Mobile Authentication with German eID

Florian Otterbein[1], Tim Ohlendorf[2], and Marian Margraf[1]

[1] Freie Universität Berlin, 14195 Berlin, Germany,
[2] Technische Universität Darmstadt, 64289 Darmstadt, Germany

Abstract. Due to the rapid increase of digitization within our society, digital identities gain more and more importance. Provided by the German eID solution, every citizen has the ability to identify himself against various governmental and private organizations with the help of his personal electronic ID card and a corresponding card reader. While there are several solutions available for desktop use of the eID infrastructure, mobile approaches have to be payed more attention. In this paper we present a new approach for using the German eID concept on an Android device without the need of the actual identity card or card reader. A security evaluation of our approach following the ISO 27001 procedure model reveals no harmful attacks on the architecture.

1 Introduction

Germany introduced an electronic ID card (eID) on November, 1st 2010. In combination with a desktop client and an RFID card reader, it enables citizens and legal residents to identify themselves online with private and governmental organizations (eID offerer). However, a consumer research from 2015 determined that only about 5% of all Germans used their eID for online authentication services within the past 12 months [10]. We can think of two important reasons. First, only few services with eID support are available on the market. Thus, users may not see a significant benefit in using eID. Second, ID card holders may not want to spend money for the required card reader (currently roughly 25 to 159 euros). In this paper, we present an eID concept that eliminates the requirement of a card reader by relying on the user's smartphone instead of an ID card. During an initial setup phase, an eID token will be installed on the users smartphone. Online authentication services can then be used without the user's ID card. This could increase the acceptance rate of eID among citizens and legal residents.

This paper is structured as follows. First, we present related work and an outline in Sec. 2. In Sec. 3 we will introduce the technologies that are used for our architecture in Sec. 4. In Sec. 5, the results of our security evaluation are presented.

2 Related Work

Existing projects provide a mobile application for the German eID by using the internal NFC controller, an external Bluetooth card reader, or an USB card reader [14, 20]. Besides the lack of usability by using an external device during the authentication process, Android's NFC stack is unable to handle extended APDUs [13]. As the eID process uses extended APDUs for communication, it is incompatible with Android smartphones. Additionally, existing eID applications only initiate the authentication process and act as a management component. All security relevant operations of the process are performed by the smart card chip inside the ID card.

Schröder developed and evaluated a concept that enables storing and using derived identities on a smartphone securely [18]. He demonstrated his approach in a proof-of-concept with the German eID and a corresponding wireless card reader.

Other countries in the European Union also provide mobile authentication solutions for their citizens. In Austria for example one can register for the so called "Handy-Signatur" [1] which is a mobile signature system offered by the Graz University of Technology, the Austrian Government and Austrian National Bank. This project, initialized 2009, first used SMS-TANs for authentication. Today a corresponding smartphone app can be used instead. Another solution for mobile authentication by the Austrian state printing office is "My Identity App" (MIA) [15]. MIA saves all governmental identification documents, like passport, drivers license, etc. centralized in a digital format. Both Austrian approaches have in common, that they require a cloud backed infrastructure. All sensitive personal information is processed and stored on the providers servers. This stands in contrast to our following approach were the sensitive data is never revealed to a third-party.

Apple Pay, a mobile payment service, serves as an example for a successful usage of digital identities.

3 Technical Background

Saving a sovereign identity on a mobile device leads to new security requirements. The storage and transmission of an identity to the eID offerer has to be protected with a similar security level as on eID cards, that use a secure element. Thus we rely on the secure element and trusted execution environment of an Android phone that are described in this section. Also a short overview of the underlying eID process is given.

3.1 Online Authentication (eID process)

The German eID system enables the owner of a German ID card (eID) to identify himself against various service providers (eID offerer) in the governmental and private sector. Therefore all components involved in the eID process have to

fulfill certain requirements described in the technical guidelines, published by the German Federal Office for Information Security (BSI) [5] [6]. The security and privacy aspects of the system were addressed in various papers and articles [3] [4].

