in Section 7.2.2 and illustrated in Figure 7.8 on page 211. For the sake of clarity, internal components of the TAN-Signer App are not modeled separately in Figure 7.21. Only the SE is shown as separate component.

Figure 7.21 shows that after the Signatory has entered the activation code and other required data to the TAN-Signer App, a secure channel is established between the SE and the ServerBKU first. Estab-

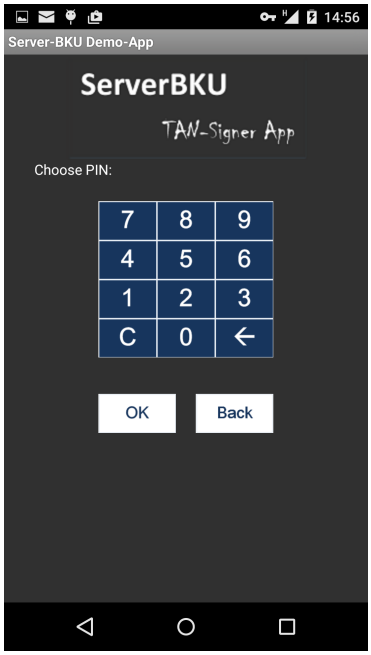

Figure 7.19: A custom PIN pad has been implemented to counter threats caused by compromised keyboards.

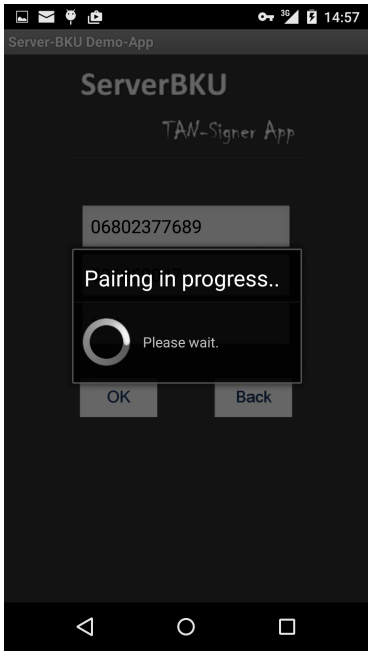

Figure 7.20: The paring process is completely transparent to the Signatory.

lishment of this secure channel is based on GlobalPlatform specifications[14]. Concretely, the GlobalPlatform's Secure Channel Protocol 03 [GlobalPlatform, 2009] is employed. Both, the TAN-Signer App and the ServerBKU have been extended to support this protocol. To establish a secure communication channel, the ServerBKU generates a random challenge and transmits this challenge to the TAN-Signer App. The TAN-Signer App forwards the challenge to the SE. Upon reception of the host challenge, the SE generates a card challenge. From the received host challenge, the generated card challenge, and the secret symmetric master key that has been installed in the SE as part of the JavaCard applet, the SE

[14] http://www.globalplatform.org/specifications.asp

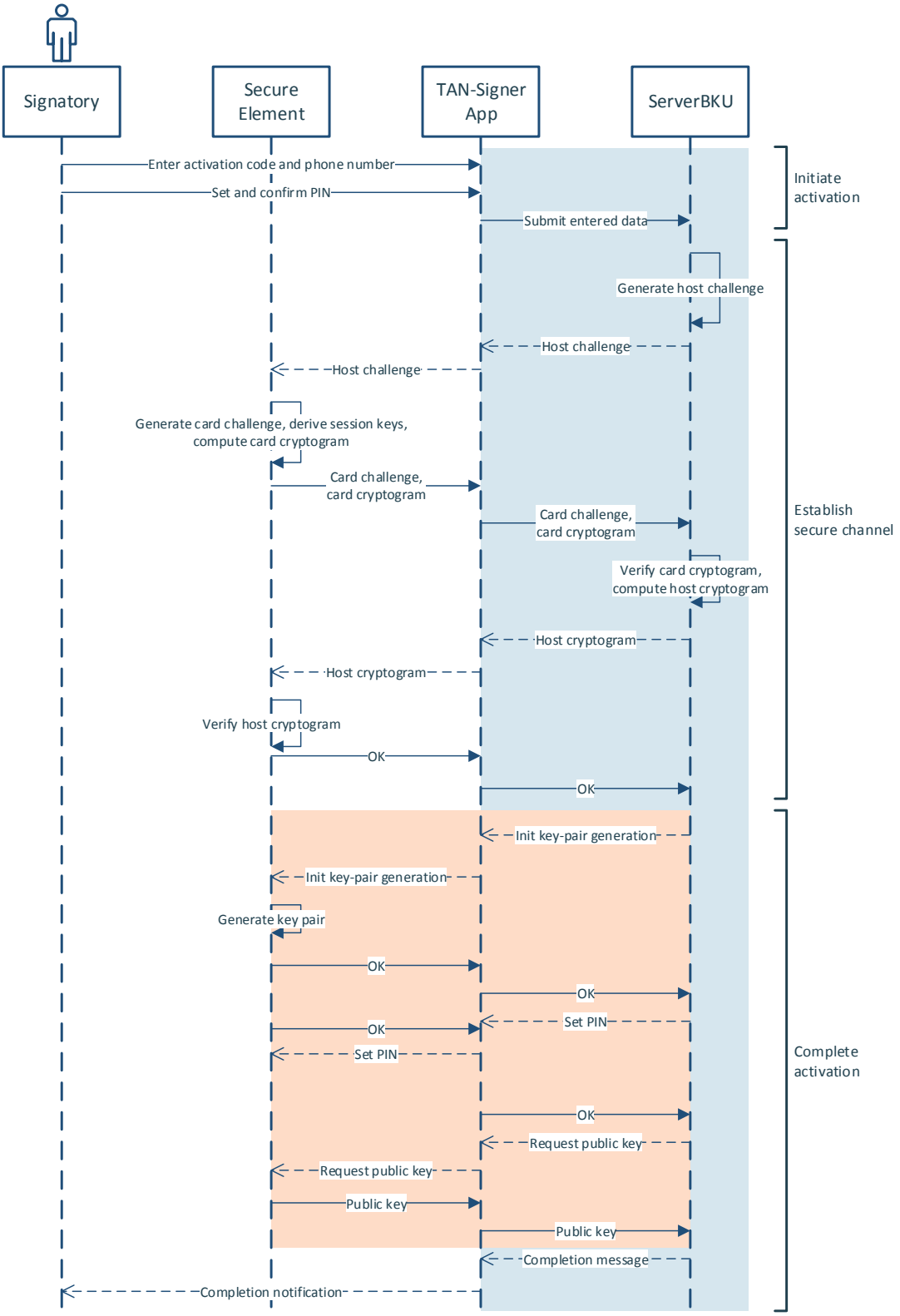

Figure 7.21: Data transmissions between local components and the ServerBKU are secured by means of SSL/TLS and secure messaging.

derives session keys. These session keys are later used to protect data exchanged between the SE and the ServerBKU. In addition, the SE computes a card cryptogram from the host challenge, the card challenge, and one of the derived session keys. The computed card cryptogram is sent to the TAN-Signer App and further to the ServerBKU together with the generated card challenge. From the provided data, the ServerBKU is able to derive the same session keys as the SE, as it is aware of the same secret cryptographic master key. Using the derived session keys, the ServerBKU can cryptographically verify the received card cryptogram and compute a host cryptogram from the host challenge and the card challenge. This host cryptogram is then sent to the TAN-Signer App, which forwards it to the SE. Being aware of the required session keys, the SE is able to verify the host cryptogram. If verification succeeds, the SE sends an OK message to the ServerBKU.

The final OK message completes the establishment of the secure channel between the ServerBKU and the SE. All subsequent communication steps between the ServerBKU and the SE can be cryptographically protected by means of the derived session keys. As derivation of these keys is based on a secret master key, external entities, which are not aware of this key, cannot decrypt exchanged messages. Note that not even the TAN-Signer App or any of its internal components is able to decrypt exchanged messages, as it is neither aware of the secret master key, nor of the derived session keys. Hence, when communication between the SE and the ServerBKU takes place over the established secure channel, the TAN-Signer App merely acts as proxy and communication relay for encrypted messages. Once the secure channel has been successfully established, the remaining steps of the activation process are executed. First, the ServerBKU sends a command to the SE, in order to initiate the generation of a new key pair. The SE generates the key pair and returns an OK message to the ServerBKU. Then, the ServerBKU transmits the PIN defined by the Signatory to the SE. The SE sets the PIN to protect the private key of the generated key pair. Subsequently, the SE again returns an OK message to the ServerBKU. Finally, the generated public key is requested from the ServerBKU and returned by the SE. All these communication steps take place over the established secure channel. Hence, confidentiality and integrity of exchanged data are protected at any time during the activation process.

Protection of data exchanged during the activation process is crucial. This especially applies to the transmission of the locally created public key to the ServerBKU. If an attacker is able to intercept this key and to replace it with an own key, signed TANs can be successfully forged during signature-creation processes. Hence, the secure and reliable transmission of the public key from the SE to the ServerBKU is crucial for the security of the entire system. Applied means to protect data exchanged between the ServerBKU and the local components TAN-Signer App and Secure Element are also illustrated in Figure 7.21. In general, communication between the TAN-Signer App and the remote ServerBKU is protected by means of SSL/TLS. All processing steps that are protected by these means are shown in front of a blue background. As an additional layer of protection, part of the communication between the ServerBKU and the Secure Element is protected by secure messaging. Communication steps affected by these means of protection are shown in front of a red background.

The successful execution of all processing steps illustrated in Figure 7.21 completes the pairing process. When the ServerBKU has received the Signatory's public key, it sends a completion message to the TAN-Signer App. This enables the TAN-Signer App to notify the Signatory about the successful completion of the pairing process. Additionally, successful completion of the pairing process is also indicated by the web-based activation wizard. This is shown in Figure 7.22.

Upon successful completion of the pairing process, activation of the new virtual signature token in the remote ServerBKU is complete. The created token can be used during subsequent signature-creation processes. In the following, the realization of such signature-creation processes is presented.

7.3.2.4 Realization of the Signature-Creation Process

Compared to the activation process, the signature-creation process is rather simple. It requires fewer processing steps and also fewer interactions with the Signatory. Furthermore, no additional web-based

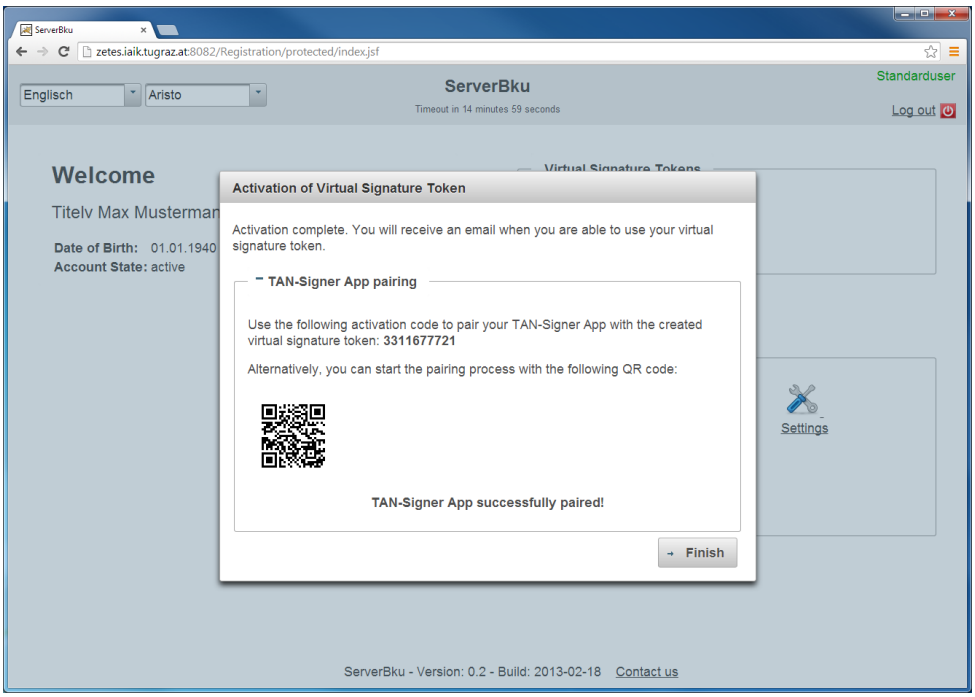

Figure 7.22: The pairing form notifies the Signatory about the successful completion of the pairing process.

user interface is required during signature creation, which again improves simplicity and efficiency. As signature-creation processes are usually applied more frequently than activation processes, they must be efficient and fast.

In contrast to the activation process, secure messaging is not employed during signature creation. During signature creation, the only data being exchanged between the SE and the ServerBKU is the signed TAN. The signed TAN is unique for each transaction. Hence, its confidentiality does not necessarily need to be assured, as a specific signed TAN cannot be reused for other transactions. Thus, transmission of the signed TAN over a secure channel would not improve the overall security of the user-authentication process. Furthermore, a basic level of security is still provided, as the communication channel between the TAN-Signer App and the ServerBKU is protected by means of SSL/TLS. Other critical data such as phone number and password associated with the chosen virtual signature token could indeed benefit from a transmission over a secure channel. However, these data are entered by the Signatory and are hence present in the TAN-Signer App anyways. Furthermore, a sufficient level of security is provided for these data, as they are transmitted over communication channels protected by SSL/TLS. Taking these considerations into account, it is reasonable to limit the application of secure messaging to the activation process and to not use it during signature creation.

As no additional user interface is required during signature creation, the number of required interactions with the Signatory is rather limited. The following screenshots show a typical signature-creation process from the Signatory's perspective and illustrate realized user interfaces.

At the beginning of the signature-creation process, the Signatory needs to be identified. For this purpose, the Signatory enters the phone number and password associated with his or her virtual signature token. These data have been defined during the activation process, in which the token has been created. As the Signatory can activate an arbitrary number of virtual signature tokens, this processing step is also required to give the Signatory the opportunity to select the preferred token. The user interface of the TAN-Signer App that enables the Signatory to specify phone number and password is shown in Figure 7.23.

The entered credentials are transmitted to the ServerBKU. There, the virtual signature token refer-

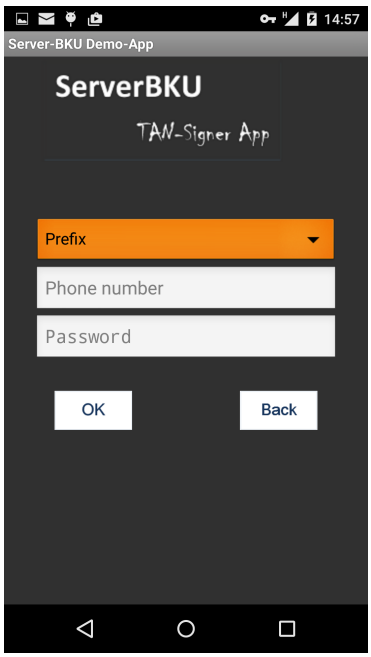

Figure 7.23: The TAN-Signer App requests the Signatory to enter phone number and password associated with the preferred virtual signature token.

enced by the provided data is fetched from the database and decrypted using the Signatory's provided password. This way, the password is implicitly verified. The decrypted but still wrapped password is loaded into the HSM, in order to prepare the signature-creation process. Before this process is authorized, the second step of the user-authentication process is carried out. Therefore, a random TAN is sent to the Signatory's mobile end-user device. The Signatory is requested to enter the received TAN into the TAN-Signer App and to sign it. Hence, the Signatory first needs to choose the respective TAN-Signer Module. For this purpose, the dialog shown in Figure 7.24 is displayed.

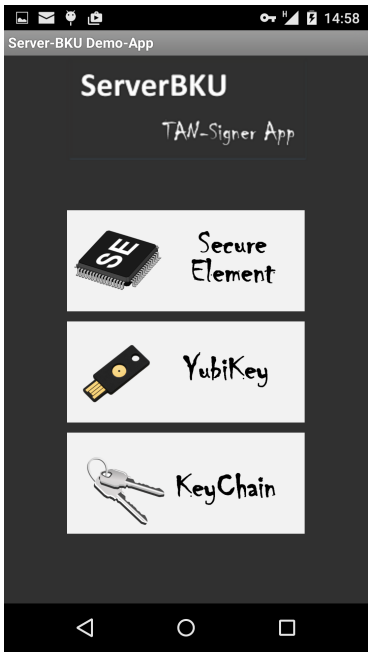

Figure 7.24: The TAN-Signer App requests the Signatory to select the respective TAN-Signer Module.

