
# Requirements Report

## Deliverable D 35.1

| Document Identification | |
|---|---|
| **Date** | 26/04/2013 |
| **Status** | Final |
| **Version** | 1.1 |

| | | | |
|---|---|---|---|
| **Related SP / WP** | SP 3/ WP 3.5 | **Document Reference** | D35.1 |
| **Related Deliverable(s)** | | **Dissemination Level** | Confidential |
| **Lead Participant** | Giesecke & Devrient | **Lead Author** | Dr. F.-M. Kamm (G&D) |
| **Contributors** | IFAG, TUG,TUD, IBM, G&D | **Reviewers** | Eray Özmü (USTUTT)<br>Detlef Hünlein (ECS) |

This document is issued within the frame and for the purpose of the FutureID project. This project has received funding from the European Unions Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 318424

# 1. Abstract

The FutureID project works on building a comprehensive, flexible, privacy-aware and ubiquitously usable identity management infrastructure for Europe which integrates existing eID technology and trusted infrastructures. As the client is the main interface between the user and a secure identity token, it needs to handle secret cryptographic information with an appropriate security level. As long as no end-to-end security channel is established between the secure token and the backend servers, the client platform may become a weak link in the whole infrastructure if it cannot reach a sufficient trust and security level. Therefore it is important to investigate on the requirements that a trustworthy client platform has to fulfil. This deliverable therefore discusses the relevant aspects of trustworthy client platforms and develops the requirements for FutureID.

One of the main questions in specifying the trustworthy properties of the client platform is the level of trust and security that the different credential systems already inherently provide by themselves and/or require from the surrounding platform. Based on these investigations, five trust and security levels are defined in section 6, taking the various boundary conditions into account. Existing eID systems are mapped into this scheme in the same section and relevant standards in this field are discussed.

Section 7 describes the typical components of a trustworthy platform and discusses their trustworthy properties, including privacy-preserving credentials. The actual requirements for each trust and security level are defined in section 8 for both stationary devices as well as mobile devices. General requirements resulting from privacy-preserving credentials are also defined in this section.

# 2. Document Information

## 2.1 Contributors

| Name | Partner |
|---|---|
| Dr. Frank-Michael Kamm | Giesecke & Devrient GmbH |
| Dr. Detlef Houdeau | Infineon Technologies AG |
| Prof. Dr. Peter Lipp | TU Graz |
| DI Christof Rath | TU Graz |
| Christian Wagner | TU Graz |
| Dr. Pouyan Sepehrdad | TU Darmstadt |
| Dr. Gregory Neven | IBM Research GmbH |

## 2.2 History

| Version | Date | Author | Changes |
|---|---|---|---|
| 0.1 | 01.02.2013 | Frank-Michael Kamm | Initial Version |
| 0.2 | 05.03.2013 | Gregory Neven | Initial list of trust levels |
| 0.3 | 19.03.2013 | Frank-Michael Kamm | G&D part |
| 0.4 | 20.03.2013 | Frank-Michael Kamm | Mobile requirements |
| 0.5 | 26.03.2013 | TUG/TUD | TUG/TUD part added |
| 0.6 | 27.03.2013 | Gregory Neven | IBM part integrated |
| 0.7 | 03.04.2013 | Detlef Houdeau | Infineon part added |
| 1.0 | 03.04.2013 | Frank-Michael Kamm | First complete version |
| 1.1 | 22.04.2013 | Frank-Michael Kamm | Reviewer comments integrated |

# 3. Table of Contents

# 4. Glossary/Definitions

| | |
|---|---|
| ABC | Attribute-based credential (privacy-preserving) |
| API | Application Programming Interface |
| ASLR | Address Space Layout Randomized |
| ASSD-Card | Advanced Security Secure Digital Card |
| CD | Compact Disc |
| CFI | Control Flow Integrity |
| DAA | Direct Anonymous Attention |
| GSM | Global System for Mobile Communication |
| HW | Hardware |
| IFD | Interface Device |
| LED | Light Emitted Diode |
| MAC | Message Authentication Code |
| MTM | Mobile Trusted Module |
| NFC | Near Field Communication |
| OASIS | Open Accessible Space Information System |
| OS | Operating System |
| OTA | Over The Air |
| PACE | Password Authentication Connection Establishment |
| PC | Personal Computer |
| PC/SC | Personal Computer / Smart Card |
| PIN | Password Identification Number |
| PKCS | Public Key Cryptography Standard |

| | |
|---|---|
| PKI | Public Key Infrastructure |
| ROM | Read Only Memory |
| SD-Card | Secure Digital Card |
| SIM | Subscriber Identification Module |
| SW | Software |
| SWP | Single Wire Protocol |
| TEE | Trusted Execution Environment |
| TMP | Trusted Platform Module |
| TNC | Trusted Network Connect |
| UICC | Universal Integrated Circuit Card |

**Definitions:**

**Platform** The platform comprises the hardware and software environment of the device on which the FutureID client is running. The device may be a stationary device (e.g. PC) or a mobile device (e.g. Smartphone, tablet). Any hardware-based secure elements, like a TPM, a SIM card or an embedded UICC and any software-based secure environments (like a Trusted Execution Environment) which are part of the device hardware or software belong to the platform. A separate hardware-based eID token (e.g. a smart card) is not considered to be part of the platform.

# 5. Introduction

The FutureID project works on building a comprehensive, flexible, privacy-aware and ubiquitously usable identity management infrastructure for Europe which integrates existing eID technology and trusted infrastructures. One key element of the FutureID infrastructure is the client platform, which connects the user and potentially external trusted hardware to the backend IT-network. Based on this central position of the client within the electronic identification framework, the client platform needs to provide sufficient trust and security for the user, the identity provider and the service provider depending on the respective requirements of each use case. The required trust level typically increases from a simple password-based identification process to the dedicated authentication work flow or an electronic signature application.

Typically two levels of client platforms are defined in running eID programs on the national level:

a) Client platform for identification and authentication of the user,

b) Client platform for electronic signature procedure.

For platform a) the name “class-1” was defined for the functionality, usability and security, if a PC/laptop is used. For platform b) the name “class-3” was defined, which captures all aspects of the class-1 and the device certification if a PC/laptop is used. For the electronic identification based on mobile phones, other classes should be defined and/or requested from the government side or related organizations. For class-1 platforms, a list of trustworthy devices is typically published by government-related institutions, including the manufacturer name and the device identity. For class-3 client devices a security certification along ISO 15408 is typically requested.