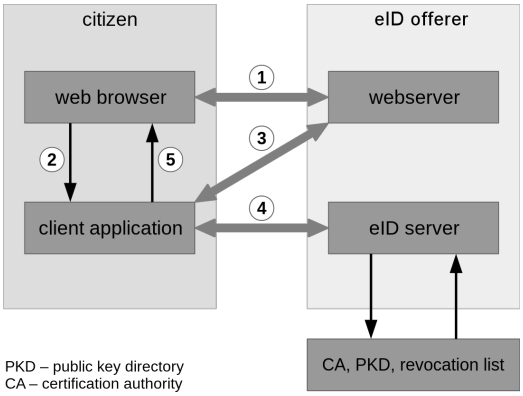


Fig. 1. eID process between citizen and eID offerer

Fig. 1 shows a schematic representation of an eID process which will be explained in the following:

1. The user wants to get his identity authenticated by an eID offerer. Therefore he clicks on an special hyperlink (TcTokenURL) on the website of the eID offerer.
2. The user's browser forwards the URL to the eID client application.
3. The client software recognizes the TcTokenURL and fetches the request (TcToken) about the required personal data of the user (name, surname, date of birth, ...) and the responsible eID server for the current eID process.
4. The eID client establishes a secure connection to the eID server and starts the authentication process between the eID card and the eID server. Finally the id card sends the required personal data to the eID server.
5. The eID offerer receives the needed authentication data of the user from the eID server. The user will now be redirected to an authenticated web session with the eID offerer.

To ensure a secure connection between eID card and eID server the EAC (Extended Access Control) protocol [5] is used. It authenticates the two involved participants against each other and is divided in two subprotocols called Terminal Authentication (TA) and Chip Authentication (CA). TA guarantees the eID card a communication with a valid and authentic eID server. CA guarantees the eID server to communicate with a valid and authentic eID card. To provide this, Public-Key-Infrastructures are used (Country Verifying Certificate Authority (CVCA) and Country Signing Certificate Authority (CSCA)) with

CVCA and CSCA certificates as root instances. The underlying cryptographic primitive is a DiffieHellman key exchange, the public keys of both parties are authenticated using signature algorithms.

3.2 Secure Element

A secure element (SE) is a tamper-resistant microcontroller that provides a secure environment for applications and data [18]. Its computing power and storage capacity is limited and can be found on the Universal Integrated Circuit Card (UICC), SD card or on a non-replaceable chip embedded in the smartphone (see Fig. 2). The secure element is divided into security domains that can be programmed using Java Card applets[3].

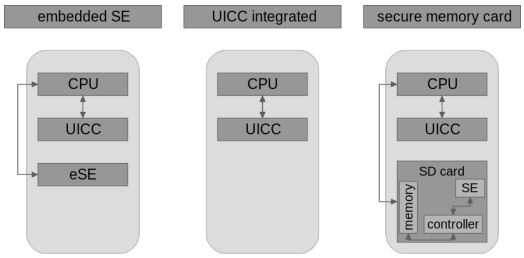


Fig. 2. SE form factor overview on mobile devices

A Trusted Service Manager (TSM) is the link between the secure element manufacturer (issuer) and the application developer (service provider). The TSM is responsible for provisioning and personalizing the secure element. Without a TSM, the service provider must have a contract with every issuer.

3.3 Trusted Execution Environment

A Trusted Execution Environment (TEE) enables the execution of security-critical applications inside a runtime environment separated from the rest of the system [8]. This isolated environment can be implemented as an external coprocessor or as different runtime modes on the main CPU. Many microprocessors on the today's market (ARM TrustZone, TI M-Shield, Intel SGX) feature a TEE. Focusing on mobile systems, ARM CPUs are currently most relevant for this consideration. In ARMs TEE implementation TrustZone [2], trustlets are security-critical applications that can be executed in an isolated environment (secure world), while the actual Rich OS is running in a non-isolated environment (normal world), see Fig. 3. Both worlds have their own virtual allocated storage area on the physical RAM and in the registers of the CPU. Communication between the worlds is managed by a monitor, located in the secure world and waiting for requests (Secure Monitor Calls) from the normal world.

[3] http://www.oracle.com/technetwork/java/javacard. Accessed: 11 Apr 2016.

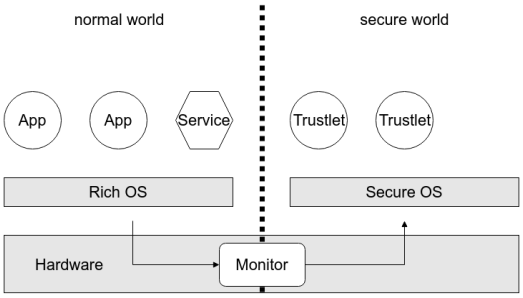


Fig. 3. Schematic representation of the ARM TrustZone architecture

At the time of writing, third-party developers who want to make use of the TEE are locked out. There are currently no official mechanisms provided for a third-party developer to install a custom trustlet. However the GlobalPlatform association is currently working on standards for implementing and managing TEEs and enabling third-party developers the access for their applications.