In order to be able to sign the received TAN using the selected TAN-Signer Module, the TAN-Signer App needs to read the TAN from the Signatory. For this purpose, the app displays the user interface shown in Figure 7.25.

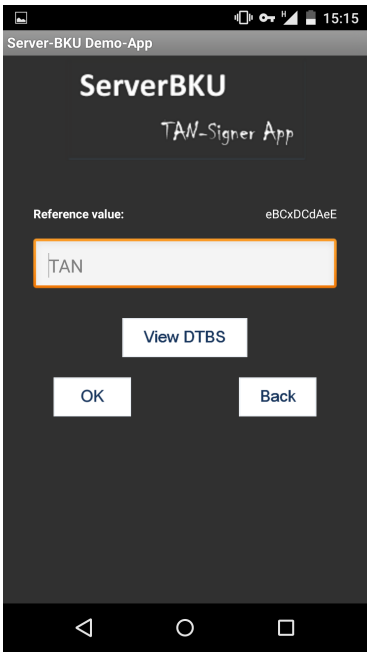

Figure 7.25: The TAN-Signer App requests the Signatory to enter the received TAN.

In addition to reading the TAN from the Signatory, the provided user interface provides two additional features. First, it displays a reference value, which is unique for the current authentication process. The same reference value is also part of the SMS message that is used to transmit the TAN to the Signatory. This way, the Signatory can establish an unambiguous binding between the received challenge, i.e. the TAN, and the current transaction. Second, the provided user interface enables the Signatory to review the DTBS. By pressing the respective button, the Signatory can open a simple DTBS viewer, which displays the data to be signed. This is illustrated in Figure 7.26.

The TAN entered by the Signatory is finally signed by the TAN-Signer App using the selected TAN-Signer Module, i.e. the SE. As the private key stored in the SE and used to sign TANs is protected by a secret PIN, the Signatory needs to enter this PIN in order to authorize the signing of the TAN. The PIN can again be entered by means of a custom PIN pad, which is provided by the TAN-Signer App as shown in Figure 7.27. This way, the confidentiality of the PIN is guaranteed, even if the keyboard of the mobile end-user device has been compromised.

The signed TAN is then transmitted to the ServerBKU. There, it is verified with the help of the public key that has been received during the activation process. If the signed TAN can be verified successfully, the signature-creation process is authorized in the central HSM and the requested electronic signature is finally created on behalf of the Signatory.

7.3.3 Realization Variant 2: Cryptography-Enabled NFC Tokens

The use of SEs shows that required functionality of the TAN-Signer App can indeed be realized with the help of state-of-the-art technology. Representing secure hardware, SEs are well-suited for the realization of the app's TAN-Signer Module. While the use of SEs is clearly beneficial in terms of security and usability, this technology unfortunately suffers from a rather poor feasibility, as current off-the-shelf end-user devices do not feature SEs that can be accessed from third-party apps. In order to provide an alternative solution for these devices, the implemented TAN-Signer App also supports cryptography-

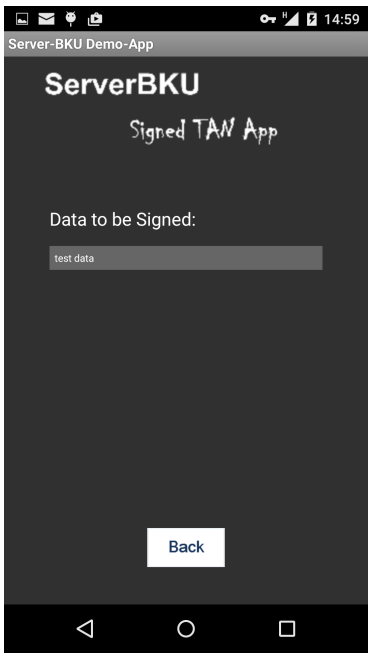

Figure 7.26: The TAN-Signer App features a simple DTBS viewer.

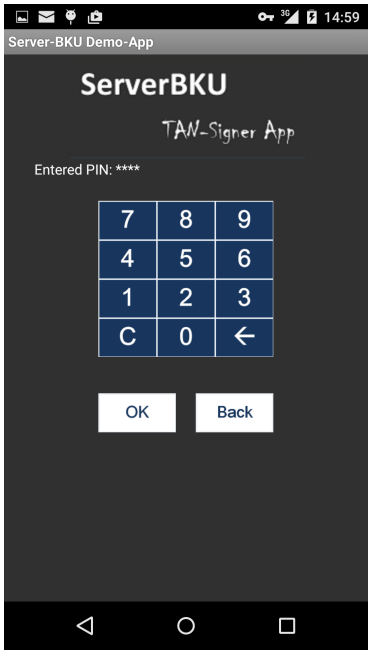

Figure 7.27: The required PIN can be entered by means of a custom PIN pad.

enabled NFC tokens as TAN-Signer Modules. The use of these tokens in the context of the developed solution is presented and discussed in this section.

7.3.3.1 Technology

NFC bases on Radio-Frequency Identification (RFID), a technology, which dates back to pioneering work by Stockman [1948], and which makes use of electromagnetic fields to establish a communication channel between an active reader an a passive tag or transponder. The passive transponder has no own power source and collects all required energy from the electromagnetic field generated by the reader.

The simple nature of RFID tags facilitates their cheap mass production. Accordingly, RFID technology has been used early for the identification of products and has evolved as an attractive alternative to bar codes. Well-known standards of RFID technology are MIFARE specified in ISO/IEC [2008] or FeliCa developed by Sony [Sony, 2015]. Key properties of RFID-based communication are low bandwidth, short range, and immediate communication set-up.

NFC bases on RFID technology and combines the established RFID standards MIFARE and FeliCa. Relevant standards defining NFC technology are ISO/IEC 18092 [ISO/IEC, 2013] and ISO/IEC 21481 [ISO/IEC, 2012]. In contrast to RFID, NFC does not strictly divide communicating devices into active readers and passive tags. Instead, each NFC device can be operated in three different modes. First, each NFC device can be operated in card-emulation mode and behave as a passive tag. Second, an NFC device can also be operated in reader/writer mode and act as active reader. Third, two NFC-enabled devices can directly communicate with each other when being operated in peer-to-peer mode. This way, NFC is applicable in more application scenarios and use cases compared to simple RFID solutions. Still, NFC features the same communication properties such as short range, low bandwidth, and immediate communication set-up.

Although NFC has been available for more than ten years, adoption of this technology has been low for a long time. This has changed recently, when NFC has been identified as promising enabler for contactless payment systems, such as those analyzed in Zefferer [2013]. This promising use case has motivated device manufactures to integrate NFC technology into smartphones and related mobile end-user devices. Today, NFC technology is supported by most major platforms including Android and iOS [NFC World, 2015].

With the emergence of more complex use cases, passive NFC tags have become more complex and powerful as well. For instance, contactless payment systems and other security-critical application scenarios require passive NFC tags to carry out cryptographic operations. This has yielded a new generation of powerful NFC-enabled security tokens. Contactless smart cards are an example for such tokens, which are frequently used by contactless payment systems to authenticate card holders and to authorize financial transactions at dedicated payment terminals. In addition to contactless smart cards, cryptography-enabled NFC tokens can be obtained in other form factors as well. For instance, the company Yubico[15] offers NFC-enabled Universal Serial Bus (USB) tokens under the name YubiKey [Yubico, 2015]. Similar to SEs, these tokens support JavaCard technology and can be programmed with own JavaCard applets to define their functionality.

The support of NFC technology by modern mobile end-user devices and the availability of powerful cryptography-enabled NFC tokens make these tokens an interesting alternative to classical SEs. In contrast to SEs, cryptography-enabled NFC tokens have a considerable advantage: the Signatory has to actively enable access to the token by putting it in the short range of the mobile device's NFC reader. Similarly, the Signatory can easily prevent a possible misuse of the token simply by removing it from the reader. This adds an additional level of protection for scenarios, in which the mobile end-user device is compromised e.g. by local malware. We have demonstrated capabilities of cryptography-enabled NFC tokens to act as TAN-Signer Module by integrating support for YubiKey tokens into the developed TAN-Signer App. Details of this integration are provided in more detail in the following sections.

7.3.3.2 Technology-Specific Internal Structure of the TAN-Signer App

Based on the decision to rely on cryptography-enabled NFC tokens, more specifically on YubiKeys, to implement the TAN-Signer Module, the TAN-Signer App's general architecture defined in Figure 7.9 on page 214 can be further refined. The resulting technology-specific architecture that is tailored to the use of YubiKeys is shown in Figure 7.28.

In many aspects, this architecture resembles the one of the SE-based realization variant shown in

[15] https://www.yubico.com/

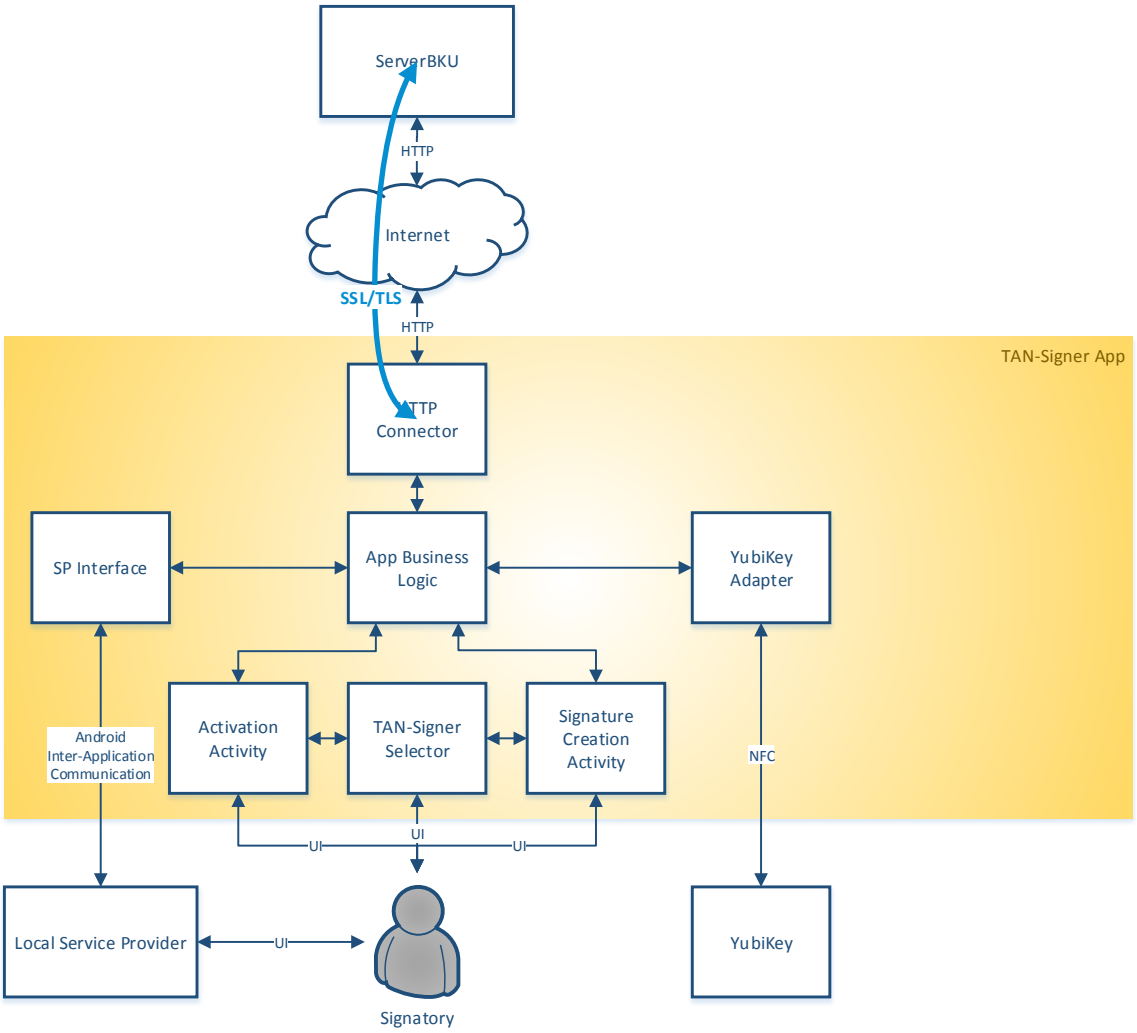

Figure 7.28: The architecture of the YubiKey-based TAN-Signer App shows relevant internal building blocks and interfaces.

Figure 7.10 on page 217. However, two basic differences can be identified. The first difference is due to the alternative technology that is used to realize the TAN-Signer Module. Instead of an SE, a YubiKey is used for this purpose. Accordingly, the TAN-Signer App's Secure Element Adapter has been replaced by a YubiKey Adapter. The added YubiKey Adapter implements all NFC-based communication to the external NFC token and hence provides the App Business Logic access to the YubiKey.

The second basic difference to the SE-specific architecture concerns the establishment of a secure channel between the ServerBKU and the TAN-Signer Module, i.e. the YubiKey. The YubiKey-based implementation sketched in Figure 7.28 does not implement such a secure channel. There are several reasons for that. First and foremost, avoidance of a secure-channel establishment facilitates the deployment of YubiKeys. By omitting secure messaging with the YubiKey, required cryptographic functionality that has to be implemented by the YubiKey is reduced to the generation of a cryptographic key pair and the creation of electronic signatures. These features are often already covered by JavaCard applets pre-installed on shipped tokens. This way, off-the-shelf tokens can be deployed and used without the need to personalize them with proprietary JavaCard applets. Furthermore, the intentional avoidance of secure messaging demonstrates the flexibility of the entire solution, which supports secure messaging with the TAN-Signer Module, but does not define it as mandatory requirement.

The architecture shown in Figure 7.28 explicitly defines the use of a YubiKey as TAN-Signer Module. However, other NFC tokens can be used as well, as long as they support the required cryptographic functionality. YubiKeys have been used for the present realization mainly because of their current popularity and flexibility. The following two sections describe the integration of YubiKeys into the implemented activation and signature-creation processes.

7.3.3.3 Realization of the Activation Process

Replacing the SE with a YubiKey has only minor impacts on the overall activation process. Again, this process consists of two parts. In the first part, the Signatory creates a new virtual signature token through the web-based user interface provided by the ServerBKU. In the second part, the newly created virtual signature token is paired with the Signatory's TAN-Signer App. The first part of the activation process is always identical, irrespective of the concrete realization of the TAN-Signer Module. Hence, when relying on a YubiKey as TAN-Signer Module, the Signatory has to go through the same processing steps as described in Section 7.3.2.3 and illustrated in Figure 7.11 to Figure 7.16. After successful completion of these processing steps, the Signatory has created a new virtual signature token and the ServerBKU displays an activation code and a QR code in the Signatory's web browser.

For the subsequent second part of the activation process, in which the Signatory's TAN-Signer App is paired with the newly created virtual signature token, several YubiKey-specific differences can be identified compared to the SE-specific activation process. The pairing process can again be started either manually by starting the TAN-Signer App on the mobile end-user device, or by scanning the displayed QR code. In either case, the TAN-Signer App first requests the Signatory to select the preferred technology for the TAN-Signer Module. This is illustrated in Figure 7.29.

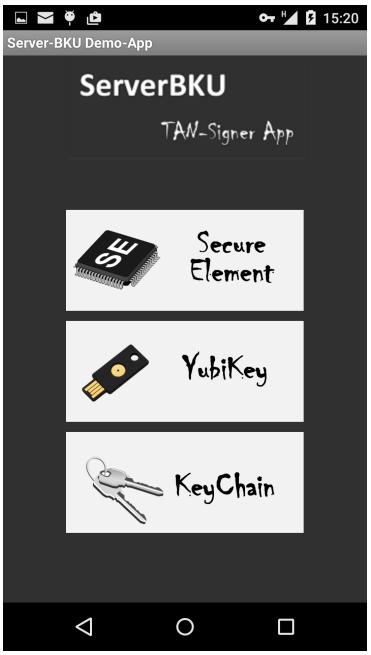

Figure 7.29: The TAN-Signer App provides a simple user interface to select the preferred technology used to sign TANs.