One of the major scopes of qualifications and security certifications of client platforms is it to avoid unintended functionality, such as backdoor functions, eave dropping and data skimming. Since the client is part of the backend network and therefore also part of the World Wide Web, the criteria of ICT-infrastructure security for components in the web apply also for this application. This includes all measures to avoid cyber attacks such as identity theft.

As the client is the main interface between the user and a secure identity token, it needs to handle secret cryptographic information with an appropriate security level. As long as no end-to-end security channel is established between the secure token and the backend servers, the client platform may become a weak link in the whole infrastructure if it cannot reach a sufficient trust and security level. Therefore it is important to investigate on the requirements that a trustworthy client platform has to fulfil. The report D35.1 discusses the relevant aspects of trustworthy client platforms and develops the requirements for FutureID.

# 6. Existing eID and Credential Systems, Trustworthy Properties

One of the main questions in specifying the trustworthy properties of the client platform is the level of trust and security that the different credential systems already inherently provide by themselves and/or require from the surrounding platform. Since FutureID shall be capable of handling different credential systems, including the classical username/password combination, we need to distinguish between the following trustworthiness levels:

- **Level 1: Password or PIN code in untrustworthy execution environment.** To authenticate using his eID (i.e. the user name in this case), the user has to log in to a central online system using a password or PIN code. The password or PIN is entered on a regular computing device (e.g., PC, Smartphone), either in a web browser, or in a dedicated eID application installed on the device. There is no trusted hardware to protect the input environment, meaning that the password or PIN code can fall victim to phishing attacks if the device is infected by malware or if the user enters its password on a website that mimics the central online system.

- **Level 2: Cryptographic secret in untrustworthy execution environment.** The eID involves a strong cryptographic secret (e.g., a client-side public-key certificate) to authenticate the user. The secret is stored on a regular storage medium on a regular computing device, without trusted hardware or secure environment to protect the secret. Thereby, the secret may be compromised by malware running on the user's machine.

- **Level 3: Password or PIN code in trustworthy execution environment.** The user is authenticated by a username and password or PIN code that is entered either on a trustworthy input device (e.g., a dedicated hardware token or card reader) or on a regular device that is equipped with a piece of trustworthy hardware (e.g., a PC with TPM, a Smartphone with secure element) or software (Trusted Execution Environment) that can verify the state of the device and thus can guarantee the confidentiality of the PIN/password. In the latter two options, the trustworthy hardware or Trusted Execution Environment can check that the software or user interface in which the user enters his password is authentic and that it connects to the correct online service, so that the user's key strokes cannot be intercepted by malware or by a fraudulent website. The un-trusted device must have some way of indicating to a user that the currently running application is trustworthy, e.g., a dedicated LED on the device, or a special symbol in a dedicated area of the screen.
  We note that a trustworthy execution environment in software protects against software attacks by malware, but does not protect against hardware attacks, e.g., where the attacker has access to the data bus. Trusted hardware does protect against such attacks.

- **Level 4: Cryptographic secret on trustworthy hardware.** The user is authenticated by means of a strong cryptographic secret (e.g., digital signature key, MAC key, or privacy-ABC) that is stored

on a piece of trustworthy hardware. The interface exposed by the trustworthy hardware gives access, possibly after entering a password or PIN (in a potentially insecure environment), to basic cryptographic routines that involve the "core" cryptographic secret (e.g., a signing routine), but does not expose the cryptographic secret itself. Therefore, malware or viruses on the user's computation device may be able to affect individual transactions, but cannot steal the cryptographic secret to fully impersonate the user. The cryptographic secret may only be a small part of the information required to authenticate the user. More information such as attribute values or secondary secrets may be stored on other un-trusted devices, but the core secret must be involved to produce a valid authentication.

- **Level 5: Full authentication on trustworthy hardware.** There is a piece of trustworthy hardware that contains one or more cryptographic secrets and all necessary information to perform the authentication. All computation and PIN entry takes place on the trustworthy hardware; the surrounding un-trusted device is merely used as a communication channel to the outside world. On mobile devices, this level can also be achieved by a combination of a secure element and a Trusted Execution Environment for secure PIN entry and secure display. In this case, the relevant cryptographic algorithms and protocols are either performed in the trustworthy hardware or within the Trusted Execution Environment.

FutureID has to be capable of detecting the achieved level and has to handle this information in a secure and integrity-protected way. Depending on the actual level of trustworthiness, the user shall only be able to access a service that requires the achieved level or a lower one.

## 6.1 Trustworthy Properties of existing eID Systems

In the previous section five trust and security levels have been defined. To link this abstract classification to the real world of running roll-outs of eID in Europe, the trustworthy properties of existing eID systems in Europe and their related client platforms are discussed using concrete examples for each level.

**Level 1: Password or PIN code in untrustworthy execution environment**

Some few references are running on level 1, for example in UK and Austria the annual e-tax declaration of the citizen, in Netherlands the e-government services offered from the municipality to the citizens and in Norway various services from different government organization on local and national level. In the case of Norway a brief view on the framework gives an impression on this: Since CY2009 Norway has an e-Service platform for e-Government purposes named MyPage in use. Back to CY2005 the decision was made to modernize Norway and its citizens on the way to the digital world. This platform uses a password in the health domain, for child support, in the driving license domain and many others. 4.6

million citizens have a unique ID number today, 75% have a PC at home, 60% use the PC every day, 60% have internet access at home, 50% have access to xDSL.

**Level 2: Cryptographic secret in untrustworthy execution environment**

In the public domain this framework is not in use in Europe. In the private domain such architecture is known for e-mail correspondence between two parties based on S/MIME encryption.

**Level 3: Password or PIN code in trustworthy execution environment**

Not many states in Europe follow this approach. Typically citizen have the freedom to select and buy PC, laptops and mobile phone in HW and SW as client platform along their own criteria, like price, performance, functionality and many others. In the case of Estonia, the government offers e-services via mobile phone to the citizens, but the mobile phones need a special configuration. Since CY2003 Estonia has e-Services in use based on two platforms

-eID card in Estonia, 2-factor authentication

-mobile phone, 1-factor authentication (special configuration)

The e-services capture all kinds of services in the public and enterprise domain as well as in the public transport. Services based on electronic signatures are only possible with the national eID-Card and the PKI-infrastructure. Estonia, with a population of 1.3 million citizens has more than 88% internet user and more than 100% mobile penetration. More than 1000 free internet access points are available in the country. The roll-out of the eID card started in Jan 2002. In May 2007 the government has launched the application with a standard mobile phone. Only a change of the SIM-Card to a PKI-SIM card is needed. A SW change is not requested. The major GSM operator EMT has supported this service.