4 Design

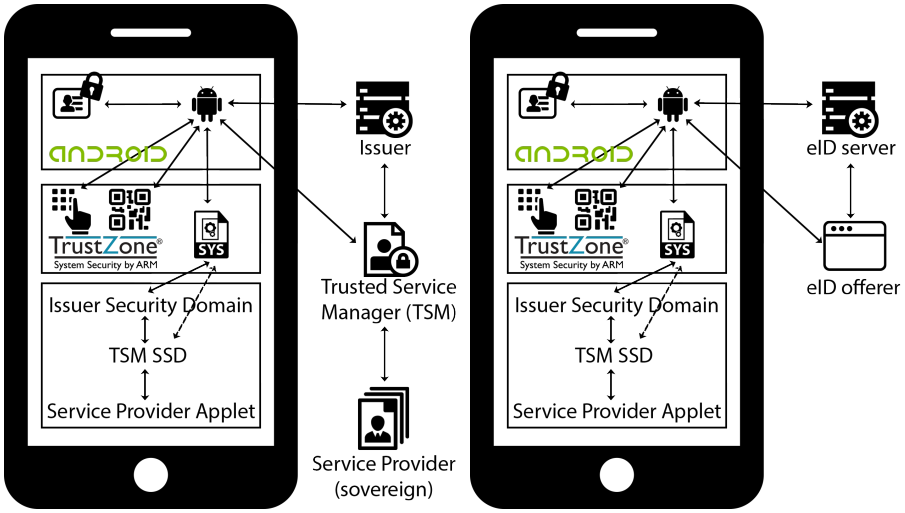
4.1 Initialization

In this section the architecture and installation process is described. Issuer, TSM and Service Provider each have a local instance (security domain/applet) and a remote instance (external server). During the smartphone's manufacturing process, only the security domain of the issuer is installed. Figure 4 shows all instances and their connections.

Initial setup. The user installs the eID app from Google's Play Store. The app checks if a TEE and a secure element is available and then installs the trustlet for the TEE.

Installing TSM's security domain. The Android app sends a request for installing a new security domain[4] to TSM's remote instance. The TSM forwards the command to the remote instance of the issuer. Local and remote instance of the issuer authenticate each other by using a challenge-response protocol. The issuer's remote instance sends an encrypted installation command to his local instance and sets new access keys for the TSM. The issuer sends the new keys to the TSM, who can directly connect to his new local instance for setting his own keys.

[4] also called supplementary security domains (SSD)



(a) Actors and communication channels during initialization process

(b) Actors and communication channels during authentication process

Fig. 4. Mobile eID architecture with Service Provider, TSM, Issuer, eID server, eID offerer, TEE, and secure element. Inside the TEE (e.g. ARMs TrustZone), users can enter secret information. The TEE also works as a bridge between the non-isolated environment and the secure element. Inside the secure element, isolated security domains and applets are implemented that are managed by their external instances. Dashed lines show that this communication channel can only be used for setting new encryption and authentication keys, based on GlobalPlatform's definition [11].

Installing the applet. The implementation of protocols and installation of certificates is needed to support the eID process. Thus an applet for the Android app and the root certificates CVCA and CSCA have to be installed. The TSM's remote instance sends an encrypted and signed applet to the issuer, whose local instance is responsible for installing the applet. TSM's local instance has to verify the applet to release the installation process. This verification process is called DAP-Verification [11]. Afterwards the service provider has its own non-personalized applet inside the security domain.

Personalize security domain. To protect the Android app against abuse, an access PIN is needed, which is defined by the user during the personalization process. A key exchange is used to establish a secure connection between the service provider and its applet. As Android's NFC stack is incompatible with German eID (see Sec. 2), we use the smartphone camera for capturing the ID card. This process is legally compliant with the German Money Laundering Act [7] and is already practiced by some German-based companies.

Beside the traditional PIN of the ID card, a personal private key (represented by a QR-Code) has to be added to the postal letter of the sovereign service provider, that is sent to the citizen after ordering an ID card.