When the YubiKey has been selected as technology of choice, the TAN-Signer App displays a dialog to obtain required data from the Signatory. This dialog is shown in Figure 7.30 and contains two input fields. The Signatory is requested to enter his or her phone number in the first input field. If supported by the mobile operating system, the phone number is obtained automatically from the mobile device and is pre-populated. The activation code displayed in the Signatory's web browser has to be entered to the

second input field displayed. In case the TAN-Signer App has been started by scanning the displayed QR code, the activation code is entered automatically to the provided input field.

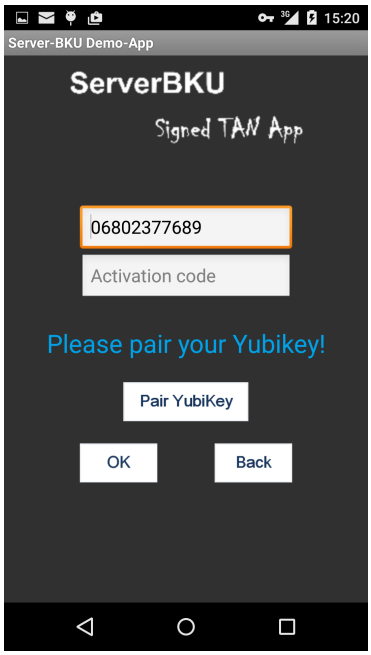

Figure 7.30: The TAN-Signer App provides a simple user interface to enter data required for the paring process.

In addition to the two input fields, the dialog displayed also features a button to pair the YubiKey. When pressing this button, the Signatory is requested to tap his or her personal YubiKey to the mobile end-user device. This is illustrated in Figure 7.31. As soon as the YubiKey is detected, the TAN-Signer App connects to it and reads out the public key of the token-specific key pair. According to the developed JavaCard applet that defines the YubiKey's functionality, the key pair is already generated, when the applet is installed on the token. This is beneficial, as it reduced the period of time, in which the YubiKey has to be present in the NFC field during the activation process. Once the public key has been successfully read from the YubiKey, the previous dialog is displayed again and the Signatory is notified accordingly as illustrated in Figure 7.32. By pressing the OK button and confirming all entered data, the Signatory initiates completion of the pairing process. All entered data and the public key read from the YubiKey are sent to the central ServerBKU and associated with the Signatory's newly created virtual signature token. Similar to the SE-specific realization variant, the Signatory is finally notified about the successful completion of the entire pairing process.

In contrast to the SE-specific realization variant, the Signatory does not have to define a secret PIN during the YubiKey-specific activation process. Accordingly, the YubiKey's signature-creation functionality is not protected by means of a PIN. While this obviously reduces security, it improves usability, as no additional data entry is required during signature-creation processes. Instead of entering a PIN, the Signatory indicates willingness to sign a TAN by putting the YubiKey in range of the mobile device's NFC reader. Omitting the use of PINs in the context of cryptography-enabled NFC tokens is common practice and also applied by various contactless payment systems. If required, support for local authentication, i.e. local PIN verification, could be easily added by modifying the YubiKey's JavaCard applet accordingly.

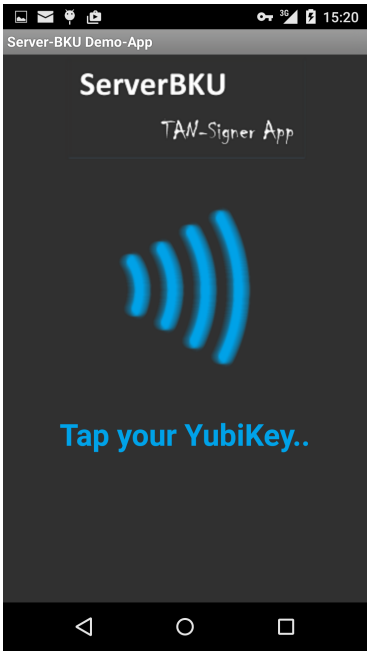

Figure 7.31: The TAN-Signer App requests the Signatory to tap his or her YubiKey to the mobile end-user device.

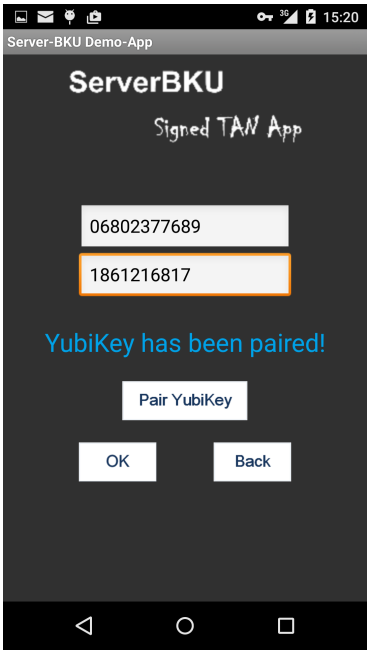

Figure 7.32: The Signatory can complete the pairing process by confirming all entered data.

7.3.3.4 Realization of the Signature-Creation Process

Refraining from using local authentication affects the signature-creation process as well. Still, the signature-creation process of the YubiKey-specific realization resembles to a large extent the one of the SE-specific implementation. The TAN-Signer App first requests the Signatory to enter his or her personal credentials, in order to identify the preferred virtual signature token. Concretely, the TAN-Signer App displays a dialog, through which the Signatory can enter his or her phone number and the secret password associated with the respective virtual signature token. This is illustrated in Figure 7.33.

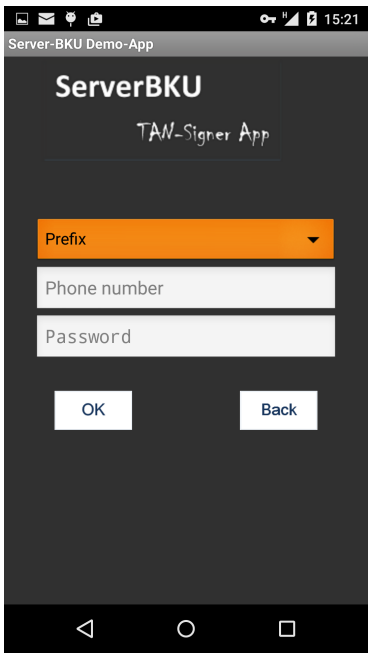

Figure 7.33: The TAN-Signer App requests the Signatory to enter phone number and password associated with the preferred virtual signature token.

Upon pressing the OK button, the TAN-Signer App transmits the entered credentials to the Server-BKU. This data transmission is protected by means of SSL/TLS. The ServerBKU verifies the obtained credentials by decrypting the virtual signature token referenced by the provided data. If this decryption process is successful, the ServerBKU generates a random TAN and sends it to the Signatory's mobile end-user device. To sign the received TAN locally, the Signatory needs to choose the preferred TAN-Signer Module first. For this purpose, the TAN-Signer App again displays the dialog shown in Figure 7.34.

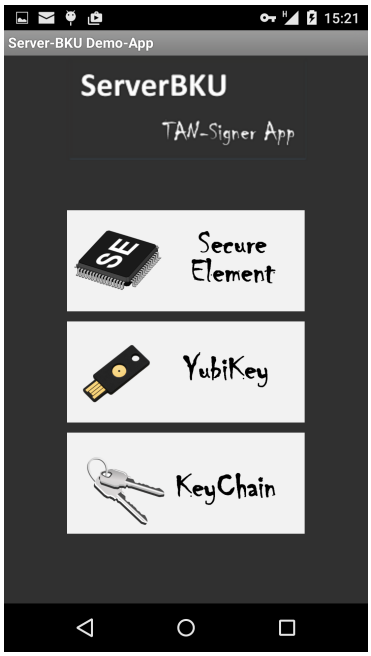

Figure 7.34: The TAN-Signer App requests the Signatory to select the respective TAN-Signer Module.

When the Signatory has selected the YubiKey as the preferred TAN-Signer Module, the TAN-Signer App displays a dialog to enter the received TAN. This dialog equals the one shown in the SE-specific implementation and provides the same features. Again, it displays a reference value to enable the Signatory to establish a binding between the received TAN and the current transaction. Furthermore, the shown dialog contains a button, which can be used to open a simple DTBS viewer. For the sake of completeness, this dialog is again shown in Figure 7.35.

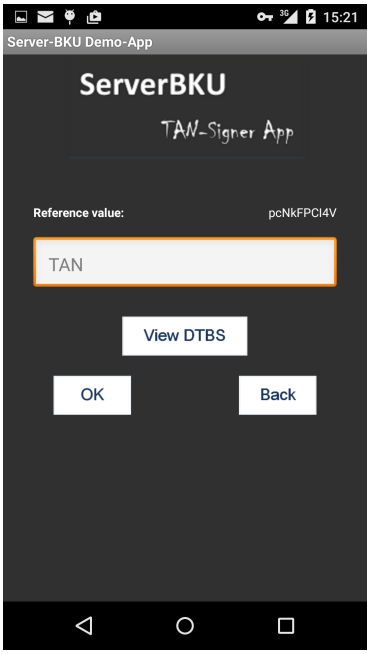

Figure 7.35: The TAN-Signer App requests the Signatory to enter the received TAN.

When the Signatory has entered the TAN and pressed the OK button, he or she is requested to tap his or her YubiKey to the mobile device. This enables the TAN-Signer App to send the entered TAN to the YubiKey, where it is finally signed. This is illustrated in Figure 7.36.

The YubiKey signs the obtained TAN. It then returns the signed TAN to the TAN-Signer App, which forwards it to the ServerBKU. There, the signed TAN is verified with the help of the Signatory's public key that has been obtained during the activation process. If this verification succeeds, the signature-creation process is completed.

Compared to the SE-specific signature-creation process, two basic differences can be identified. First, the Signatory does not need to enter a PIN to authorize the local signing of the TAN. Second, the Signatory needs to manually tap the YubiKey to the mobile device, in order to enable the local signing of the TAN. As tapping an NFC token is typically less complex than entering a PIN, the YubiKey-based realization variant can be regarded as more usable.

7.3.4 Realization Variant 3: KeyChains

Both realization variants discussed so far provide a sufficient level of security, but still suffer from a few drawbacks. Its rather poor feasibility is the main problem of the SE-based implementation. At present, only few mobile end-user devices are available that feature suitable SEs. In addition, capabilities to access available SEs from third-party apps heavily depend on the mobile operating system. Among the most popular platforms, only Android provides sufficient means to access SEs so far. In contrast to the SE-specific solution, the presented realization variant that relies on cryptography-enabled NFC tokens can be regarded as more feasible. However, this approach requires the Signatory to acquire and possess an additional NFC token. Depending on the respective deployment scenario, this might cause additional

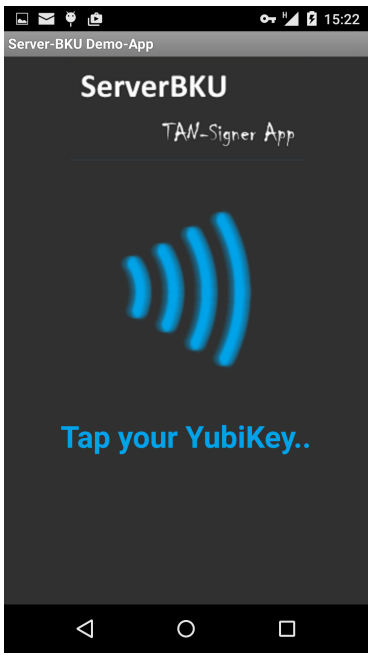

Figure 7.36: The TAN-Signer App requests the Signatory to tap his or her YubiKey to the mobile end-user device to sign the entered TAN.

costs and hence reduce user acceptance.

To mitigate problems of SE-based and NFC-based approaches, a third realization variant has been developed that relies on KeyChain implementations of current mobile operating systems. KeyChains are especially protected credential stores that can be used by third-party apps to store passwords, electronic certificates, or cryptographic key material. Provided features and underlying security concepts of current KeyChain implementations depend on the respective mobile platform and operating system. In the following subsections, current KeyChain implementations are briefly sketched and their use for the signing of TANs is discussed by means of a concrete realization.

7.3.4.1 Technology

The secure and reliable storage of credentials and cryptographic key material is a recurring problem also on mobile end-user devices. As these data usually have to be kept confidential, means must be provided that restrict access to credentials and keys to authorized entities. For this purpose, most mobile operating systems and platforms implement the concept of KeyChains. A KeyChain can be regarded as secure local storage for security-critical data and can be used by arbitrary third-party apps running on the mobile end-user device. Security concepts implemented by these KeyChains assure that access to these data is restricted to legitimate apps and users. Functionalities provided and security concepts implemented by KeyChains depend heavily on the particular mobile operating system and platform. In the following, the KeyChain implementations of the mobile platforms Android and iOS are exemplified. We have discussed details of these platform's KeyChain implementations in Teufl et al. [2013b] and Teufl et al. [2014a].

Android's KeyChain has been introduced in Android 4.0 and enables the secure storage of system-wide credentials that can be accessed by arbitrary apps if access is granted by the user. With Android 4.3, the platform's KeyChain implementation has been revised. The credential store has been renamed to Android Key Store[16] and now builds the basis for two features. First, it is used by Android's KeyChain

[16] https://developer.android.com/training/articles/keystore.html

API[17] to securely store credentials and cryptographic keys. Second, the Android Key Store is also used by the Android Keystore Provider[18], which has also been introduced in Android 4.3 and enhances the functionality of the Android KeyChain by enabling storage of app-specific credentials. Android 4.3 has also introduced support for hardware-backed Android Key Stores. Credentials and cryptographic keys can be stored such that they are not only stored in software but are protected by a secure hardware element. Depending on the capabilities of the underlying mobile end-user device, the hardware-backed Android Key Store can for instance be implemented by an SE, a TPM, or with the help of ARM's TrustZone technology[19]. As hardware support for the Android Key Store depends on the respective mobile end-user device, it is up to the developer to check whether hardware support is available and hence an increased level of security is provided.

The Android Key Store can be accessed and used through Android's KeyChain API or by means of the Android Keystore Provider. The KeyChain API enables third-party apps to install private cryptographic keys and associated electronic certificates to the Android Key Store. In addition, the KeyChain API enables third-party apps to retrieve stored keys and certificates. In contrast, the Android Keystore Provider can be accessed by means of standard JCE APIs. Compared to the KeyChain API, it provides additional functionality. Concretely, stored cryptographic keys and associated certificates can be used to carry out cryptographic operations in a secure way. For instance, the Android Keystore Provider enables third-party apps to create cryptographic key pairs in hardware-backed Android Key Stores. Furthermore, it also enables apps to securely use stored keys, e.g. to create electronic signatures. At the same time, it prevents private keys being generated and stored inside a hardware-backed Key Store from being exported.

Similar to Android, also iOS provides third-party apps a KeyChain implementation. Its underlying concept and basic functionality are comparable to Android's solution. The iOS KeyChain is mainly intended for the secure storage of credentials such as passwords or cryptographic keys. Its underlying security concept bases on the iOS-specific protection-class system, which provides means for data encryption on file level. This system is also employed by the iOS KeyChain and enables app developers to define different protection classes for KeyChain entries. The defined protection class essentially determines capabilities to access the respective KeyChain entry and specifies the way, in which the stored entry is protected by means of cryptographic mechanisms.

The iOS KeyChain has originally been designed for the secure storage of user credentials such as passwords. Today, is can also be used by third-party apps to securely store cryptographic keys and associated electronic certificates. For this purpose, iOS provides a C-based API to create and request certificate objects, to import certificates, keys, and identities, and to create asymmetric cryptographic key pairs. Although the iOS KeyChain relies on a hardware-based protection mechanism, KeyChain entries themselves are not directly stored in secure hardware. Instead, the secure hardware is required to derive the decryption key that is needed for the decryption of stored KeyChain entries.

Although their concrete implementations vary between different mobile platforms, KeyChains represent an interesting and easy-to-use alternative for the realization of the TAN-Signer App's TAN-Signer Module. The technology-specific internal structure of the TAN-Signer App, which can be derived from an integration of KeyChain technology, is presented and discussed in the next section.