**Level 4: Cryptographic secret on trustworthy hardware**

Currently there are no implementations known in Europe.

**Level 5: Full authentication on trustworthy hardware**

15 European countries have such an approach in roll-out based on a secure token combined with a client platform and a special SW running on the client platform. The secure token is in the format of a smart

card. The citizen can obtain this smart card along an application process in the local municipality together with an information paper about the application and the requested client platform device in HW and SW. The distribution way would be

a) Secure client platform: the citizen must buy it on a regular market place (shop or internet) or in the municipality; the government publishes a list of qualified or certified client platform devices containing the producer and the device identity code;

b) SW for the secure client platform: the citizen can obtain a CD-ROM (e.g. in Italy, Estonia etc.) or can download the SW from a certified internet provider or form the government service portal (e.g. in Germany).

These 15 countries are: Sweden, Finland, Estonia, Lithuania, Belgium, Germany, Austria, Monaco, Italy, Spain, Ireland, Czech Republic, Serbia, Albania and Portugal. A detail view can be found in deliverable D32.1.

## 6.2 Trustworthy Properties of existing Credential Systems

Kerckhoff's principle imposes that all secrecy in a cryptographic system must be concentrated in a well-defined high-entropy key. In particular, the security of the system must never rely on the secrecy of the algorithms: the adversary is assumed to have full knowledge of the internal workings of the system. Any system violating this principle relies on "security through obscurity", which usually is a very risky strategy.

Cryptographic authentication credentials therefore always contain a strong cryptographic key that allows the user to authenticate to a relying party. This key can be a symmetric key for a symmetric encryption scheme or a message authentication code, an asymmetric key for a signing algorithm or public-key encryption scheme, or the user secret of a privacy-preserving attribute-based credential (Privacy-ABC) system [CKL12]. To authenticate, the user does not reveal the strong secret itself to the relying party, but rather creates a verifiable cryptographic token that is derived from the strong secret.

Strong cryptographic secrets provide better security than a human-memorisable passwords or biometrics, but managing them securely is not straightforward. They are usually too long to be entered manually on a keyboard whenever they are needed, and they are definitely too long for a human user to remember by heart. The secret can be stored on an untrustworthy medium such as the hard drive of a personal computer or the application memory of a Smartphone, but then any virus or malware installed on the device may be able to confiscate the secret and impersonate the user. Encrypting the secret under a password offers only a minimal amount of protection given the low entropy of human-memorisable passwords and the efficiency of modern graphical processing units in cracking them [Goo12].

A much better approach is to store the cryptographic secret on trustworthy hardware or in a Trusted Execution Environment in such a way that the secret cannot be exported in plaintext form. Rather, the interfaces offered by the hardware token or execution environment can be used to generate the derived authentication tokens, or at least a crucial part of these authentication tokens that cannot be computed without knowing the secret.

In a symmetric-key system, for example, the trustworthy interfaces may generate message authentication codes on messages of a particular format. The relying party, who must also know the symmetric key embedded in the hardware, can then verify these message authentication codes. In an X.509 public-key system, the trustworthy interfaces may produce signatures computed using an embedded signing key, but not give access to the signing key itself. Likewise, in a Privacy-ABC system, the trustworthy device may expose interfaces to generate presentation tokens, but will not output the user secret or credentials itself.

Since trustworthy devices are often limited in terms of memory and computational power, one can often optimize the implementation so that the device only stores part of the access credential and does not assemble the complete token that will be sent to the relying party, but only performs a crucial cryptographic step in the process. For example, in an X.509 system, the device may contain the signing key, but the certificate containing the user's attributes and the certificate on the issuer's public key could be stored on external, un-trusted storage media. The computation of the actual cryptographic signature must take place on the device, but the assembling of that signature into, for example, an XML Signature object or a SAML token can take place on an external device. A similar approach was taken for Privacy-ABC systems in the pilots of the ABC4Trust project, where the user secret and credentials are stored on a smart card. The user's PC can read most of the credentials from the card, but not the user secret. When generating a presentation token, the PC takes care of the XML parsing and some of the cryptographic computations on the credentials, but the smart card plays a crucial role in producing a proof of knowledge of the secret key.

There are advantages to both users and issuer of credentials to embed cryptographic keys in trusted hardware or trusted environments. Since the secrets cannot be extracted from the devices, issuers maintain control over the number of deployed devices and can be assured that users cannot share their authentication credentials with other users. Users, on the other hand, are protected from having their cryptographic secrets compromised by malicious software on their machines.

A major disadvantage of this approach, however, is that backup of authentication credentials becomes difficult, if not impossible, because the core cryptographic secret cannot be exported from the trusted environment. Designing solutions for this problem is one of the technical research challenges that are addressed by the FutureID project.

Reflecting these considerations, the security levels as defined in this section require a Trusted Execution environment and/or trusted hardware for the higher levels (i.e. levels 3 and above).

## 6.3 Relevant Standards

Within the context of a trustworthy client platform there is no single standard or specification that would cover all aspects but rather a collection of many standards and specifications that address certain aspects of trust and security. These aspects include the security properties of the platform components as well as the interfaces and protocols used for communication. The following list may provide an indication of the most relevant standards and specifications but can hardly provide a complete overview.