By capturing the ID card and this key, the mobile eID has the same security level as the traditional eID process (see Sec. 5). The external service provider receives the captured data and checks its integrity. If the validation process is successful, the service provider sends back the eID token containing all personalized information, concatenated with the private key of the Chip Authentication (see Sec. 3.1) to the security domain. The data package is encrypted with the public key of the QR-Code. The applet can decrypt the token, encrypt it with a symmetric key and store the data on Android's file system. This is necessary as the smart card only provides little storage capacity. The private key for the Chip Authentication remains inside the secure element.

Authentication process After the installation process, the eID token is installed and the ID card is not needed anymore. The eID token's decryption key can only be accessed by entering the PIN into the TEE. After retrieving the key, the secure element in the smartphone can act like the secure element in the ID card.

4.2 Authentication

The authentication protocols (Sec. 3.1) used by the eID app are determined in TR-03110. Some modifications regarding the access policy are necessary. The communication channels and all actors are shown in Fig. 4.

1. The user clicks on the eID offerer's link. A secure connection between Android app and eID server is established.
2. By entering the PIN into the TEE, the access for the user to the secure element is granted.
3. The terminal authentication between secure element and eID server is carried out[5].
4. The chip authentication between secure element and eID server is carried out.
5. The encrypted token in the untrusted system storage is received by the secure element and gets decrypted by the key stored inside the secure element.
6. The user determines what kind of personal data inside the TEE should be redirected to the eID server.
7. Secure element and eID server can directly communicate using encrypted and authenticated Application Protocol Data Unit (APDU) commands.
8. Once the authentication process is finished, the access to the secure element is locked again.

5 Security Evaluation

The security evaluation follows a similar approach to the standardized procedure model ISO 27001 [12]. Therefore, the evaluation will be divided into following steps:

[5] The Java Card algorithm `ALG_EC_SVDP_DH_PLAIN` has to be supported by the secure element.

1. Initialization: Description of business processes and security objectives
2. Structural Analysis: Description of actors, components and communication channels
3. Security Requirements: Definition of assets that need to be protected and analysis of their protective needs
4. Modeling of IT Architecture: Summary of actors and components
5. Description of Protective Measures: Protective measures are established to ensure the security of assets that need to be protected. Specific security functions are used to implement the protective measures.
6. Evaluation of Protective Measures

5.1 Initialization

We consider the use cases *initialization* and *authentication*. A description of these use cases can be found in Sec. 4.1 and 4.2, respectively.

We will distinguish between the security objectives confidentiality, authenticity and availability:

- *Confidentiality:* Protecting data from disclosure to unauthorized parties
- *Authenticity:* Ensure, that data was not sent by unauthorized parties (includes integrity).
- *Availability:* Ensure, that data can be accessed by authorized parties when needed.

5.2 Structural Analysis

All actors and communication channels are described in Sec. 4. We assume, that the existing protocols for the authentication process are secure. Security evaluations regarding the communication between smartphone, eID offerer and eID server already exist [17]. We do not need to evaluate the communications between two external instances for the same reason. Therefore, it is sufficient to investigate the following components, that are necessary to get the personalized token on the smartphone at the initial setup (use case initialization) and using the personalized token for authentication (use case authentication):

- Smartphone including all components and their communication
  - Android app
  - Internal storage of Android app
  - Host CPU
  - TEE
  - Secure element
- Issuer
- TSM
- Service Provider
- Communication between Android app and security domains (over TEE)
- Communication between Android app and issuer
- Communication between Android app and TSM

5.3 Security Requirements

Due to the high protection needs of the business process "authentication" (see Sec. 4), the asset "eID token" has high protection needs and thereby the smartphone and the components being in contact with this asset as well. Furthermore, we can determine the protection needs by analyzing the asset "eID token" regarding their specific security objectives:

- *Confidentiality* High protection needs regarding confidentiality, because the eID token contains personalized data. By having access to all data stored inside the token, an attacker could harm the user.
- *Authenticity* High protection needs regarding authenticity, because the business process aims to authenticate the user with the help of the eID token on different services. For examples, using the eID token for opening a bank account requires a high security level.
- *Availability* Normal protection needs regarding availability. If an authorized user isn't able to access the eID token when it's needed, he can't authenticate online and loses comfort.