7.3.4.2 Technology-Specific Internal Structure of the TAN-Signer App

Based on the decision to realize the TAN-Signer Module by means of KeyChain technology, the general architecture of the TAN-Signer App, which is shown in Figure 7.9 on page 214, can be further refined. The resulting technology-specific architecture is illustrated in Figure 7.37.

[17] http://developer.android.com/reference/android/security/KeyChain.html
[18] https://developer.android.com/reference/java/security/KeyStore.html
[19] http://www.arm.com/products/processors/technologies/trustzone/index.php

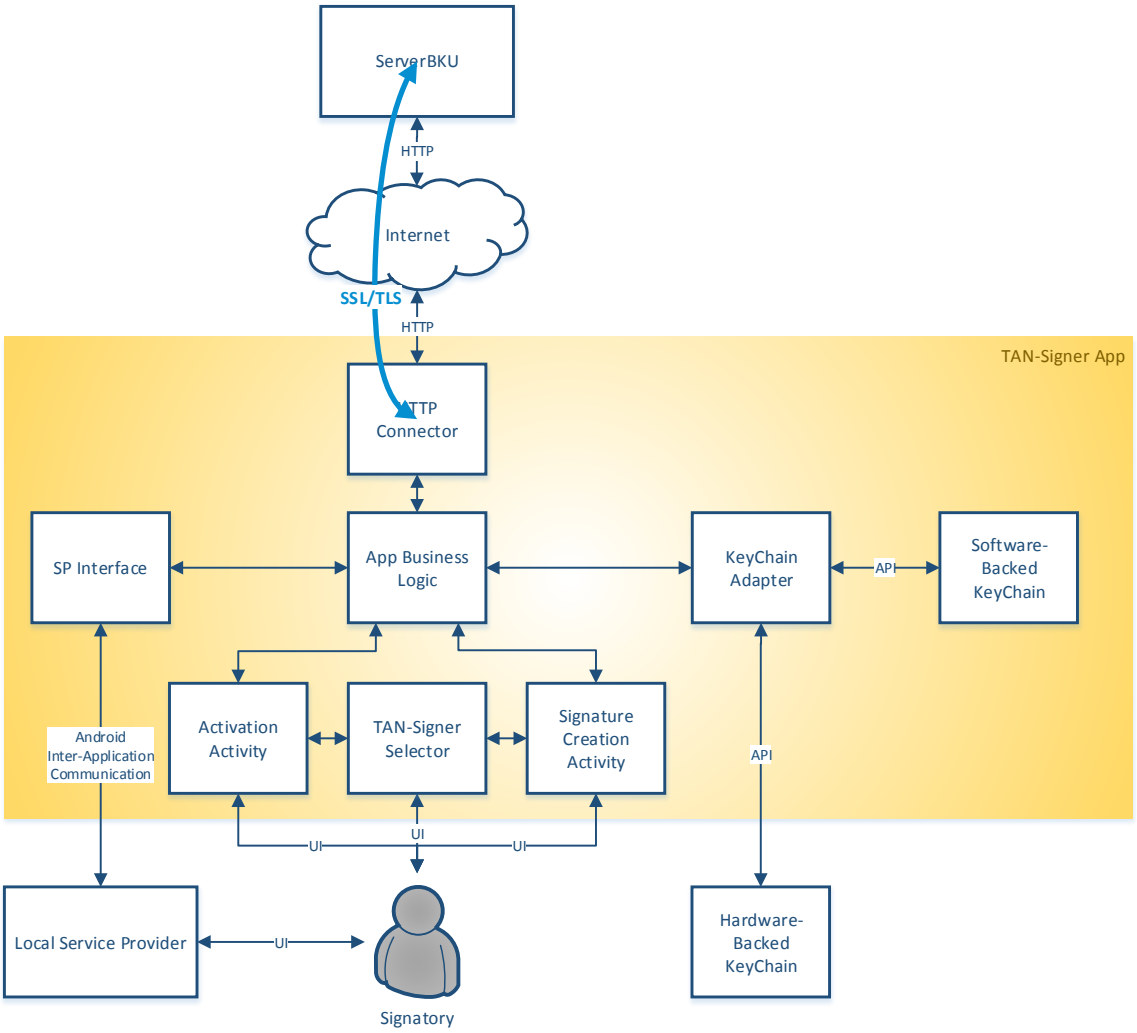

Figure 7.37: The architecture of the KeyChain-based TAN-Signer App shows relevant internal building blocks and interfaces.

To enable access to the underlying mobile platform's KeyChain implementation, the architecture shown in Figure 7.37 features a KeyChain Adapter. This component communicates with the respective KeyChain implementation through APIs provided by the mobile platform. Figure 7.37 also shows that depending on the underlying platform and on capabilities of the respective end-user device, employed KeyChains can be either software-backed or hardware-backed. In most cases, implementation details should be hidden by the provided API anyways. If this is not the case, the KeyChain Adapter needs to implement access to different available KeyChain implementations, in order to provide the App Business Logic a common implementation-independent interface.

Similar to the realization variant that bases on cryptography-enabled NFC tokens, the architecture shown in Figure 7.37 does not establish a secure channel between the remote ServerBKU and the local TAN-Signer Module, i.e. the KeyChain. For the NFC-based solution, secure messaging over a secure channel has been forgone by design, although cryptography-enabled NFC tokens would be technically capable to support this feature. This is not the case for the KeyChain-specific realization variant. Due to the limited set of KeyChain functionality that can be accessed through provided APIs, establishment of a secure channel between the remote ServerBKU and the local KeyChain is infeasible. Hence, communication between the TAN-Signer App and the ServerBKU is protected by means of SSL/TLS only.

7.3.4.3 Realization of the Activation Process

Reliance on KeyChains provided by the respective mobile platform limits the degree of freedom with regard to implementation of the TAN-Signer App. In particular, the achievable level of security can hardly be influenced and depends mainly on the KeyChain's security provided by the respective mobile platform. This low degree of freedom also affects the realization of the KeyChain-specific activation process. As no local authentication data needs to be defined and no communication with external tokens must be implemented, the resulting activation process is simpler compared to the activation processes of the NFC-based and SE-based implementation variants.

The first part of the activation process is identical for all implementation variants. In this part, the Signatory creates a new virtual signature token by accessing the ServerBKU's web interface with his or her web browser. After successful completion of the first part, an activation code is displayed that can be used to pair the Signatory's personal TAN-Signer App. In addition, a QR code, which contains the activation code, is displayed as well.

To start the second part of the activation process, i.e. the pairing process, the Signatory needs to launch the TAN-Signer App either manually or by scanning the displayed QR code. Similar to the activation process of the other two implementation variants, the TAN-Signer App first requests the Signatory to choose the preferred technology for the TAN-Signer Module by displaying the dialog illustrated in Figure 7.38.

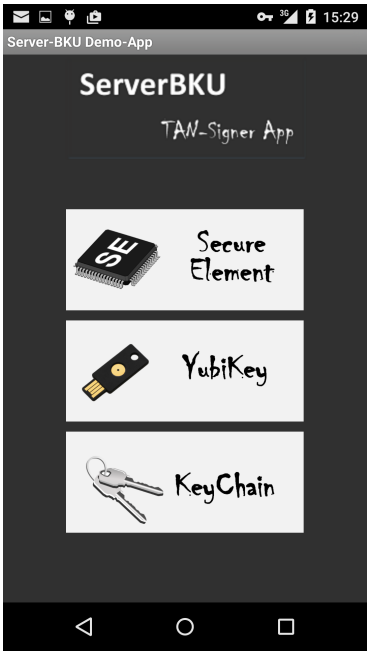

Figure 7.38: The TAN-Signer App provides a simple user interface to select the preferred technology used to sign TANs.

If the Signatory selects the KeyChain as the preferred technology, the TAN-Signer App displays a dialog to retrieve phone number and activation code from the Signatory. Similar to the other two realization variants, these data are pre-populated if possible, i.e. if the mobile operating systems allows automatic retrieval of the phone number and if the TAN-Signer App has been started by scanning the QR code. The displayed dialog is illustrated in Figure 7.39. In contrast to the SE-based or the NFC-based realization variant, the dialog does neither ask the Signatory to specify local authentication data, nor request the pairing of an external token.

When the Signatory presses the displayed OK button, the TAN-Signer App creates a new cryptographic key pair using the mobile device's KeyChain implementation, exports the public key from the

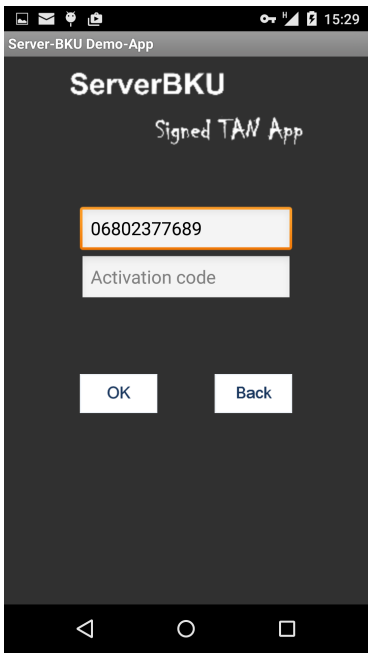

Figure 7.39: The TAN-Signer App provides a simple user interface to enter data required for the paring process.

KeyChain, and transfers the public key together with the data entered by the Signatory to the ServerBKU. There, the transferred data is stored and associated with the Signatory's newly created virtual signature token. Finally, the ServerBKU notifies the Signatory about the successful completion of the activation process.

7.3.4.4 Realization of the Signature-Creation Process

Similar to the activation process, also the signature-creation process benefits from reduced complexity, if the KeyChain is used as preferred technology. This becomes apparent from the following screenshots, which illustrate required interactions with the Signatory during a typical signature-creation process. To authorize a requested signature creation, the Signatory first needs to provide phone number and password associated with the preferred virtual signature token. For this purpose, the TAN-Signer App displays the dialog shown in Figure 7.40.

Entered and transferred data are verified as usual by the ServerBKU, which subsequently generates a random TAN and sends it to the Signatory via SMS. The Signatory, who is requested to sign the received TAN, again needs to select the appropriate technology first. This can again be accomplished by means of the dialog displayed by the TAN-Signer App and illustrated in Figure 7.41.

Selecting the KeyChain as preferred technology causes the TAN-Signer App to display another dialog, which enables the Signatory to view the DTBS and to enter the received TAN. This dialog is identical irrespective of the chosen technology and is already known from the other discussed realization variants. It is again illustrated in Figure 7.42 mainly for the sake of completeness.

Pressing the OK button causes the TAN-Signer App to request the KeyChain to sign the entered TAN. Internal security features of the respective KeyChain implementation assure that this process can be initiated by the TAN-Signer App and by the legitimate user only. The signed TAN is finally returned to the TAN-Signer App, which forwards is to the remote ServerBKU. There, the signed TAN is verified and the signature-creation process is completed.

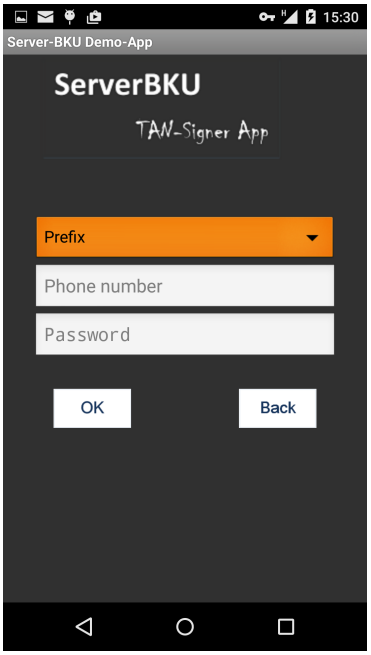

Figure 7.40: The TAN-Signer App requests the Signatory to enter phone number and password associated with the preferred virtual signature token.

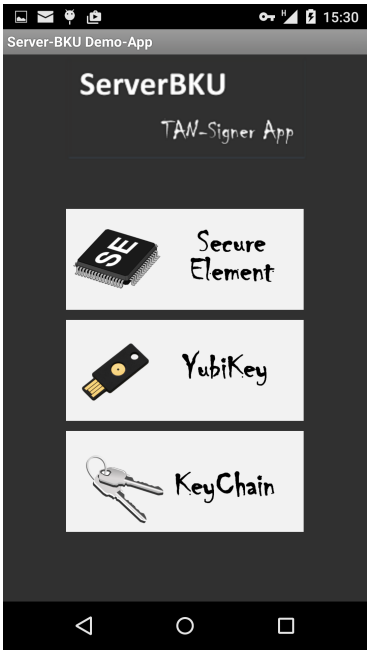

Figure 7.41: The TAN-Signer App requests the Signatory to select the respective TAN-Signer Module.

7.3.5 Future Work

The presented realization shows that the solution developed in this thesis, i.e. the Smartphone Signature, is feasible on current mobile end-user devices. Required cryptographic functionality has been realized on the client side by means of three different technologies. This has shown that the proposed solution is flexible enough to facilitate easy adoption of different technologies. This assures that the solution provides a sufficient degree of sustainability. At the same time, its concrete realization has also revealed that the proposed solution needs to cope with a trade-off between security, usability, and feasibility.

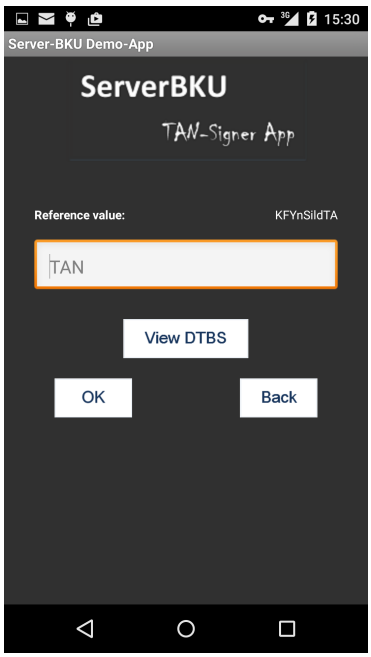

Figure 7.42: The TAN-Signer App requests the Signatory to enter the received TAN.

Technologies such as SEs that provide an increased level of security usually lack broad support on current mobile end-user devices. In addition, they often induce higher complexity, as they require additional user interactions. In contrast, widely supported and easy-to-use technologies such as KeyChains usually provide a lower level of security.

Taking into consideration these basic findings, future work on the proposed solution and its realization can be classified into two categories: integration of further technologies and migration of client components to alternative mobile platforms. These two categories of future work are motivated in the following subsections.

7.3.5.1 Integration of Further Technologies

Realization of the proposed solution has revealed that the choice of the technology used to implement TAN-signing functionality is a key criterion that influences the solution's security, usability, and feasibility. So far, the developed solution supports three different technologies, which all suffer to a certain extent from a trade-off between security, usability, and feasibility. To overcome this issue, mobile cutting-edge technologies will be continuously assessed in future and will be integrated into the existing solution to extend the set of supported technologies.

For the near future, ARM's TrustZone technology[20] appears to be a promising alternative. Following the approach taken by TrustZone technology, the entire mobile end-user device is divided into a Normal World and a Secure World. While the Normal World runs the rich mobile operating system, the Secure World is intended to execute small pieces of security-critical code. Essentially, components of the Normal World cannot access or intercept data being processed in the Secure World. In contrast to e.g. SE technology, the Secure World is not necessarily limited to a single secure hardware element, but can theoretically cover arbitrary system components including the CPU, memory, keyboards, and displays. This way, TrustZone technology does not only enable secure storage of security-critical data and secure execution of cryptographic functions, it also enables the realization of secure user interfaces. This makes ARM's TrustZone technology a powerful alternative to established security-preserving techniques.