**Smart Card Related Standards**

| Number/Body | Name/Topic |
|---|---|
| ISO 7816 | Integrated Circuit Cards |
| ISO 14443 | Contactless cards |
| ISO 7501 1-3 | Machine readable travel documents |
| ISO7810 | ID cards |
| ISO 15693 | Vicinity cards |
| ISO 17839 | Biometric system on card |
| ISO 18013 | Driving license |
| CEN 15480 | European Citizen Card |
| ISO 11889 | Trusted Platform Module (TPM) |
| Global Platform | TEE Specifications: Client API, Internal API, System Architecture |
| Global Platform | Device Application Security Management Specification |

**Interfaces, APIs:**

| Number/Body | Name/Topic |
|---|---|
| ISO 24727 | Card API |
| ISO 18092 | Near Field Communication (NFC) |
| ISO 21481 | NFC interface and protocol |

| | |
|---|---|
| ISO 9798 | Entity authentication |
| ISO 15443 | Security Assurance Framework |
| ISO 11770 | Key management |
| ISO 29115 | Entity authentication assurance framework |
| ISO 24760 | A framework for identity management |
| SIM Alliance | Open Mobile API |
| PKCS#11 | Cryptographic Token Interface |
| PC/SC | Smart Card interface stack specification |

**Security Evaluation**

| **Number** | **Name/Topic** |
|---|---|
| ISO 15408 | Common Criteria |
| ISO 19791 | Security Assessments of operational systems |

**Credentials, Protocols**

| **Number/Body** | **Name/Topic** |
|---|---|
| ITU-T | X.509 certificates |
| OASIS | Security Assertion Markup Language |

Generally, it has to be assumed that the mentioned standards, whenever they apply and whenever they are required, will be implemented correctly and completely, especially with respect to their security-related requirements. An incorrect implementation may create a security loophole and may therefore not deliver the level of trust that is expected and required. As a consequence, the client platform may not be able to provide the required level of trust and security even if the incorrectly implemented component is not directly part of the platform.

# 7. Components for trustworthy Platforms

## 7.1 Secure Elements for Mobile Devices

### 7.1.1 SIM/UICC

The Subscriber Identity Module (SIM) is the most frequently used secure element in mobile devices and is based on a tamper-proof security IC with an appropriate secure operating system. Since the operating system is mainly encoded with a ROM-mask during hardware manufacturing, it cannot be tampered with by a simple software attack. Only a suitable hardware attack could modify the code. Therefore, the chip provides several measures to prevent or at least detect a hardware attack. Secret information stored on the card is used for authentication within a mobile network and access to the card is protected by a PIN in combination with a retry counter. The card also provides cryptographic functions.

Over the recent years, this card has evolved to a UICC (universal IC card) which can host several applications in parallel. In this case the SIM functionality is one of several applications on the card. New applications can be loaded onto the UICC via the internet ("over-the-internet", OTI) or via the cellular network ("over-the-air", OTA) allowing the support of additional applications after field rollout. Appropriate card management functions have been defined to allow a secure managing of card applications and to avoid interference or information leakage between the applications.

For mobile devices supporting Nearfield Communication (NFC), the NFC interface can securely connect directly to the UICC via the Single Wire Protocol (SWP). Thus, NFC-based applications, like mobile payment can rely on the trust and security functionality of a secure element.

While the SIM/UICC is owned by the network operator and is interchangeable, the same functionality can be located on an embedded secure IC which is owned by the device manufacturer. Since the element is not removable, all applications and credentials stored on it need to be manageable from remote. In principal, an embedded secure element could be used in addition to a classical removable UICC and could support additional security applications independent from the mobile network operator.

### 7.1.2 Embedded Secure Element

While the SIM/UICC is owned by the network operator and is interchangeable, the same functionality can be located on an embedded secure IC which is owned by the device manufacturer. Since the element is not removable, all applications and credentials stored on it need to be manageable from remote. In principal, an embedded secure element can be used in addition to a classical removable UICC and can support additional security applications independent from the mobile network operator.

### 7.1.3 Secure microSD Card

Another way to integrate a secure element into a mobile phone is to use a secure microSD card. This is a smartcard chip combined with a flash memory chip inside a conventional microSD memory card. The smartcard can communicate with the host via the Advanced Security SD (ASSD) interface and its feature set includes multi-application support and hardware-accelerated cryptography like other smartcards (e.g. UICC see above).

As microSD cards are removable elements, they are neither bound to the mobile network operator nor to the mobile phone manufacturer, and can thus be controlled and used by an independent third party. Generally these cards are available on any phone with a microSD card slot. On Android, secure microSD cards can be accessed in the same way as SIM cards and embedded secure elements using the Open Mobile API.

### 7.1.4 Contactless Smartcards via NFC

As the NFC interface is compatible to ISO 14443, it can provide access to external contactless smartcards. This allows using for example eID cards as secure elements in a convenient way, by touching them with the phone and without requiring a dedicated smartcard terminal.

### 7.1.5 Properties of Secure Elements

Independent of the actual form factor (i.e. embedded or removable), secure elements offer a large set of security and trust properties. Some of these functions depend on the actual implementation of the card and may not be available in all types of cards (e.g. depending on price level). The following overview assumes that at least the most common features have been implemented:

| | |
|---|---|
| **Tamper-proof hardware** | The security IC is typically tamper-proof and provides a larger set of functions to prevent or at least detect hardware attacks. These functions include active shielding, bus scrambling, dynamic memory scrambling, temperature detectors, photo detectors and clock frequency detectors. |
| **Side channel leakage resistance** | The combination of hardware and operating system is optimized with respect to reduced side channel leakage. This form of information leakage offers a potential threat by which secret information can be obtained without actually attacking the direct channel, i.e. the cryptographic algorithm or the communication protocol. Side channel attacks can be timing attacks, power analysis attacks or fault injection attacks. Both, |

| | |
|---|---|
| | hardware as well as operating system implement measures to suppress these types of attacks as much as possible. |
| **System Integrity Attestation** | During booting, the secure elements can verify the integrity of their firmware by calculating a cryptographic checksum over the code. |
| **Secure Authentication** | By using cryptographic secrets and appropriate authentication protocols, the card allows for secure authentication. A successful authentication can be the requirement for accessing certain data or performing certain functions on the card. |
| **PIN/PW Protection** | Another way of protecting information on the card is to request a PIN or password entry. Only after successful entry certain information or functionality can be accessed. |
| **PIN retry counter** | This counter allows limiting the number of allowed false attempts for a PIN or password entry. It has to be resistant against modification or timing-based interrupts of the power supply. |
| **Secure Messaging** | Secure messaging allows protecting the communication between the card and the interface device with respect to integrity (cryptographic signature) and/or confidentiality (encryption). |
| **Secure Storage** | The combination of tamper-proof hardware and access control allows for secure storage of secret information like cryptographic keys. It can be enforced that secret information never leaves the secure environment of the card. |
| **Cryptographic Services** | The card typically provides algorithms for encryption and decryption, signature creation and verification and key generation (symmetric/ asymmetric). A cryptographic key pair can be generated on the card, while the secret key never leaves the card environment. |

## 7.2 Trusted Platform Modules, TPM/MTM

In computing, Trusted Platform Module (TPM) is both the name of a published specification detailing a secure crypto-processor that can store cryptographic keys that protect information, and the general name of implementations of that specification, often called the "TPM chip" or "TPM Security Device". The TPM specification is the work of the Trusted Computing Group. The current version of the TPM specification is 1.2 Revision 116, published on March 3, 2011. This specification is also available as the international standard ISO/IEC 11889.