It can be concluded, that an attacker can cause high damage by manipulating or having access to another eID token. In combination with the high spread of Android malware [9] and thereby a high probability of being attacked, a high risk exists.

5.4 Modeling of IT Architecture

As a summary of Sec. 4, Fig. 4 shows all actors and components during the initialization and authentication process.

5.5 Description of Protective Measures

In this section we will describe the protective measures that are necessary to fulfill the security requirements mentioned in Sec. 5.3.

Keys of issuer Issuer keys are needed to ensure a confidential and authentic communication between issuer and its security domain. During the manufacturing process, the symmetric keys `S-ENC` and `S-MAC` are stored inside the issuer's security domain. The keys don't leave the secure element. The issuer and its security domain calculate a session encryption key by encrypting a 16 Byte random number with the key `S-ENC`. To ensure the authentication of messages sent between issuer and its security domain, the cryptographic algorithm `HMAC` with the key `S-MAC` is used.

Keys of TSM After the issuer installs a security domain for the TSM, the access keys of the security domain are transferred from the issuer to the TSM. The TSM can now access his new security domain over an confidential and authentic communication channel. The GlobalPlatform specification [11] allows a connection without the help of the issuer (see dotted lines in Fig. 4) if the TSM sets new keys. Thereby the issuer can't manipulate or tap further communication between TSM and its security domain.

Access PIN for Android application Only the smartphone owner should use his credentials stored inside the smartphone to authenticate against an eID offerer. Therefore the OwnerPIN API[6] is used to access the secure element. The user enters his PIN inside the TEE to ensure, that no attacker can tap his inputs. The PIN is evaluated inside the secure element by the internal Java Card applet of the service provider.

Personal private key (QR code) Nobody should be allowed to store the eID token of another person on his smartphone. A two-factor authentication is used during the initialization process to ensure only the ID owner is able to use the eID token. An additional QR code, which represents a private key, is printed on the letter received by the ID owner. The associated public key is used by the service provider to encrypt the eID token. The user scans his QR code inside the TEE. To decrypt the received eID token, the personal private key is necessary.

eID token's symmetric encryption/decryption key The limited resources of the secure element prevent the persistent storage of the eID token. Therefore the token has to be stored on Android's file system. A symmetric key is generated and stored on the secure element to encrypt and decrypt the eID token.

5.6 Evaluation of Protective Measures

The evaluation of the used cryptographic algorithms and physical attacks against the secure element are excluded. It's assumed that the TSM is trustworthy, the attacker has limited resources and no implementation errors have been made. All commands that are executed on the Host CPU of the Android smartphone are particularly critical, because of high attack potential.

In [16], a detailed description of attacks against the security objectives has been carried out for all the components and communication channels. As a result, two vulnerabilities have been found:

Sniffing SELECT commands The SELECT command is responsible for addressing the correct security domain inside the secure element. The GlobalPlatform organization is responsible for defining secure element standards. In their definition about the communication, it is said, that no encryption of the SELECT

[6] https://www.win.tue.nl/pinpasjc/docs/apis/jc222/javacard/framework/OwnerPIN.html. Accessed: 15 Jul 2016.

command is possible. Thus it is possible to obtain information about the identification number of the security domain. No harmful attacks can be executed with this information, because the security domain can only be accessed by the eID Android application and with an additional PIN.

Relay attack During the installation of the security domain from the TSM or service provider, an attacker could redirect the commands between Android app and the secure element. If a user compares the installation process of the security element in a TEE with the installation process in a rich execution environment, it would be possible to detect a successful attack in the past. No sensitive information is compromised with this attack. During the personalization process, the transferred token can also be redirected. Due to cryptographic mechanisms, attackers can't decrypt the sensitive information.

6 Conclusion and Future Work

In this paper we demonstrated, that hardware-based security solutions are needed to protect sensitive data. Smartphone owners with custom flashed kernels and root privileges are able to obtain sensitive information, when only software-based security mechanisms are used. The security evaluation showed, that no harmful attacks could be executed, which means that neither the eID token nor the cryptographic keys were compromised.

It's possible to transfer the described concept to eID solutions of other countries as long as a sovereign service provider managing the eID tokens is available and the processes are compliant with the law.

Nevertheless, it is complicated to enroll the concept to Android because of the heterogeneous devices: Each device needs a secure element and a trusted execution environment which is able to understand the APDU commands sent by the service provider.