[20] http://www.arm.com/products/processors/technologies/trustzone/index.php

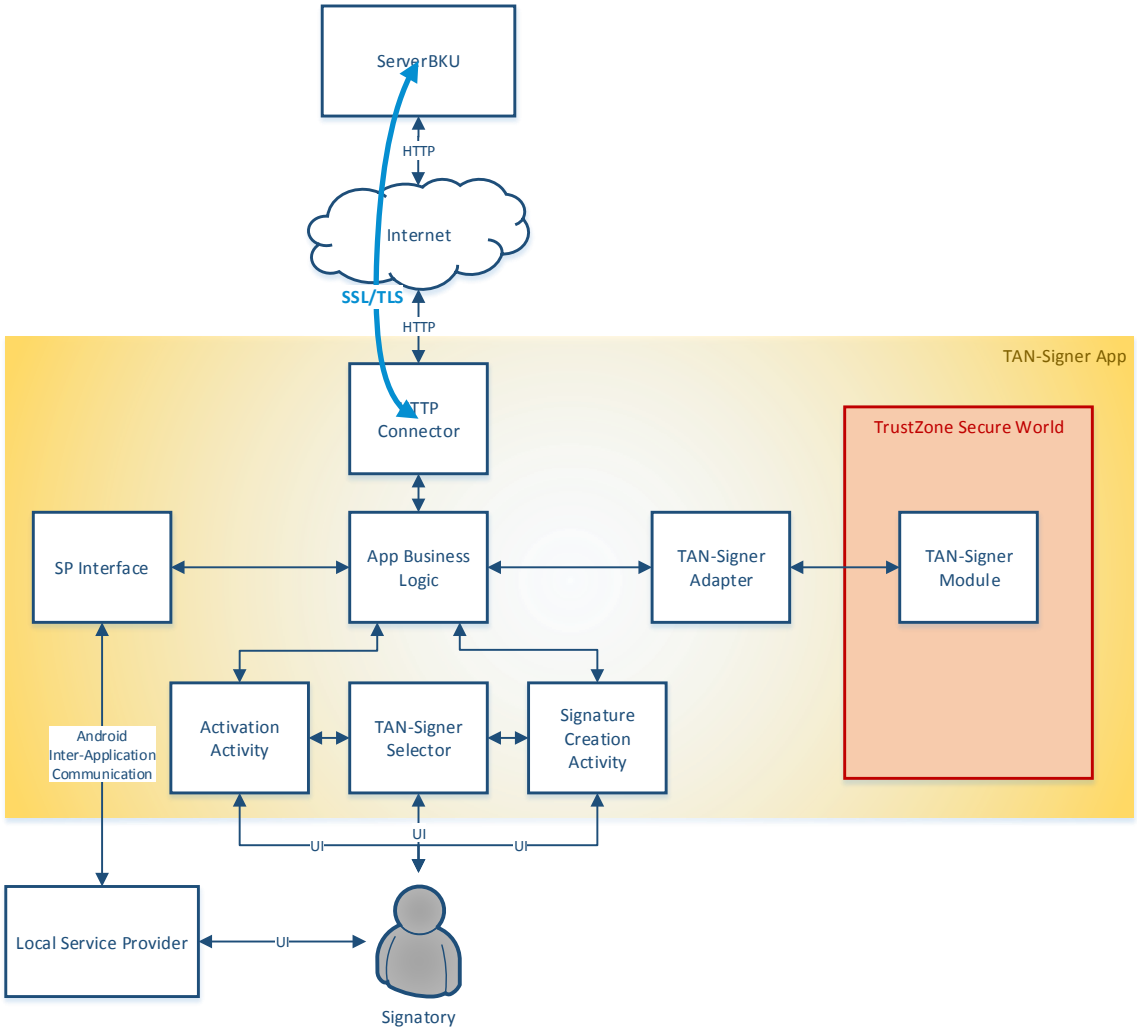

Figure 7.43: The possible architecture of the TrustZone-based TAN-Signer App shows relevant internal building blocks and interfaces.

A possible architecture of the TAN-Signer App that integrates TrustZone technology is shown in Figure 7.43. According to this architecture, the TrustZone's Secure World could be used to provide a secure execution environment for the TAN-Signer Module. If required, application of the Secure World could also span over additional components, e.g. to implement a secure user interface.

Its limited support by current mobile platforms is the main drawback of ARM's TrustZone technology. Although various devices already feature TrustZone-enabled hardware, current mobile operating systems provide limited support for this technology only. Broad and flexible use of TrustZone technology is hence still difficult to achieve for third-party apps. This makes TrustZone technology rather a promise for the future than a ready-to-use alternative.

Another possible alternative for the realization of TAN-signing functionality on the local mobile end-user device has recently emerged with the introduction of powerful wearable technologies. Popular examples are Google Glass[21], a wearable computer with an optical head-mounted display, or different

[21] https://www.google.com/glass/start/

kinds of smartwatches like Apple Watch[22]. Wearable technologies have in common that they are typically used as additional device together with a smartphone. Furthermore, they feature considerable computing power and communication technologies that enable a data exchange with other devices. Thus, wearable technologies are an interesting alternative to outsource security-critical operations from potentially insecure smartphones. In the context of the solution proposed in this thesis, a smartwatch could for instance be used to sign received TANs. This could prevent the smartphone from the need to securely store and apply confidential cryptographic key material. A possible architecture of the developed TAN-Signer App that makes use of wearable technology to sign TANs is shown in Figure 7.44.

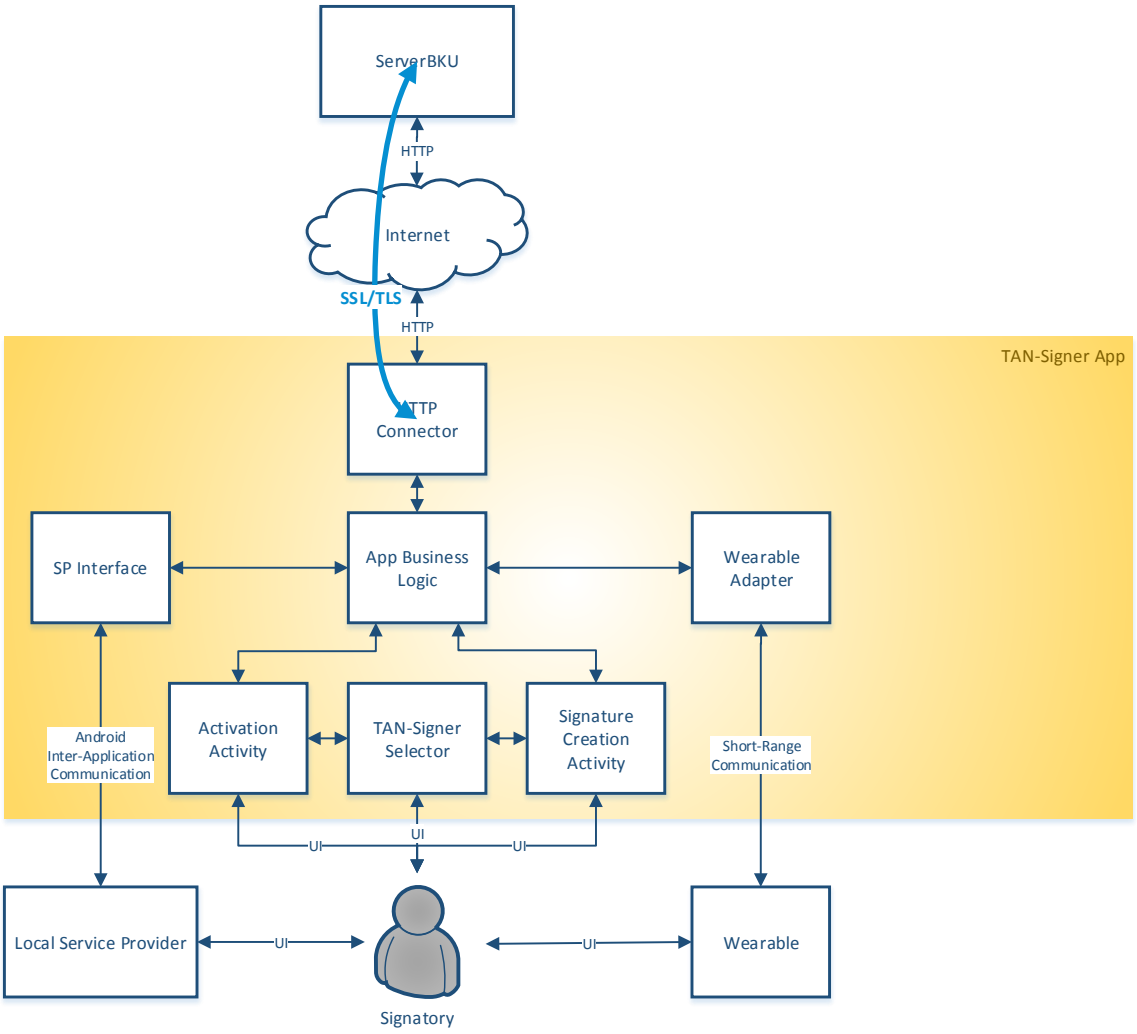

Figure 7.44: The possible architecture of a TAN-Signer App that is based on wearable technology shows relevant internal building blocks and interfaces.

In general, integration of an additional device increases the overall security, as a successful attack requires two distinct devices to be compromised. However, this approach is only applicable, if wearable technologies are sufficiently widespread. Even though this is not the case yet, wearable technology is expected to significantly gain relevance [Statista, 2015], making it a promise for the future.

[22] https://www.apple.com/watch/

7.3.5.2 Migration to Alternative Mobile Platforms

The current realization of the mobile signature solution proposed in this thesis has been based on the mobile platform Google Android. Choosing Android as the first target platform is reasonable, as this platform raises most challenges with regard to security. Current statistics indicate that 97% of mobile malware is on Android [Kelly, 2014]. Hence, a solution that satisfies relevant security requirements on Android should also be able to satisfy the same requirements on any other mobile platform. Realizing the proposed solution on Android first can hence be regarded as acid test for the solution's underlying concepts.

However, it must not be neglected that Android also provides application developers more flexibility compared to other mobile platforms. For instance, even though both Android and iOS devices support NFC technology, capabilities of provided APIs that can be used by third-party apps to communicate with NFC tags differ significantly between these platforms. Thus, migrating the developed solution to other mobile platforms and adapting implemented functionality where necessary in order to cope with limitations of these platforms is regarded as relevant future work as well.

7.4 Chapter Conclusions

In this chapter, a practical implementation of the proposed mobile signature solution Smartphone Signature has been presented. To assure applicability on different mobile end-user devices and to demonstrate the implementation's general flexibility, required cryptographic operations on the client side have been realized by means of three different technologies. The presented implementation has been developed by combining concepts of the proposed Smartphone Signature with the existing signature solution ServerBKU. Concretely, the ServerBKU has been extended and adapted such that it can be used following a single-device approach. This way, the ServerBKU has been released from the restriction to be used with two separate end-user devices and has been prepared for typical smartphone-based use cases and application scenarios.

The presented implementation has been realized on the Google Android platform. This platform has been chosen, as it currently has the highest market share among all mobile operating systems and provides developers a high degree of flexibility. Furthermore, Android must currently be regarded as the most problematic platform with regard to security. Thus, choosing this platform for a first implementation can be regarded as a perfect acid test for the underlying concept and architecture. However, porting the existing realization to other major mobile platforms such as Apple iOS or Microsoft Windows Phone 8 is considered for future work. In addition, the continuous assessment and integration of upcoming mobile technologies for the local realization of required cryptographic operations is regarded as future work as well.

The implementation presented in this chapter completes the evaluation of proposed models and solutions for the creation of qualified electronic signatures on mobile end-user devices. This way, it represents the third and final milestone of this thesis according to the methodology followed. The presented implementation shows that the proposed signature solution Smartphone Signature is indeed applicable and realizable in practice. Furthermore, it evaluates the implementation-independent and technology-agnostic model proposed in Part II of this thesis. As the Smartphone Signature has been derived from this model, the presented implementation shows this model's suitability to act as basis for the development of concrete signature solutions for mobile end-user devices.

Chapter 8

Conclusions

" It is good to have an end to journey toward, but it is the journey that matters in the end."

[Ursula K. Le Guin, American Author.]

During the past few years, the predominating computing paradigm has undergone significant changes. Classical end-user devices such as desktop computers and laptops have been gradually replaced by powerful mobile devices including smartphones and tablet computers. At the same time, user habits have changed accordingly. In the current always-on society, users expect to have access to information and services everywhere and at any time. This new mobile computing paradigm requires service providers to react and to adapt their services to changed circumstances and requirements. This also applies to public-sector agencies and administrations, which have followed the classical computing paradigm for years, in order to provide e-government services to citizens. In these days, public-sector organizations are requested to make their services ready for a use with mobile end-user devices and to complete the transition from e-government to m-government.

Enabling a successful transaction from classical e-government to m-government has been the main topic and basic goal of this thesis. Work on this thesis has yielded more than 50 publications in conference proceedings and scientific journals. To reach the thesis's goal, a thorough methodology has been followed. From an analysis of the current state of the art, the lack of suitable solutions to create legally binding electronic signatures on mobile end-user devices has been identified as main obstacle towards transactional m-government. A solution to overcome this obstacle has been proposed, discussed, and evaluated. The proposed solution represents a server-based signature solution that can be used on modern mobile end-user devices and enables the provision of transactional m-government services on these devices.

To assure its sustainability, the proposed solution has been defined and provided on different levels of abstraction. Long-term sustainability is assured by means of an abstract model that is independent from concrete technologies. In addition to this abstract model, a concrete solution called Smartphone Signature, which relies on currently available mobile technologies, has been proposed as well. Feasibility and applicability of this solution and its underlying model have finally been evaluated by means of a concrete implementation. By providing solutions on different levels of abstraction, sustainability and the capability to respond to frequent technological changes is guaranteed.

The signature solution for mobile end-user devices proposed in this thesis enables m-government services to integrate electronic-signature functionality. This way, the proposed solution overcomes the main obstacle that currently hinders a broad application of transactional m-government services. By enabling users to create legally binding electronic signatures on their mobile end-user devices, this thesis paves the way for transactional mobile services and thereby contributes to the successful evolution from e-government to m-government. In that sense, this thesis can be regarded as part of the journey towards

successful m-government and hence takes up the above quotation by Ursula K. Le Guin stating that it is good to have an end to journey toward, but that it is the journey that matters in the end.

Bibliography

Al-Hadidi, Ahmed and Yacine Rezgui [2009]. *Critical Success Factors for the Adoption and Diffusion of m-Government Services: A Literature Review*. In *Proceedings of the European Conference on e-Government, ECEG*, pages 21–28. ISBN 978-1-906638-33-7. ISSN 20491034. (Cited on pages 34 and 36.)

Al-khamayseh, Shadi, Elaine Lawrence, and Agnieszka Zmijewska [2007]. *Towards Understanding Success Factors in Interactive Mobile Government*. http://www.mgovernment.org/. (Cited on pages 34 and 36.)

Almulhem, Ahmad [2011]. *A Graphical Password Authentication System*. In *World Congress on Internet Security (WorldCIS-2011), London, UK, February 21*, volume 23. (Cited on page 174.)

Alrazooqi, Mansoor and Rohan De Silva [2010]. *Mobile and Wireless Services and Technologies for m-Government Solution Proposal for Dubai Government*. *WSEAS Transactions on Information Science and Applications*, 7, pages 1037–1047. ISSN 17900832. (Cited on page 34.)

ANSI [2005]. *Public Key Cryptography for the Financial Services Industry, The Elliptic Curve Digital Signature Algorithm (ECDSA)*. http://webstore.ansi.org/RecordDetail.aspx?sku=ANSI+X9.62:2005. (Cited on pages 24, 92 and 177.)

Antovski, Ljupco and Marjan Gusev [2005]. *M-Government Framework*. *Proceedings EURO mGov*, pages 10–12. (Cited on page 33.)

Apple [2013]. *Apple iPhone 5S Features*. http://www.apple.com/iphone-5s/features/. (Cited on page 147.)

Apple [2014]. *iOS Dev Center*. https://developer.apple.com/devcenter/ios/index.action. (Cited on page 107.)

Arnellos, Argyris, Dimitrios Lekkas, Dimitrios Zissis, Thomas Spyrou, and John Darzentas [2011]. *Fair Digital Signing: The Structural Reliability of Signed Documents*. *Computers & Security*, 30(8), pages 580–596. ISSN 0167-4048. doi:http://dx.doi.org/10.1016/j.cose.2011.09.001. http://www.sciencedirect.com/science/article/pii/S016740481100112X. (Cited on page 94.)