Today round 70 to 80% of all new produced and sold PCs and laptop have TPMs hardwired on the mother board in the standard configuration. This TPM can be used for example for encryption of e-mails, e.g. along the S/MIME architecture and for other use cases.

The Trusted Platform Module offers facilities for the secure generation of cryptographic keys, and limitation of their use, in addition to a random number generator. It also includes capabilities such as remote attestation and sealed storage.

- "Remote attestation" creates a nearly unforgeable hash-key summary of the hardware and software configuration. The program encrypting the data determines the extent of the summary of the software. This allows a third party to verify that the software has not been changed.
- "Binding" encrypts data using the TPM endorsement key, a unique RSA key burned into the chip during its production, or another trusted key descended from it.
- "Sealing" encrypts data in similar manner to binding, but in addition specifies a state in which the TPM must be in order for the data to be decrypted (unsealed).

Software can use a Trusted Platform Module to authenticate hardware devices. Since each TPM chip has a unique and secret RSA key burned in as it is produced, it is capable of performing platform authentication.

Generally, pushing the security down to the hardware level in conjunction with software provides more protection than a software-only solution. However even where a TPM is used, a key would still be vulnerable while a software application that has obtained it from the TPM is using it to perform encryption/decryption operations, as has been illustrated in the case of a cold boot attack. This problem is eliminated if key(s) used in the TPM are not accessible on a bus or to external programs and all encryption/decryption is done in the TPM.

The mirror embedded security architecture of TPM in a PC is the MTM in Smartphones. “MTM” stands for Mobile Trusted Module. MTM v1.0 use cases, in the context of the TPM 1.2 command subset, include platform integrity, device authentication, secure channel between device and UICC and secure software download. Mobile commerce use cases are defied for example for mobile banking, ticketing and rights management. Since mid of CY 2012 MTM v2.0 is available to address TEE and related APIs to foster easy use and to increase flexibility.

## 7.3 Trusted Execution Environments

Mobile devices so far have little security software installed like a firewall, a virus scanner and intrusion detection software, although they are always connected and store a lot important user data. Compared to desktop devices they are therefore more vulnerable to malware being brought onto the device,

especially to the limited awareness of users when selecting applications from the app store. The concept of controlled app stores has not prevented malware from spreading, especially for Android-based Smartphones. Therefore, the trustworthiness of a mobile device platform can be impacted negatively by malware that can spy on password entries or eavesdrop on the communication with secure elements.

One approach to enhance the platform trustworthiness again is the concept of a Trusted Execution Environment (TEE). It offers a trustworthy execution environment that is completely separated from the normal ("rich") operating system context. Mechanisms are implemented to protect the integrity of the TEE software itself with a secure boot process and the integrity of applications running within the TEE context. In addition, the confidentiality of the other assets within the TEE is protected as well. The separation of the TEE and the rich OS is deeply supported by the mobile device processor architecture (e.g. ARM TrustZone or Intel TXT). The processor provides a dedicated hardware section to run trusted applications and to establish a trusted environment. Additionally, the access to shared resources (e.g. memory) is controlled and strictly separated.

Although not a secure element in the strict sense, the Trusted Execution Environment (TEE) for mobile devices offers security against a large range of software attacks on mobile devices. A TEE can host trusted services ("Trusted Applications") for security critical applications like eID services, payment applications and digital rights management (DRM). A typical application would be the secure password entry. If the entry is run in the secure TEE, no other malware running in the normal OS could read out key strokes or the keyboard buffer. During execution of the secure password entry Trusted Application, the keyboard driver and the display are completely controlled by the TEE while the normal OS has no control over it anymore. This purely software-based solution would be somehow comparable to the PIN or password entry on a separate smart card reader connected to a desktop computer.

The TEE can also encrypt secret data and store it in encrypted form in the memory. In the ideal case, this memory would be physically separated from the normal memory and could only be accessed by the TEE. In most cases the TEE will have exclusive access to a certain part of the normal memory. Although the data is encrypted, this approach is not comparable to a tamper-proof secure element, where the uncontrolled readout is additionally suppressed by physical means. It is always possible however to combine a TEE with a tamper-proof secure element and to store secret data only in the secure element.

Trusted Execution Environments are currently standardized by Global Platform. The following security features are provided by a Trusted Execution Environment:

| | |
|---|---|
| **System Integrity Attestation** | By establishing a boot sequence of trust, the TEE can ensure that its integrity is maintained and has not been altered by malware. Therefore, the secure world part of the mobile can trust on the integrity of the TEE. |

| | |
|---|---|
| **Secure Authentication** | By using cryptographic secrets and appropriate authentication protocols, the TEE allows for secure authentication. A successful authentication can be the requirement for accessing certain data or performing certain trustlets within the TEE. |
| **PIN/PW Protection** | Another way of protecting information on the card is to request a PIN or password entry. Only after successful entry certain information or functionality can be accessed. |
| **PIN retry counter** | A PIN retry counter can also be established within the TEE environment. This counter allows limiting the number of allowed false attempts for a PIN or password entry. Since the counter is not realized in a tamper-proof hardware the security level may not be as high as in a smart card. |
| **Secure Messaging** | Secure messaging allows protecting the communication between the normal world and the TEE with respect to integrity (cryptographic signature) and/or confidentiality (encryption). |
| **Secure Storage** | The TEE can automatically encrypt secret information that shall only be handled in the TEE context. It is therefore not accessible in plaintext by the normal OS. However, without support of a secure element, the TEE does not allow the same security level for storage as a tamper-proof hardware device. |
| **Cryptographic Services** | The TEE typically provides algorithms for encryption and decryption, signature creation and verification and key generation (symmetric/ asymmetric). |

## 7.4 Security Protocols

The FutureID client platform requires protocols to communicate with and to use the underlying security hardware. On all platforms an Interface Device (IFD) service should be available to access secure elements and support protocols for (remote) terminal authentication and secure messaging setup, like PACE for the German eID card. In case the platform provides a Trusted Execution Environment (TEE), this should be used to create trusted paths between input/output devices and secure elements for security sensitive user interaction (like PIN input), as well as the establishment of trusted network channels. As defined in section 6, a secure user interface is required in security level 3.