In the future it is planned to reduce the complexity of the described concept by using deduced identities. Schröder et al. [19] discussed an approach by using the existing eID infrastructure combined with the U-Prove technology to supply trustworthy deduced identities.

References

1. A-Trust GmbH Handy-Signatur, Der digitale Ausweis. https://www.handy-signatur.at/hs2/. Accessed 24 May 2016
2. ARM Limited: ARM Security Technology - Building a Secure System using TrustZone Technology (2009). http://infocenter.arm.com/help/topic/com.arm.doc.prd29-genc-009492c/PRD29-GENC-009492C_trustzone_security_whitepaper.pdf. Accessed 24 Apr 2016.
3. Bender, J., Dagdelen, Ö., Fischlin, M. and Kügler, D.: "Security analysis of the pace key-agreement protocol." International Conference on Information Security. Springer Berlin Heidelberg (2009).

4. Bender, J., Dagdelen, Ö., Fischlin, M., and Kügler, D.: "The PACE— AA protocol for machine readable travel documents, and its security." International Conference on Financial Cryptography and Data Security. Springer Berlin Heidelberg (2012).
5. Bundesamt für Sicherheit in der Informationstechnik: Advanced Security Mechanisms for Machine Readable Travel Documents and eIDAS Token (2015). https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TR03110/BSI_TR-03110_Part-2-V2_2.pdf. Accessed 28 Jun 2016.
6. Bundesamt für Sicherheit in der Informationstechnik: Technische Richtlinie 03112 - Das eCard-API-Framework (2014). https://www.bsi.bund.de/DE/Publikationen/TechnischeRichtlinien/tr03112/index_htm.html. Accessed 28 Jun 2016.
7. Bundesanstalt für Finanzdienstleistungsaufsicht: Auslegung des §6 Abs. 2 Nr. 2 GwG. In: Rundschreiben 1/2014 (GW) - Verdachtsmeldung nach §§11, 14 GwG (2014).
8. Cooijmans, T.: Secure Key Storage and Secure Computation in Android. Radboud University Nijmegen (2014).
9. GData: Mobile Malware Report (2015). https://file.gdatasoftware.com/web/de/documents/whitepaper/G_DATA_Mobile_Malware_Report_Jan-Mar_2015_German.pdf. Accessed 8 Jul 2016
10. GfK SE (2015) Fünf Prozent nutzen elektronischen Personalausweis. http://www.gfk.com/insights/news/fuenf-prozent-nutzen-elektronischen-personalausweis. Accessed: 21 Apr 2016.
11. GlobalPlatform: GlobalPlatform Card Specification Version 2.2.1 Public Release (2011). http://www.globalplatform.org/specificationscard.asp. Accessed 21 Apr 2016.
12. International Standard Organisation: ISO/IEC 27000:2014 (2014). http://www.iso.org/iso/catalogue_detail?csnumber=63411. Accessed 21 Apr 2016.
13. International Standard Organisation: ISO/IEC 7816-4:2013 (2013). http://www.iso.org/iso/catalogue_detail.htm?csnumber=54550. Accessed 21 Apr 2016.
14. Open eCard Team: Open eCard - Überblick (2013). https://www.openecard.org/ecard-api-framework/ueberblick. Accessed 21 Apr 2016.
15. Österreichische Staatsdruckerei MIA - My Identity App. https://www.mia.at/. Accessed 24 May 2016
16. Otterbein, F.: Konzeption und Sicherheitsevaluierung einer mobilen Online-Ausweisfunktion. Masterthesis (2015).
17. Ritscher, M.: Sicherheitskonzept für PersoApp auf Android OS (2014). http://www.persoapp.de/wp-content/uploads/2015/02/D11-Sicherheitskonzept-fuer-PersoApp-auf-Android-OS.pdf. Accessed 1 Apr 2016.
18. Schröder, M.: Sichere Bereitstellung von Identitätstoken auf mobilen Endgeräten. Humboldt-Universität zu Berlin, Mathematisch-naturwissenschaftliche Fakultät II, Institut für Informatik (2013).
19. Schröder, M. and Morgner, F.: eID mit abgeleiteten Identitäten. Datenschutz und Datensicherheit-DuD, Volume 37, Issue 8 pp. 530-534, Springer (2013).
20. Kubieziek J., Kahlo C.: D06-QM Architekturkonzept der Open-Source-Core. PersoApp Projektdokumentation (2013)