Barkan, Elad, Eli Biham, and Nathan Keller [2008]. *Instant Ciphertext-Only Cryptanalysis of GSM Encrypted Communication*. *Journal of Cryptology*, 21, pages 392–429. ISSN 09332790. doi:10.1007/s00145-007-9001-y. (Cited on page 66.)

Barrera, David, Paul C. van Oorschot, and Anil Somayaji [2010]. *A Methodology for Empirical Analysis of Permission-Based Security Models and its Application to Android Categories and Subject Descriptors*. In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, pages 73–84. CCS '10, ACM. ISBN 9781450302449. doi:10.1145/1866307.1866317. http://doi.acm.org/10.1145/1866307.1866317. (Cited on pages 65 and 118.)

Ben-Asher, Noam, Joachim Meyer, Sebastian Müller, and Roman Englert [2009]. *An Experimental System for Studying the Tradeoff between Usability and Security.* In *International Conference on Availability, Reliability and Security, 2009.*, pages 882–887. IEEE Computer Society. doi:10.1109/ARES.2009.174. http://dblp.uni-trier.de/db/conf/IEEEares/ares2009.html#Ben-AsherMME09. (Cited on page 171.)

Bhattacharyya, Debnath, Rahul Ranjan, Alisherov Farkhod, and Minkyu Choi [2009]. *Biometric Authentication: A Review*. *International Journal of u-and e-Service, Science and Technology*, 2(3), pages 13–28. (Cited on page 147.)

Bläsing, Thomas, Leonid Batyuk, Aubrey Derrick Schmidt, Seyit Ahmet Camtepe, and Sahin Albayrak [2010]. *An Android Application Sandbox System for Suspicious Software Detection*. In *Proceedings of the 5th IEEE International Conference on Malicious and Unwanted Software, Malware 2010*, pages 55–62. ISBN 9781424493555. doi:10.1109/MALWARE.2010.5665792. (Cited on page 65.)

Bowling, Drew [2012]. *Open Sesame: Google's Newest Security Log-In Uses QR Codes*. http://www.webpronews.com/open-sesame-googles-newest-security-log-in-uses-qr-codes-2012-01. (Cited on page 154.)

Carroll, Jennie [2005]. *Risky Business: Will Citizens Accept m-Government in the Long Term?* In *Proceedings of the First Mobile Government Conference (Euro mGov 2005)*, pages 77–87. Brighton. http://www.m4life.org/proceedings/2005/PDF/9_R376JC.pdf. (Cited on pages 33, 34 and 36.)

CEN [2004a]. *CWA 14169 - Secure Signature-Creation Devices "EAL 4+"*. Technical Report, European Committee for Standardization. (Cited on pages 94 and 96.)

CEN [2004b]. *CWA 14170 - Security Requirements for Signature Creation Applications*. http://standards.cen.eu/dyn/www/f?p=204:110:0::::FSP_PROJECT,FSP_ORG_ID:23764,400296&cs=1C1B2F4DF3464C9FD768CB422F16D3387. (Cited on pages 94, 95 and 97.)

CEN/ISSS [2001]. *Protection Profile - Secure Signature-Creation Device Type 3*. http://www.commoncriteriaportal.org/files/ppfiles/pp0006b.pdf. (Cited on pages 118 and 120.)

Check Point Software Technologies Ltd. [2012]. *Media Alert: Check Point and Versafe Uncover New Eurograbber Attack*. http://www.checkpoint.com/press/2012/120512-media-alert-cp-versafe-eurograbber-attack.html. (Cited on pages 65 and 66.)

Chin, Erika, Adrienne Porter Felt, Kate Greenwood, and David Wagner [2011]. *Analyzing Inter-Application Communication in Android*. In *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services, MobiSys 2011*, pages 239–252. MobiSys '11, ACM Press. ISBN 9781450306430. doi:10.1145/1999995.2000018. http://www.eecs.berkeley.edu/~emc/papers/mobi168-chin.pdf. (Cited on pages 107, 129 and 130.)

Common Criteria [2013]. *Common Criteria*. http://www.commoncriteriaportal.org/. (Cited on page 118.)

Curioso, Walter, Bryant Karras, Pablo Campos, Clara Buendia, King Holmes, and Ann Marie Kimball [2005]. *Design and Implementation of Cell-PREVEN: A Real-Time Surveillance System for Adverse Events Using Cell Phones in Peru. Annual Symposium Proceedings / AMIA Symposium*, pages 176–180. ISSN 1942-597X. doi:54572[pii]. (Cited on page 42.)

de Medeiros Napoles, S. H. L. and C. Zanchettin [2012]. *Offline Handwritten Signature Verification Through Network Radial Basis Functions Optimized by Differential Evolution*. In *The 2012 International Joint Conference on Neural Networks (IJCNN)*, pages 1–5. ISSN 2161-4393. doi:10.1109/IJCNN.2012.6252720. (Cited on page 91.)

Diaz, Jesus [2013]. *iPhone 5S Fingerprint Security Can Be Easily Broken, Hackers Show*. http://gizmodo.com/hackers-iphone-5s-fingerprint-security-is-not-secure-1367817697. (Cited on page 147.)

Dimauro, G., S. Impedovo, M. G. Lucchese, R. Modugno, and G. Pirlo [2004]. *Recent Advancements in Automatic Signature Verification*. In *Proceedings - International Workshop on Frontiers in Handwriting Recognition, IWFHR*, pages 179–184. ISBN 0769521878. ISSN 15505235. doi:10.1109/IWFHR.2004.85. (Cited on page 91.)

Donohue, Brian [2013]. *Weak Encryption Enables SIM Card Root Attack*. https://threatpost.com/weak-encryption-enables-sim-card-root-attack/101557. (Cited on page 132.)

El-Kiki, Tarek [2007]. *mGovernment: A Reality Check*. In *Conference Proceedings - 6th International Conference on the Management of Mobile Business, ICMB 2007*, page 37. IEEE. ISBN 0769528031. doi:10.1109/ICMB.2007.42. (Cited on pages 34, 35 and 36.)

El-Kiki, Tarek and Elaine Lawrence [2006]. *Mobile User Satisfaction and Usage Analysis Model of mGovernment Services*. In *Proceedings of the Second European Mobile Government Conference*, pages 91–102. (Cited on pages 34 and 36.)

EMarketer [2014]. *Worldwide Smartphone Usage to Grow 25% in 2014*. http://www.emarketer.com/Article/Worldwide-Smartphone-Usage-Grow-25-2014/1010920. (Cited on page 4.)

Enck, William and Damien Octeau [2011]. *A Study of Android Application Security. Proceedings of the 20th USENIX Conference on Security*, pages 21–21. ISSN 0364-2348. doi:10.1007/s00256-010-0882-8. http://www.usenix.org/event/sec11/tech/slides/enck.pdf. (Cited on pages 65 and 118.)

Enck, William, MacHigar Ongtang, and Patrick McDaniel [2009]. *Understanding Android Security*. *IEEE Security & Privacy*, 7, pages 50–57. ISSN 15407993. doi:10.1109/MSP.2009.26. (Cited on page 65.)

ENISA [2013]. *eID Authentication Methods in e-Finance and e-Payment Services*. Technical Report, European Union Agency for Network and Information Security (ENISA). http://www.enisa.europa.eu/activities/identity-and-trust/library/deliverables/eIDA-in-e-finance-and-e-payment-services. (Cited on page 151.)

ETSI [2003]. *ETSI TS 102 204: Mobile Commerce (M-COMM); Mobile Signature Service; Web Service Interface*. http://docbox.etsi.org//ec_files/ec_files/ts_102204v010104p.pdf. (Cited on page 29.)

ETSI [2009a]. *ETSI TS 101 903 - XML Advanced Electronic Signatures (XAdES)*. http://uri.etsi.org/01903/v1.4.1/ts_101903v010401p.pdf. (Cited on pages 92 and 93.)

ETSI [2009b]. *ETSI TS 102 778-1 - Electronic Signatures and Infrastructures (ESI); PDF Advanced Electronic Signature Profiles; Part 1: PAdES Overview - A Framework Document for PAdES*. http://www.etsi.org/deliver/etsi_ts/102700_102799/10277801/01.01.01_60/ts_10277801v010101p.pdf. (Cited on pages 92 and 93.)

ETSI [2013]. *ETSI TS 101 733 - Electronic Signatures and Infrastructures (ESI); CMS Advanced Electronic Signatures (CAdES)*. Technical Report, European Telecommunications Standards Institute. http://www.etsi.org/deliver/etsi_ts/101700_101799/101733/02.02.01_60/ts_101733v020201p.pdf. (Cited on page 93.)

ETSI [2014a]. *Conformity Assessment for Signature Creation and Validation Applications*. http://docbox.etsi.org/esi/Open/Latest_Drafts/prEN_419103_v002_conformity-assessment-sign-creation-validation_COMPLETE-draft.pdf. (Cited on page 94.)

ETSI [2014b]. *Electronic Signature*. http://www.etsi.org/technologies-clusters/technologies/security/electronic-signature. (Cited on page 93.)

European Commission [2014a]. *Digital Agenda for Europe*. http://ec.europa.eu/digital-agenda/. (Cited on pages 5, 20 and 23.)

European Commission [2014b]. *EU eGovernment Report 2014 Shows That Usability of Online Public Services is Improving, But Not Fast*. http://ec.europa.eu/digital-agenda/en/news/eu-egovernment-report-2014-shows-usability-online-public-services-improving-not-fast. (Cited on page 4.)

European Commission [2014c]. *Pillar I: Digital Single Market*. http://ec.europa.eu/digital-agenda/our-goals/pillar-i-digital-single-market. (Cited on page 5.)

European Council [2004]. *Multidisciplinary Ad Hoc Group of Specialists on Legal, Operational and Technical Standards for e-Enabled Voting (IP1-S-EE)*. https://wcd.coe.int/ViewDoc.jsp?id=768817&Lang=en. (Cited on page 118.)

Fang, Zhiyuan [2002]. *E-Government in Digital Era : Concept , Practice, and Development*. *International Journal of The Computer, The Internet and Management*, 10(2), pages 1–22. (Cited on page 20.)

Felt, Asrienne Porter, Elizabeth Ha, Serge Egelman, Ariel Haney, Erika Chin, and David Wagner [2012]. *Android Permissions: User Attention, Comprehension, and Behavior*. *Proceedings of the Eighth Symposium on Usable Privacy and Security*, pages 1–16. doi:10.1145/2335356.2335360. (Cited on pages 59, 66 and 118.)

Fingas, Jon [2014]. *Android Climbed to 79 Percent of Smartphone Market Share in 2013, But Its Growth Has Slowed*. http://www.engadget.com/2014/01/29/strategy-analytics-2013-smartphone-share/. (Cited on page 106.)

Flood, Derek, Rachel Harrison, Claudia Iacob, and David Duce [2013]. *Evaluating Mobile Applications: A Spreadsheet Case Study*. *International Journal of Mobile Human Computer Interaction (IJMHCI)*, 4(4), pages 37–65. (Cited on page 137.)

Frøkjær, Erik, Morten Hertzum, and Kasper Hornbæk [2000]. *Measuring Usability : Are Effectiveness, Efficiency, and Satisfaction Really Correlated?* In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 345–352. ACM, ACM. ISBN 1581132166. doi:10.1145/332040.332455. (Cited on page 136.)

GlobalPlatform [2009]. *GlobalPlatform Card Technology - Secure Channel Protocol 03 - Card Specification v 2.2 - Amendment D*. http://www.globalplatform.org/specificationscard.asp. (Cited on page 223.)

Goodin, Dan [2014]. *New iOS Flaw Makes Devices Susceptible to Covert Keylogging, Researchers Say*. http://arstechnica.com/security/2014/02/new-ios-flaw-makes-devices-susceptible-to-covert-keylogging-researchers-say/. (Cited on page 131.)

Google [2011]. *Introducing Android 4.0*. http://www.android.com/about/ice-cream-sandwich/. (Cited on page 147.)

Google [2014]. *Introduction to Android*. https://developer.android.com/guide/index.html. (Cited on page 107.)

Govindarajan, M. and R.M. Chandrasekaran [2011]. *Signature Verification Using Radial Basis Function Classifier*. *2011 3rd International Conference on Electronics Computer Technology*, 5, pages 182–185. doi:10.1109/ICECTECH.2011.5941981. (Cited on page 91.)

Gribsgy, Dan [2009]. *Apple Approved iPhone Inter-process Communication*. http://mobileorchard.com/apple-approved-iphone-inter-process-communication/. (Cited on page 109.)

GSMA [2013]. *Mobile Economy Europe 2013*. Technical Report, GSMA. http://gsmamobileeconomyeurope.com/GSMA_MobileEconomyEurope_v9_WEB.pdf. (Cited on page 28.)

Gutmann, Peter and Ian Grigg [2005]. *Security Usability*. *IEEE Security & Privacy*, 3(4), pages 56–58. ISSN 1540-7993. doi:10.1109/MSP.2005.104. (Cited on page 171.)

Harrison, Rachel, Derek Flood, and David Duce [2013]. *Usability of Mobile Applications: Literature Review and Rationale for a New Usability Model*. *Journal of Interaction Science*, 1(1), page 1. ISSN 2194-0827. doi:10.1186/2194-0827-1-1. (Cited on pages 136 and 137.)

Hassan, Mohammad, Tareq Jaber, and Zina Hamdan [2009]. *Adaptive Mobile-Government Framework*. In *Proceedings of International Conference on Administrative Development: Towards Excellence in Public Sector Performance*, pages 1–11. (Cited on page 37.)

Heo, Jeongyun, Dong Han Ham, Sanghyun Park, Chiwon Song, and Wan Chul Yoon [2009]. *A Framework for Evaluating the Usability of Mobile Phones Based on Multi-Level, Hierarchical Model of Usability Factors*. *Interacting with Computers*, 21(4), pages 263–275. (Cited on page 136.)

Hornbæk, Kasper [2006]. *Current Practice in Measuring Usability: Challenges to Usability Studies and Research*. *International Journal of Human Computer Studies*, 64(2), pages 79–102. (Cited on page 135.)

IDC [2014]. *Smartphone OS Market Share, Q2 2014*. http://www.idc.com/prodserv/smartphone-os-market-share.jsp. (Cited on pages 4, 32, 57 and 62.)

International Finance Corporation [2014]. *Infrastructure in Africa*. http://www.ifc.org/wps/wcm/connect/region__ext_content/regions/sub-saharan+africa/investments/infrastructure. (Cited on page 37.)

Internet Engineering Task Force [2011]. *TOTP: Time-Based One-Time Password Algorithm*. http://tools.ietf.org/html/rfc6238. (Cited on page 153.)

ISO/IEC [2008]. *ISO/IEC 14443-1:2008*. http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?csnumber=39693. (Cited on page 230.)

ISO/IEC [2010]. *ISO 9241-210:2010 - Ergonomics of Human-System Interaction – Part 210: Human-Centred Design for Interactive Systems*. http://www.iso.org/iso/catalogue_detail.htm?csnumber=52075. (Cited on page 135.)

ISO/IEC [2012]. *ISO/IEC 21481:2012*. http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=56855. (Cited on page 230.)

ISO/IEC [2013]. *ISO/IEC 18092:2013*. http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?csnumber=56692. (Cited on page 230.)

Jotischky, Nick and Sheridan Nye [2011]. *Mobilizing Public Services in Africa: The m-Government Challenge*. Technical Report, Informa UK Ltd. http://www.informatandm.com/wp-content/uploads/2012/02/ITM-M-Government-White-Paper.pdf. (Cited on page 37.)

Karadimas, Nikolaos, Katerina Papatzelou, and Agisilaos Papantoniou [2008]. *M-Government Services in Greece*. In *Proceedings of the 22nd European Conference on Modelling and Simulation*, pages 71–74. ISBN 978-0-9553018-5-8. (Cited on page 34.)

Karan, Kavita and MCH Khoo [2008]. *Mobile Diffusion and Development: Issues and Challenges of m-Government with India in Perspective*. In *Proceedings of the 1st International Conference on M4D Mobile Communication Technology for Development*, pages 138–149. (Cited on pages 34 and 36.)