If a platform does not possess a Trusted Execution Environment, it can be hardened against runtime attacks in software. This includes especially binary-level vulnerabilities like Buffer Overflow, Return-to-

libc Attacks and advanced Return Oriented Programming Attacks. There are various countermeasures against those attacks, which can be deployed both at compile-time and at runtime:

| | |
|---|---|
| **Compile Time Defences** | Many compilers allow automatically adding stack canaries to prevent the corruption of return addresses and the frame pointer. The main disadvantage of these defences is that they have to be used during compilation using the source code and cannot be added later, and there are techniques to bypass them. The new versions on GCC use this approach. |
| **Instruction Set Randomization** | Randomizing the instruction set at program launch successfully prevents any code-injection, as the attacker does not know the correct instruction set a priori. However, without specialized hardware support this technique is too expensive in practice. |
| **No-Execution Bit (NX)** | Adding an NX bit to the memory address allows the processor to distinguish between code and data segments, and subsequently disables execution of instructions in data segments and the stack. This prevents any kind of code-injection attack, but does not help against code-reuse (like Return-to-libc). |
| **Address Space Layout Randomization (ASLR)** | ASLR randomizes the layout of the memory address space. This defense tries to make code-reuse attacks harder, because the attacker does not know where to find his code segments. This countermeasure is still vulnerable to memory disclosure attacks and Just-In-Time Return Oriented Programming Attacks. |
| **Control Flow Integrity (CFI)** | CFI aims at preventing control flow attacks completely by ensuring that programs follow a predefined behavior. This is done by first extracting the correct control flow graph of the application and then checking at runtime that no control flow instruction violates this graph. These steps can also be performed on the binary, i.e. without knowledge of the program's source code. |
| **Program Shepherding** | Program shepherding enforces security policies on running programs by executing them with a machine-code interpreter. This allows imposing runtime restrictions like control flow integrity, but the performance and memory overhead is quite high. |

The security protocols discussed here are all focused on the topic of platform integrity attestation, that is, the proof towards a remote entity that the platform being attested operates in a specific state. In the context of *FutureID* the remote entity could for example be an application service or the *FutureID* server infrastructure, which wants to ensure that the client platform is in a trustworthy state. The state of the platform consists of a set of measurements that are stored in the Platform Configuration Registers (PCR) inside the TPM. Each measurement in this context is calculated on the concatenation of the previous measurement and the SHA-1 hash of a piece of code to be executed or data to be processed. The nature of the concatenation operation requires a strict order of measurements to return the same value. However, if this order can be guaranteed, the measurement of a known-good system is always the same. Hence, a remote entity can compare the measurements, provided as part of the remote attestation, with a list of valid configurations and decide whether the platform can be considered trustworthy.

### 7.4.1 Remote Attestation

Each TPM has an RSA key pair, the so-called Endorsement Key (EK), which is generated during the production of the chip and stored in a read-only part of the TPM. Hence, this key can be used as a unique identifier of the chip. Even though this key pair could, in principal, be used to sign for the authenticity of the data required during an attestation, this would be a major privacy issue, as every transaction could be linked to the same physical device.

Consequently, the remote attestation, as specified for the version 1.1 of the TPM, uses intermediate key pairs, the Attestation Identity Keys (AIK), instead of the EK. Each AIK requires a certificate that is signed by a Trusted Third Party (TTP), the so-called Privacy CA, to proof that the AIK is linked to a genuine EK and therefore a genuine TPM (but not to which one) and that the AIK resides within the TPM. In this scenario the Privacy CA can easily link the EK and the issued AIK certificates; hence, the anonymity depends on the correct operation and the goodwill of the Privacy CA.

### 7.4.2 Direct Anonymous Attestation (DAA)

The Direct Anonymous Attestation scheme was developed by Brickell, Camenisch and Chen in 2004 [Brick04] and has been adopted in the TPM specification 1.2 by the Trusted Computing Group (TCG). It is designed to provide anonymity without a Trusted Third Party (TTP) and authentication without a certificate. It is a group authentication scheme without the capability to open signatures. The basic mode of operation is the following: An issuer creates a group public key. From this issuer each TPM obtains a unique membership attestation credential. With this credential the TPM creates signatures, which can be verified by an external verifier using the group public key. The DAA scheme provides a revocation mechanism. If a TPM has been compromised in a way that the membership attestation credential has been exposed, then this credential is placed on a revocation list. The verifier can check received

signatures locally against this list. The DAA scheme provides two options concerning the base for signature generation. With the *random base option*, for every DAA signature a different base is used. Thus, any two signatures of a TPM cannot be linked. In this case revocation only works if the attestation credential is revealed to the public. If the TPM is still controlled by an adversary and the attestation credential has not been revealed, a revocation is not possible. The other option is the *named base option*. Here, a deterministic function, e.g. using the verifier's name, is used as base for the signature. Thus, all signatures by one TPM using the same base are linkable. If the verifier determines a signature computed with a compromised attestation credential, the credential can be revoked locally without knowing the value of the credential. However, this revocation cannot be shared with other verifiers, which use a different base [Brick07].

The following part is a slightly more detailed view on the DAA scheme based on [Pash05]. The DAA mechanism has two main phases: The *Join phase* and the *Sign phase*.

In the *Join Phase* the TPM obtains an attestation certificate from the issuer. The issuer is typically the manufacturer of the TPM. At first, the TPM internally generates a long-term secret key, which is never exposed outside the module. Then, it sends the public part of its Endorsement Key (EK) to the issuer and also proves its knowledge of the secret key. The issuer verifies that the TPM is genuine by comparing the submitted EK to a list of endorsement keys for genuine TPMs. If the issuer was able to verify the genuineness of the questioned TPM, a Camenisch-Lysyanskaya (CL)-signature on the blinded secret key is computed. Thus, the issuer doesn't know anything about the secret key. Therefore, the issuer also cannot revoke a secret key in case of a compromised TPM. The issuer then encrypts the CL-signature under the TPM's public EK, which can only be decrypted with the corresponding private EK part. This encrypted signature is returned to the TPM. The TPM decrypts it and removes the blinding factor, which in fact finalizes the computation of the signature on the secret key. The resulting signature is the attestation credential and is stored in the TPM's protected non-volatile memory. Like the secret key, it is never exposed outside the TPM.