Kelly, Gordon [2014]. *Report: 97% Of Mobile Malware Is On Android. This Is The Easy Way You Stay Safe*. http://www.forbes.com/sites/gordonkelly/2014/03/24/report-97-of-mobile-malware-is-on-android-this-is-the-easy-way-you-stay-safe/. (Cited on pages 54, 68, 74, 213 and 246.)

Ketola, Pekka and Mika Röykkee [2001]. *The Three Facets of Usability In Mobile Handsets*. In *CHI 2002 Workshop on Mobile Communications: Understanding Users, Adoption & Design*. (Cited on page 136.)

Khan, Wazir Zada, Mohammed Aalsalem, and Yang Xiang [2011]. *A Graphical Password Based System for Small Mobile Devices*. *International Journal of Computer Science Issues*, 8(5), pages 145–154. (Cited on page 174.)

Klockar, Tomas, David Carr, Anna Hedman, Tomas Johansson, and Fredrik Bengtsson [2003]. *Usability of Mobile Phones*. In *Proceeding of the 19th International Symposium on Human Factors in Telecommunications*, pages 197–204. (Cited on page 136.)

Knall, Thomas, Arne Tauber, Thomas Zefferer, Bernd Zwattendorfer, Arnaldur Axfjord, and Haraldur Bjarnason [2011]. *Secure and Privacy-Preserving Cross-Border Authentication: The STORK Pilot SaferChat*. In *Proceedings of the Conference on Electronic Government and the Information Systems Perspective (EGOVIS 2011)*, pages 94–106. Springer. (Cited on pages 13 and 24.)

Kocher, Paul, Joshua Jaffe, and Benjamin Jun [1999]. *Differential Power Analysis*. In *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, pages 388–397. Springer. ISBN 978-3-540-66347-8. ISSN 03029743. (Cited on page 68.)

Kocher, PC [1996]. *Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems*. *Advances in Cryptology*, 1109, pages 104–113. ISSN 03029743. doi:10.1007/3-540-68697-5_9. (Cited on page 68.)

Krnjic, Vesna, Philip Weber, Thomas Zefferer, and Bernd Zwattendorfer [2013]. *Effizientes Testen von E-Government Komponenten in der Cloud*. In *D-A-CH Security 2013*, pages 225–236. syssec. (Cited on page 15.)

Kumar, Manish and Omesh Prasad Sinha [2007]. *M-Government - Mobile Technology for e-Government*. In *International Conference on e-Government*, pages 294–301. (Cited on pages 35, 36 and 37.)

Kushchu, Ibrahim and M. Halid Kuscu [2004]. *From e-Government to m-Government : Facing the Inevitable*. *Proceedings of the 3rd European Conference on eGovernment*, pages 1–13. (Cited on page 33.)

Lashkari, Arash Habibi, Azizah Abdul Manaf, Maslin Masrom, and Salwani Mohd Daud [2011]. *Security Evaluation for Graphical Password*. In *Digital Information and Communication Technology and Its Applications*, pages 431–444. Springer. (Cited on page 174.)

Leitold, Herbert, Arno Hollosi, and Reinhard Posch [2002]. *Security Architecture of the Austrian Citizen Card Concept*. In *18th Annual Computer Security Applications Conference, 2002. Proceedings.*, pages 391–400. ISSN 1063-9527. doi:10.1109/CSAC.2002.1176311. (Cited on pages 23 and 94.)

Leitold, Herbert, Reinhard Posch, and Thomas Rössler [2009]. *Media-Break Resistant eSignatures in eGovernment - an Austrian Experience*. In *Emerging Challenges for Security, Privacy, and Trust - 24th IFIP SEC*, pages 109–118. IFIP Advances in Information and Communication Technologies, Springer. (Cited on pages 81 and 92.)

Lenz, Thomas, Klaus Stranacher, and Thomas Zefferer [2013a]. *Enhancing the Modularity and Applicability of Web-Based Signature-Verification Tools*, pages 173–188. Springer. ISBN 978-3-662-44299-9. (Cited on page 15.)

Lenz, Thomas, Klaus Stranacher, and Thomas Zefferer [2013b]. *Towards a Modular Architecture for Adaptable Signature-Verification Tools.* In *Proceedings of the 9th International Conference, WEBIST 2013*, pages 325–334. SciTePress. ISBN 978-989-8565-54-9. (Cited on pages 15 and 86.)

Madden, Gary, Erik Bohlin, Hajime Oniki, and Thien Tran [2013]. *Potential Demand for M-Government Services in Japan*. *Applied Economics Letters*, 20, pages 732–736. ISSN 1350-4851, 1350-4851. doi:10.1080/13504851.2012.736939. (Cited on page 34.)

Mairiza, D and D Zowghi [2010]. *An Ontological Framework to Manage the Relative Conflicts Between Security and Usability Requirements*. In *Third International Workshop on Managing Requirements Knowledge (MARK)*, pages 1–6. doi:10.1109/MARK.2010.5623814. (Cited on page 171.)

Mengistu, Desta, Hangjung Zo, and Jae Jeung Rho [2009]. *M-Government: Opportunities and Challenges to Deliver Mobile Government Services in Developing Countries*. In *ICCIT 2009 - 4th International Conference on Computer Sciences and Convergence Information Technology*, pages 1445–1450. ISBN 9780769538969. doi:10.1109/ICCIT.2009.171. (Cited on page 35.)

Microsoft Corporation [2012]. *Windows Phone 8 Security Overview*. Technical Report, Microsoft Corporation. (Cited on page 61.)

Misra, D. C. [2010]. *Make m-Government an Integral Part of e-Government: An Agenda for Action*. In *National Forum on Mobile Applications for Inclusive Growth and Sustainable Development*. (Cited on pages 33 and 37.)

Mobi Solutions Ltd. [2010]. *Mobile Government: 2010 and Beyond*. Technical Report, MobiSolutions Ltd. (Cited on page 37.)

MobiForge [2014]. *Global Mobile Statistics 2014 Home: All the Latest Stats on Mobile Web, Apps, Marketing, Advertising, Subscribers, and Trends...* `http://mobiforge.com/research-analysis/global-mobile-statistics-2014-home-all-latest-stats-mobile-web-apps-marketing-advertising-subscriber`. (Cited on page 4.)

MobiThinking [2014]. *Global Mobile Statistics 2014 Part A: Mobile Subscribers; Handset Market Share; Mobile Operators*. `http://mobithinking.com/mobile-marketing-tools/latest-mobile-stats/a`. (Cited on page 213.)

Moon, M Jae [2010]. *Shaping M-Government for Emergency Management: Issues and Challenges*. *Journal of E-Governance*, 33, pages 100–107. ISSN 18787673. doi:10.3233/GOV-2010-0217. (Cited on page 35.)

Nariman, Dahlan and Susumu Yamamoto [2008]. *An Evaluation of Information Quality of e-Government in Indonesia*. In *Proceedings of ICEG 2008 The International Conference on e-Government*, pages 304–318. (Cited on pages 20 and 21.)

Network Working Group [2005]. *HOTP: An HMAC-Based One-Time Password Algorithm*. http://tools.ietf.org/html/rfc4226. (Cited on page 153.)

Network Working Group [2008]. *RFC 5246: The Transport Layer Security (TLS) Protocol Version 1.2*. http://tools.ietf.org/html/rfc5246. (Cited on pages 25 and 130.)

NFC World [2015]. *NFC phones: The definitive list*. http://www.nfcworld.com/nfc-phones-list/. (Cited on page 230.)

Nguyen, Vu, Michael Blumenstein, Vallipuram Muthukkumarasamy, and Graham Leedham [2007]. *Off-line Signature Verification Using Enhanced Modified Direction Features in Conjunction with Neural Classifiers and Support Vector Machines*. In *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, volume 2, pages 734–738. ISBN 0769528228. ISSN 15205363. doi:10.1109/ICDAR.2007.4377012. (Cited on page 91.)

Nielsen [2014]. *How Smartphones are Changing Consumer's Daily Routines Around the Globe*. Technical Report, Nielsen. http://www.nielsen.com/us/en/insights/news/2014/how-smartphones-are-changing-consumers-daily-routines-around-the-globe.html. (Cited on page 32.)

Nielsen, Jakob [1993]. *Usability Engineering*. Interactive Technologies, Morgan Kaufmann. ISBN 0125184069. doi:10.1145/1508044.1508050. http://www.useit.com/jakob/useengbook.html. (Cited on page 137.)

NIST [2001]. *Announcing the ADVANCED ENCRYPTION STANDARD (AES)*. http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf. (Cited on page 58.)

Norris, Donald F and M Jae Moon [2005]. *Advancing e-Government at the Grassroots: Tortoise or Hare? Public Administration Review*, 65, pages 64–75. ISSN 0033-3352. doi:10.1111/j.1540-6210.2005.00431.x. (Cited on page 37.)

Oberauer, Stephan [2012]. *Access of Mobile Security Card with Android*. Bachelor thesis, Graz University of Technology. (Cited on page 215.)

Orthacker, Clemens, Martin Centner, and Christian Kittl [2010]. *Qualified Mobile Server Signature*. In *Proceedings of the 25th TC 11 International Information Security Conference SEC 2010*. (Cited on pages 30, 47, 103, 183, 192, 193, 194, 195, 196, 197, 198 and 201.)

Orthacker, Clemens and Thomas Zefferer [2011]. *Accessibility Challenges in e-Government: An Austrian Experience*. In *Proceedings of the Forth International Conference on Internet Technologies and Applications (ITA 11)*, pages 221–228. Glyndwr University. (Cited on page 12.)

Oxford University Press [2014]. *Oxford Dictionaries*. http://www.oxforddictionaries.com. (Cited on page 23.)

PC/SC Workgroup [2014]. *PC/SC Workgroup Specifications*. http://www.pcscworkgroup.com/specifications/overview.php. (Cited on page 216.)

Posch, Karl-Christian, Reinhard Posch, Arne Tauber, Thomas Zefferer, and Bernd Zwattendorfer [2011]. *Secure and Privacy-preserving eGovernment - Best Practice Austria*. In *Rainbow of Computer Science*, pages 259–269. Springer. ISBN 987-3-642-19391-0. (Cited on pages 13, 21 and 77.)

Posch, Reinhard, Clemens Orthacker, Klaus Stranacher, Arne Tauber, Thomas Zefferer, and Bernd Zwattendorfer [2012]. *Open Source Bausteine als Kooperationsgrundlage*, pages 185–209. Springer. (Cited on page 14.)

Rath, Christof, Simon Roth, Harald Bratko, and Thomas Zefferer [in press]. *Encryption-based Second Authentication Factor Solutions for Qualified Server-side Signature Creation*. In *4th International Conference on Electronic Government and the Information Systems Perspective*. Springer. (Cited on page 16.)

Rath, Christof, Simon Roth, Manuel Schallar, and Thomas Zefferer [2014a]. *A Secure and Flexible Server-Based Mobile eID and e-Signature Solution*. In *The Eighth International Conference on Digital Society*, pages 7–12. IARIA. (Cited on pages 16, 192, 198 and 219.)

Rath, Christof, Simon Roth, Manuel Schallar, and Thomas Zefferer [2014b]. *Design and Application of a Secure and Flexible Server-Based Mobile eID and e-Signature Solution*. *International Journal on Advances in Security*, 7, pages 50–61. (Cited on page 16.)

Reimair, Florian, Peter Teufl, and Thomas Zefferer [2015]. *WebCrySIL - Web Cryptographic Service Interoperability Layer*. In *11th International Conference on Web Information Systems and Technologies*, pages 35–44. SciTePress. (Cited on page 11.)

Reiter, Andreas and Thomas Zefferer [2015]. *Paving the Way for Security in Cloud-Based Mobile Augmentation Systems*. In *3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (IEEE Mobile Cloud 2015)*, pages 89–98. IEEE Computer Society. (Cited on page 11.)

Renner, Ray, Matt Moran, Zohra Hemani, Ellins Thomas, Harold Pio, and Alejandro Vargas [2011]. *A Comparison of Mobile GIS Development Options on Smart Phone Platforms*. In *Proceedings of the 2nd International Conference on Computing for Geospatial Research & Applications*, page 44:1. COM.Geo '11, ACM, New York, NY, USA. ISBN 978-1-4503-0681-2. doi:10.1145/1999320.1999365. (Cited on page 107.)

Republik Österreich [2008]. *Verordnung des Bundeskanzlers über elektronische Signaturen (Signaturverordnung 2008 - SigV 2008)*. `https://www.ris.bka.gv.at/GeltendeFassung.wxe?Abfrage=Bundesnormen&Gesetzesnummer=20005618`. (Cited on page 146.)

Republik Österreich [2010]. *Bundesgesetz über elektronische Signaturen (Signaturgesetz - SigG)*. `http://ris.bka.gv.at/GeltendeFassung.wxe?Abfrage=Bundesnormen&Gesetzesnummer=10003685`. (Cited on page 146.)

Rivest, Ronald Linn, Adi Shamir, and Leonard Adleman [1978]. *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*. *Commun. ACM*, 21(2), pages 120–126. ISSN 0001-0782. doi:10.1145/359340.359342. (Cited on pages 24, 92 and 177.)

Rogers, Michael and Mark Goadrich [2012]. *A Hands-on Comparison of iOS vs. Android*. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*, page 663. SIGCSE '12, ACM, New York, NY, USA. ISBN 978-1-4503-1098-7. doi:10.1145/2157136.2157330. (Cited on pages 65 and 107.)

Sareen, Mamta, Devendra Kumar Punia, and Lovneesh Chanana [2013]. *Exploring Factors Affecting Use of Mobile Government Services in India*. *Problems and Perspectives in Management : PPM.-Sumy: Business Perspectives*, 11, pages 86–93. (Cited on pages 34 and 36.)

Schindler, Werner [2000]. *A Timing Attack Against RSA with the Chinese Remainder Theorem*. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 1965 LNCS, pages 109–124. ISBN 354041455X. ISSN 03029743. doi:10.1007/3-540-44499-8-8. (Cited on page 68.)

Scholl, Jochen Hans, Raya Fidel, and Jens-Erik Mai [2006]. *The Fully Mobile City Government Project (MCity)*. In *Proceedings of the 2006 National Conference on Digital Government Research*, pages 131–132. doi:10.1145/1146598.1146639. (Cited on page 34.)

Shabtai, Asaf, Yuval Fledel, Uri Kanonov, Yuval Elovici, Shlomi Dolev, and Chanan Glezer [2010]. *Google Android: A Comprehensive Security Assessment*. *IEEE Security & Privacy*, 8, pages 35–44. ISSN 15407993. doi:10.1109/MSP.2010.2. (Cited on page 65.)

Sheng, Hong and Silvana Trimi [2008]. *M-Government: Technologies, Applications and Challenges*. *International Journal on Electronic Government*, 5, pages 1–18. ISSN 1740-7494. doi:10.1504/EG.2008.016124. (Cited on page 37.)

Silverman, Dwight [2011]. *Android 4.0's Facial Recognition Is Cool, But Don't Trust It Yet*. http://blog.chron.com/techblog/2011/12/android-4-0s-facial-recognition-is-cool-but-dont-trust-it-yet/. (Cited on page 147.)

Sony [2015]. *FeliCa*. http://www.sony.net/Products/felica/index.html. (Cited on page 230.)

Statista [2015]. *Facts and statistics on Wearable Technology*. http://www.statista.com/topics/1556/wearable-technology/. (Cited on page 245.)

Stockman, H. [1948]. *Communication by Means of Reflected Power*. *Proceedings of the IRE*, 36. ISSN 0096-8390. doi:10.1109/JRPROC.1948.226245. (Cited on page 229.)

Stranacher, Klaus, Vesna Krnjic, and Thomas Zefferer [2012]. *Vertrauenswürdiges Open Government Data*. In *1.OGD D-A-CH-LI Konferenz*, pages 27–39. ADV. (Cited on page 14.)

Stranacher, Klaus, Vesna Krnjic, and Thomas Zefferer [2013a]. *Authentische und integritätsgesicherte Verwaltungsdaten*. *eGovernment Review*, 11, pages 30–31. (Cited on page 15.)

Stranacher, Klaus, Vesna Krnjic, and Thomas Zefferer [2013b]. *Trust and Reliability for Public Sector Data*. In *Proceedings of International Conference on e-Business and e-Government*, volume 73, pages 124–132. ACPI. (Cited on page 15.)

Stranacher, Klaus, Vesna Krnjic, Bernd Zwattendorfer, and Thomas Zefferer [2013c]. *Assessment of Redactable Signature Schemes for Trusted and Reliable Public Sector Data*. In *Proceedings of the 13th European Conference on e-Government*, pages 508–516. ACPI. (Cited on page 15.)

Stranacher, Klaus, Vesna Krnjic, Bernd Zwattendorfer, and Thomas Zefferer [2013d]. *Evaluation and Assessment of Editable Signatures for Trusted and Reliable Public Sector Data*. *Electronic Journal of e-Government*, 11, pages 360–372. (Cited on page 15.)

Stranacher, Klaus, Arne Tauber, Thomas Zefferer, and Bernd Zwattendorfer [2013e]. *The Austrian Identity Ecosystem - An e-Government Experience*, pages 288–309. IGI Global. (Cited on page 16.)

Suo, Xiaoyuan, Ying Zhu, and G. Scott. Owen [2005]. *Graphical Passwords: A Survey*. In *21st Annual Computer Security Applications Conference*, pages 463–472. ISSN 1063-9527. doi:10.1109/CSAC.2005.27. (Cited on page 174.)

Symantec [2010]. *Trojan.Zbot*. http://www.symantec.com/security_response/writeup.jsp?docid=2010-011016-3514-99. (Cited on page 66.)

Syverson, Paul [1994]. *A Taxonomy of Replay Attacks*. In *Proceedings of the 7th IEEE Computer Security Foundations Workshop*, pages 187–191. Society Press. (Cited on page 152.)

Tauber, Arne, Thomas Zefferer, and Bernd Zwattendorfer [2011a]. *Elektronisches Einschreiben im D-A-CH Raum*. In *D-A-CH Security 2011*, pages 510–521. syssec. (Cited on page 13.)

Tauber, Arne, Thomas Zefferer, and Bernd Zwattendorfer [2012]. *Approaching the Challenge of eID Interoperability: An Austrian Perspective*. *European Journal of ePractice*, 14, pages 22–39. (Cited on page 14.)

Tauber, Arne, Bernd Zwattendorfer, and Thomas Zefferer [2011b]. *A Shared Certified Mail System for the Austrian Public and Private Sectors*. In *Electronic Government and the Information Systems Perspective*, pages 356–369. Springer. (Cited on pages 13 and 22.)

Tauber, Arne, Bernd Zwattendorfer, and Thomas Zefferer [2011c]. *STORK: Pilot 4 - Towards Cross-border Electronic Delivery*. In *Joint Proceedings of Ongoing Research and Projects of IFIP EGOV and ePart 2011*, pages 295–301. Springer. (Cited on pages 13, 22 and 24.)

Tauber, Arne, Bernd Zwattendorfer, Thomas Zefferer, Yasmin Mazhari, and Eleftherios Chamakiotis [2010]. *Towards Interoperability: An Architecture for Pan-European eID-based Authentication Services*. In *Proceedings of the International Conference on Electronic Government and the Information Systems Perspective (EGOVIS)*, *Lecture Notes in Computer Science*, volume 6267/2010, pages 120–133. Springer. (Cited on page 12.)

Tauber, Arne, Bernd Zwattendorfer, Thomas Zefferer, and Klaus Stranacher [2011d]. *Grenzüberschreitendes E-Government in Europa*. *eGovernment Review*, 8, pages 8–9. (Cited on page 13.)

Teufl, Peter, Daniel Hein, Alexander Marsalek, Thomas Zefferer, and Alexander Oprisnik [2014a]. *Android Encryption Systems*. In *International Conference on Privacy & Security in Mobile Systems 2014*, pages 1–8. IEEE. (Cited on pages 11, 58, 107 and 237.)

Teufl, Peter, Thomas Zefferer, Sandra Kreuzhuber, and Christian Lesjak [2012]. *Trusted Location Based Services*. In *International Conference for Internet Technology And Secured Transactions 2012*, pages 185–192. IEEE. (Cited on pages 10, 79 and 80.)

Teufl, Peter, Thomas Zefferer, and Christof Stromberger [2013a]. *Mobile Device Encryption Systems*. In *28th IFIP TC-11 SEC 2013 International Information Security and Privacy Conference*, pages 203–216. Springer. (Cited on page 10.)

Teufl, Peter, Thomas Zefferer, Christof Stromberger, and Christoph Hechenblaikner [2013b]. *iOS Encryption Systems - Deploying iOS Devices in Security-Critical Environments*. In *SECRYPT 2013 - Proceedings of the 10th International Conference on Security and Cryptography*, pages 170–182. SciTePress. (Cited on pages 11, 60, 107, 112, 118 and 237.)

Teufl, Peter, Thomas Zefferer, Christoph Wörgötter, Alexander Oprisnik, and Daniel Hein [2014b]. *Android - On-Device Detection of SMS Catchers and Sniffers*. In *International Conference on Privacy & Security in Mobile Systems*, pages 1–8. IEEE. (Cited on pages 11, 66, 67 and 68.)

The European Parliament and the Council of the European Union [1999]. *DIRECTIVE 1999/93/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 13 December 1999 on a Community framework for electronic signatures*. `http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2000:013:0012:0020:EN:PDF`. (Cited on pages 25, 45, 92, 93, 94, 96, 145 and 146.)

The European Parliament and the Council of the European Union [2014]. *REGULATION (EU) No 910/2014 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 23 July 2014 on electronic identification and trust services for electronic transactions in the internal market and repealing Directive 1999/93/EC*. `http://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32014R0910&from=EN`. (Cited on pages 23, 25, 28, 45, 93, 94, 145 and 146.)

UK Cabinet Office [2012]. *Government Digital Strategy*. https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/296336/Government_Digital_Stratetegy_-_November_2012.pdf. (Cited on page 46.)

Windley, Phillip J. [2005]. *Digital Identity*. O'Reilly Media. ISBN 978-0596008789. (Cited on page 23.)

World Bank [2014]. *eGovernment*. http://web.worldbank.org. (Cited on page 20.)

World Health Organization [2014]. *General Information on Counterfeit Medicines*. http://www.who.int/medicines/services/counterfeit/overview/en/. (Cited on page 38.)

Yoojung, Kim, Yoon Jongsoo, Park Seungbong, and Han Jaemin [2004]. *Architecture for Implementing the Mobile Government Services in Korea.* In *Conceptual Modeling for Advanced Application Domains. ER 2004 Workshops CoMoGIS, CoMWIM ECDM, CoMoA, DGOV, and eCOMO. Proceedings.*, pages 601–612. ISSN 03029743. (Cited on page 34.)

Yubico [2015]. *YubiKey Hardware*. https://www.yubico.com/products/yubikey-hardware/. (Cited on page 230.)

Zálešák, Michal [2003]. *M-Government: More than a Mobilized Government.* http://www.developmentgateway.org/download/218309/mGov.doc. (Cited on page 37.)

Zefferer, Thomas [2011]. *Mobile Government: Stocktaking of Current Trends and Initiatives*. Technical Report, Secure Information Technologies Center - Austria. http://www.a-sit.at/pdfs/Technologiebeobachtung/mobile_government_1.0.pdf. (Cited on pages 33 and 37.)

Zefferer, Thomas [2013]. *A Survey and Analysis of NFC-based Payment Solutions for Smartphones*. In *IADIS International Conference e-Society 2013*, pages 275–282. IADIS. (Cited on pages 10 and 230.)

Zefferer, Thomas [2014]. *A Server-Based Signature Solution for Mobile Devices*. In *The 12th International Conference on Advances in Mobile Computing and Multimedia*, pages 175–184. ACM. (Cited on page 9.)

Zefferer, Thomas [2015a]. *Towards a New Generation of Mobile Government*. In *13th International Conference on e-Society (ES 2015)*, pages 191–198. IADIS. (Cited on pages 9 and 37.)

Zefferer, Thomas [2015b]. *Towards Transactional Electronic Services on Mobile End-User Devices - A Sustainable Architecture for Mobile Signature Solutions*. In *11th International Conference on Web Information Systems and Technologies*, pages 586–597. SciTePress. (Cited on pages 9 and 93.)

Zefferer, Thomas, Fabian Golser, and Thomas Lenz [2013a]. *Towards Mobile Government: Verification of Electronic Signatures on Smartphones.* In *Technology-Enabled Innovation for Democracy, Government and Governance*, *Lecture Notes in Computer Science*, volume 8061, pages 140–151. Springer. ISBN 978-3-642-40159-6. (Cited on pages 9 and 86.)

Zefferer, Thomas and Thomas Knall [2010]. *An Electronic-Signature Based Circular Resolution Database System*. In *Proceedings of the 25th Annual ACM Symposium on Applied Computing 2010*, pages 1840–1845. ACM. (Cited on page 12.)

Zefferer, Thomas, Sandra Kreuzhuber, and Peter Teufl [2013b]. *Assessing the Suitability of Current Smartphone Platforms for Mobile Government*. In *Technology-Enabled Innovation for Democracy, Government and Governance*, pages 125–139. Springer. (Cited on pages 8, 54, 107, 118, 128, 129, 130 and 213.)

Zefferer, Thomas and Vesna Krnjic [2012a]. *Towards User-friendly E-Government Solutions: Usability Evaluation of Austrian Smart-Card Integration Techniques*. In *Advancing Democracy, Government and Governance*, pages 88–102. Lecture Notes in Computer Science, Springer. (Cited on pages 14 and 27.)

Zefferer, Thomas and Vesna Krnjic [2012b]. *Usability Evaluation of Electronic Signature Based E-Government Solutions*. In *Proceedings of the IADIS International Conference WWW/INTERNET 2012*, pages 227–234. IADIS. (Cited on pages 14, 47, 136 and 139.)

Zefferer, Thomas and Vesna Krnjic [2012c]. *Usability-Evaluierung der österreichischen Handy-Signatur*. (Cited on page 14.)

Zefferer, Thomas, Vesna Krnjic, Klaus Stranacher, and Bernd Zwattendorfer [2014a]. *Measuring Usability to Improve the Efficiency of Electronic Signature-based E-Government Solutions*, pages 45–74. Springer. (Cited on page 16.)

Zefferer, Thomas, Vesna Krnjic, and Bernd Zwattendorfer [2011a]. *Ein virtuelles Testframework für E-Government Komponenten*. In *D-A-CH Security 2011*, pages 492–503. syssec. (Cited on page 12.)

Zefferer, Thomas, Arne Tauber, and Bernd Zwattendorfer [2012a]. *Improving the Security of SMS-based Services using Electronic Signatures - Towards SMS-based Processing of Transactional m-Government Services.* In *WEBIST 2012 - Proceedings of the 8th International Conference on Web Information Systems and Technologies*, pages 743–752. SciTePress. ISBN 978-989-8565-08-2. (Cited on pages 9, 76 and 79.)

Zefferer, Thomas, Arne Tauber, and Bernd Zwattendorfer [2013c]. *Harnessing Electronic Signatures to Improve the Security of SMS-based Services*, pages 331–346. Lecture Notes in Business Information Processing, Springer. (Cited on page 10.)

Zefferer, Thomas, Arne Tauber, Bernd Zwattendorfer, and Thomas Knall [2011b]. *Secure and Reliable Online-Verification of Electronic Signatures in the Digital Age*. In *Proceedings of the IADIS International Conference WWW/INTERNET 2011*, pages 269–276. IADIS. (Cited on pages 12 and 84.)

Zefferer, Thomas, Arne Tauber, Bernd Zwattendorfer, and Klaus Stranacher [2012b]. *Qualified PDF Signatures on Mobile Phones*. In *Electronic Government and Electronic Participation - Joint Proceedings of Ongoing Research and Projects of IFIP EGOV and IFIP ePart 2012*, *Informatik*, volume 39, pages 115–123. Trauner. (Cited on pages 8, 81 and 82.)

Zefferer, Thomas and Peter Teufl [2011]. *Opportunities and Forthcoming Challenges of Smartphone-based m-Government Services. European Journal of ePractice*, 13. (Cited on pages 8, 33, 35 and 36.)

Zefferer, Thomas and Peter Teufl [2013]. *Policy-based Security Assessment of Mobile End-User Devices*. In *Proceedings of the 10th International Conference on Security and Cryptography 2013*, pages 347–354. SciTePress. (Cited on pages 10, 72, 75, 118 and 128.)

Zefferer, Thomas and Peter Teufl [in press]. *Leveraging the Adoption of Mobile eID and e-Signature Solutions in Europe*. In *4th International Conference on Electronic Government and the Information Systems Perspective*. Springer. (Cited on pages 9 and 47.)

Zefferer, Thomas, Peter Teufl, David Derler, Klaus Potzmader, Alexander Oprisnik, Hubert Gasparitz, and Andrea Hoeller [2013d]. *Power Consumption-based Application Classification and Malware Detection on Android Using Machine-Learning Techniques*. In *FUTURE COMPUTING 2013, The Fifth International Conference on Future Computational Technologies and Applications*, pages 26–31. IARIA. (Cited on pages 10, 68 and 71.)

Zefferer, Thomas, Peter Teufl, David Derler, Klaus Potzmader, Alexander Oprisnik, Hubert Gasparitz, and Andrea Höller [2014b]. *Towards Secure Mobile Computing: Employing Power-Consumption Information to Detect Malware on Mobile Devices*. (Cited on page 11.)

Zefferer, Thomas, Peter Teufl, and Herbert Leitold [2011c]. *Mobile qualifizierte Signaturen in Europa*. *Datenschutz und Datensicherheit - DuD*, 35, pages 768–773. (Cited on page 8.)

Zefferer, Thomas and Bernd Zwattendorfer [2014]. *An Implementation-Independent Evaluation Model for Server-Based Signature Solutions*. In *10th International Conference on Web Information Systems and Technologies*, pages 302–309. SciTePress. (Cited on page 9.)

Zhang, Lide, Birjodh Tiwana, Robert P. Dick, Zhiyun Qian, Z. Morley Mao, Zhaoguang Wang, and Lei Yang [2010]. *Accurate Online Power Estimation and Automatic Battery Behavior Based Power Model Generation for Smartphones*. *2010 IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, pages 105–114. doi:10.1145/1878961.1878982. (Cited on page 69.)

Zhou, Yajin and Xuxian Jiang [2012]. *Dissecting Android Malware: Characterization and Evolution*. In *Proceedings - IEEE Symposium on Security and Privacy*, pages 95–109. ISBN 9780769546810. ISSN 10816011. doi:10.1109/SP.2012.16. (Cited on page 65.)

Ziegler, Chris [2011]. *Nokia CEO Stephen Elop Rallies Troops in Brutally Honest 'Burning Platform' Memo?* `http://www.engadget.com/2011/02/08/nokia-ceo-stephen-elop-rallies-troops-in-brutally-honest-burnin/`. (Cited on page 3.)

Zwattendorfer, Bernd, Thomas Zefferer, and Klaus Stranacher [2014]. *An Overview of Cloud Identity Management-Models*. In *Proceedings of the 10th International Conference on Web Information Systems and Technologies*, pages 82–92. SciTePress. (Cited on page 16.)

Zwattendorfer, Bernd, Thomas Zefferer, and Klaus Stranacher [in press]. *A Comparative Survey of Cloud Identity Management-Models*. In *WEBIST 2014 - Selected and Revised Papers*. Springer. (Cited on page 16.)

Zwattendorfer, Bernd, Thomas Zefferer, and Arne Tauber [2011a]. *A Privacy-Preserving eID based Single Sign-On Solution*. In *Proceedings of 5th International Conference on Network and System Security (NSS 2011)*, pages 295–299. Springer. (Cited on page 13.)

Zwattendorfer, Bernd, Thomas Zefferer, and Arne Tauber [2011b]. *E-ID Meets E-Health on a Pan-European Level*. In *Proceedings of the IADIS International Conference e-Health 2011*, pages 97–104. IADIS. (Cited on page 12.)

Zwattendorfer, Bernd, Thomas Zefferer, and Arne Tauber [2012]. *The Prevalence of SAML within the European Union*. In *8th International Conference on Web Information Systems and Technologies*, pages 571–576. SciTePress. (Cited on page 14.)