*Sign Phase*: Using such an attestation credential the TPM can prove to a verifier its membership to a certain group, i.e. that the TPM is genuine. The TPM can generate AIKs and use them with the attestation credential to create DAA signatures. Using a zero-knowledge proof, the TPM proves knowledge of the attestation credential to the verifier and thus, that the AIK used for the DAA signature was generated on a genuine TPM. The values of the secret key and the attestation credential are thereby not revealed to the verifier. Depending on the base used within the signature, the verifier has the possibility to revoke the membership credential as mentioned above.

### 7.4.3 Trusted Network Connect (TNC)[1]

Within the working group WG-TNC, the Trusted Computing Group (TCG) developed a suite of standards for the authentication of network endpoints as well as the exchange of integrity measurements. This protocol allows an assessment of the trustworthiness of a network endpoint during the connection phase as well as a reassessment of the state of an endpoint during the lifetime of a connection.

The protocol distinguishes three main actors:

- **Access Requestor (AR)**
  The AR, in our case for example the *FutureID* client, operates the TNC client. The purpose of the TNC client is to send the current system state to the TNC server. This component doesn't perform measurements by itself but relies on so-called *Integrity Measurement Collectors* (IMC). These IMCs are responsible to measure specific subsystems, for example, the state of a virus scanner, a firewall, or that the latest security updates have been installed. The protocol specifies that the use of a TPM is optional, however, if the chain of trust from a secure boot via a trusted operating system to a measured IMC cannot be established using a TPM the integrity of a measurement of IMC cannot be guaranteed, as malicious software might have tampered the results.
- **Policy Decision Point (PDP)**
  The PDP is responsible for the evaluation of the measurements generated by the IMCs. The server system has to be extended by a TNC server, which uses a set of *Integrity Measurement Verifiers* (IMV) that act as the counterpart of the respective IMCs. Based on the result of the IMVs the PDP forms a decision about the trustworthiness of the AR.
- **Policy Enforcement Point (PEP)**
  The PEP is the part of the network that will enforce the decision of the PDP. It is usually an active component of the network like a WLAN access point or a switch. In case the verification was successful the access to the server will be granted. Otherwise there are basically two options: Either the access will be blocked or the client will be routed to an unprivileged subnet in order to bring the AR to a trusted state, for example by updating the virus scanner or installing the latest security updates.

This system of IMC/IMV pairs allows for fine-grained policies and a reduction of the overall complexity. For example, the system policy might require a virus scanner, either a specific one or one of a group of supervised scanners. The definition files of the virus scanner can change on a daily basis, therefore also the measurements have to be changed daily. In this approach only the corresponding IMV has to be updated. The measurements of the underlying OS or other subsystems stay untouched.

[1] See: [Muel08]

Furthermore, this approach can be used on clients that do not have a TPM installed or activated. A subset of security policies could still be enforced, like a proper firewall configuration or a virus scanner. This is, on the other hand, a somewhat weak argument as a corrupted system could forge the correct measurements, since no chain-of-trust can be established to a safe system state, i.e. at power on.

### 7.4.4 WS-Attestation[2]

IBM Research developed an attestation extension to the WS-* protocol suite. They also identified the mapping of the state of a system during runtime based on measurements as not practicable as the complexity of such system becomes soon computationally infeasible. Hence, they propose the combination of multiple approaches:

1. Secure boot: A chain of trust is built during boot with the help of a TPM. This chain of trust is extended into the operating system by means of a trusted boot loader, for example Trusted-Grub.
2. Run-time measurement at OS-level: Certain tasks such as module loading and application execution are monitored by the operating system. The Integrity Measurement Architecture[3] realizes such measurement on the Linux kernel.
3. Run-time measurement at middleware-level: The middleware is responsible to monitor the code it executes. An example of the measurement at middleware is a Java™ Virtual Machine (JVM) that measures integrity of Java class files when each class is loaded.
4. Measurement by agents: A hash-based evaluation of a system has limited use when used on mutable data, e.g. configuration or policy files. Some of the configuration is host dependent; therefore a set of measurements must be maintained for each host. Furthermore, any semantically meaningless change, such as a whitespace change, would break hash-value based verification. Hence, a software agent could be used to measure certain parts of a system, for example, configuration files in a canonical form where parts that are allowed to be different (the host IP address for example) can be blanked. The agent itself must also to be part of the chain of trust.

Another interesting property of this approach is the dedicated validation service (VS), which operates as a TTP. The VS is in charge of maintaining an Integrity Database that consists of a set of vulnerability databases and a set of software repositories. Based on this data the VS can decide whether a tested platform is trustworthy and can attest this trustworthiness to an attestation requester. This mitigates the burden on the part of the attestation requester who does not need to operate an integrity database on

[2] See [Yoshi06]
[3] http://linux-ima.sourceforge.net/

her own, and also mitigates the risk of the tested platform that a malicious attestation requester gains knowledge about the exact state of the tested platform, which might give him clues about unpublished or un-patched vulnerabilities.

## 7.5 Privacy-preserving Credential Systems

Privacy-preserving Attribute-Based Credential (Privacy-ABC) systems enable users to authenticate to a relying party in a secure yet privacy-friendly fashion. Users obtain *credentials* on lists of attributes from credential issuers. From these credentials, they can derive so-called *presentation tokens* that reveal certain information about a credential or a combination of credentials, for example, a subset of the attribute values. The presentation token can be verified by the relying party using the issuers' public keys.

The privacy-preserving aspect of Privacy-ABCs means that presentation tokens reveal no other information than what is explicitly disclosed. In particular, they do not reveal any information about undisclosed attribute values. Presentation tokens are also unlinkable and untraceable. The former meaning that given two presentation tokens for the same disclosed information, one cannot tell whether they were derived by the same user or by two different users. The latter means that an issuer cannot trace back a given presentation token to the issuance of the credentials from which the presentation token was derived.

### 7.5.1 Features of Privacy-ABCs

A complete description of the concepts and features of Privacy-ABC systems can be found in the deliverables of the ABC4Trust project [CKL12], we summarize a number of selected features here.

- **User secrets and pseudonyms.** A user can generate a *user secret* that, unlike classical public-key systems, corresponds to an infinite number of public keys called *pseudonyms*. A pseudonym is derived from the user secret and enables the user to introduce a controlled form of linkability to presentation tokens by reusing the same pseudonym across different presentation tokens. Different pseudonyms are unlinkable, however, in the sense that a relying party cannot tell whether two different pseudonyms were derived from the same user secret or from different ones.
  A *scope-exclusive pseudonym* is a special kind pseudonym that is uniquely defined by a user secret and a *scope string*. Meaning, for a given scope string and a given user secret, there is only one scope-exclusive pseudonym that can be generated. This can be used, for example, to ensure that each user can only create one pseudonymous account at a particular website: the scope

string can be set to the website URL, so that different users are still pseudonymous, but each user can only create one such pseudonym.

- **Key binding.** Optionally, credentials can be *bound* to a user secret, meaning that the user has to know the user secret in order to derive a presentation token from the credential. Moreover, the user can prove in a presentation token that multiple credentials and/or pseudonyms are bound to the same user secret.

- **Minimal disclosure.** A presentation token can disclose the value of any subset of the attributes of the credential(s) from which it is derived; it does not leak any information about undisclosed attributes. Moreover, the presentation token can merely reveal the fact that one or more attribute values satisfy a certain predicate, without revealing their exact value. For example, the presentation token can prove that an attribute is greater or smaller than a given value (e.g., to prove minimum or maximum age based on a birth date) or prove that two attributes in two different credentials have the same value (e.g., to show that two credentials were issued to the same person).

- **Inspection.** In certain use cases, complete anonymity can lead to abuses because perpetrators cannot be held accountable for their actions. The feature of *inspection* adds accountability to Privacy-ABCs by letting the relying party impose that a user encrypts her identity (or more generally, the value of some attribute in her credentials) to a trusted *inspector*. The relying party cannot decrypt the encrypted identity himself, but can rest assured that the correct attribute value was encrypted. When a condition occurs that violates the pre-agreed *inspection grounds*, the relying party forwards the presentation token to the inspector, who reveals the encrypted value.

- **Revocation.** When credentials are lost, stolen, or compromised, or when attribute values change, issued credentials must be revoked so that they can no longer be used. Common approaches from classical public-key infrastructures such as revocation lists don't work for Privacy-ABCs, since the anonymity of presentation tokens means that they cannot contain a unique identifier such as a public key or certificate hash that can be checked off from a list. However, efficient revocation mechanisms have been designed and implemented that allow the user to prove that his credential was not revoked, without revealing more information.

### 7.5.2 Binding Privacy-ABCs to Trusted Hardware

The easiest way to "bind" Privacy-ABCs to trusted hardware tokens such as a smart card is by storing the credentials on the trusted hardware and only exposing interfaces to create presentation tokens, so that the credentials themselves cannot be exported from the device. A possible disadvantage of this approach, however, is that the trusted hardware may lack the memory or computing power to generate

a full presentation token, especially when complicated presentation policies need to be parsed and executed.

An alternative approach exists whereby credentials can only be used with a trusted hardware token, but most of the computation and storage can be offloaded to an untrusted device such as a PC or Smartphone. Namely, the trusted hardware can store just the user secret and offer an interface that takes care of the proof of knowledge of the user secret, for those presentation tokens that involve credentials that are bound to the stored user secret. Users thereby enjoy the increased security that viruses and malware cannot steal the users' credentials. At the same time, credential issuers can restrict sharing of access credentials to the number of hardware tokens that were distributed.

### 7.5.3 Direct Anonymous Attestation

The Trusted Computing Group adopted the Direct Anonymous Attestation (DAA) protocol [Brick04] to enable privacy-friendly remote authentication of Trusted Platform Modules (TPMs). The protocol embeds a different key in each TPM so that the compromise of a single device does not destroy the security of the entire system. The key can be used to establish different unlinkable pseudonyms with different relying parties, but the pseudonyms can be checked not to be derived from a rogue key that is known to have been compromised.

The DAA protocol can actually be seen as a particular instantiation of a Privacy-ABC system. Namely, the TPM contains a user secret and a credential issued by the TPM manufacturer and bound to the user secret. When authenticating to a relying party, the TPM generates a scope-exclusive pseudonym for a scope string that is either a one-time unique string (e.g., a randomly chosen nonce) or specific to the relying party (e.g., the relying party's URL). It then generates a presentation token proving that it has a valid credential - issued by the TPM manufacturer - that is bound to the same user secret as the scope-exclusive pseudonym. The security of the credential system ensures that only registered TPM manufacturers can distribute valid TPMs, while the anonymity of the pseudonyms and presentation tokens ensures that one user's movements across the Internet cannot be linked together.

When a rogue TPM is detected (e.g., a valid user secret and credential were found online or were embedded in some software), the user secret is distributed to all potential relying parties. Since a scope-exclusive pseudonym is uniquely determined by the user secret and the scope string, relying parties can check that the scope-exclusive pseudonym generated by a particular TPM has not been generated by one of the rogue keys.

# 8. Requirements for trustworthy eID Client Platforms

## 8.1 PC-based Platforms

For all security levels, all relevant standards and specifications shall be implemented correctly.

### 8.1.1 Security Level 1 & 2

For these security levels, no special requirements on the platform apply. However, a basic set of security software consisting at least of up-to-date virus/malware scanner and firewall and the latest security updates should be installed.

### 8.1.2 Security Level 3

**R1.1** The PC-platform must contain a TPM and it must be activated. The platform must also support extended instruction sets like Intel's TXT or AMD's Presidio.

**R1.2** The operating system of the Trusted Execution Environment must fulfil the requirements of a Trusted-OS, that is, it must support secure boot and the extension of the chain of trust within the runtime environment.

**R1.3** Software-based credentials have to be decrypted and processed completely in the trusted environment. In an encrypted format the files may be stored in the normal OS (in case of virtualization).

**R1.4** Password or PIN entry has to occur completely in the Trusted Execution Environment or on a dedicated trustworthy input device, for example the keypad of a certified card reader. The normal OS shall not have any control over the keyboard, the keyboard driver or the keyboard buffer during entry. After completion of password/PIN entry, the keyboard buffer must be empty.

**R1.5** Relevant data, for example data to be signed, must be displayed in a viewer within the trusted environment. The normal OS shall not have access to the frame buffer while the trusted viewer is visible.

### 8.1.3 Security Level 4

**R1.6** The PC-platform must support a secure storage space. This can be an encrypted file, the decryption key, however, must be stored on a hardware device, for example a TPM or a smart card.

**R1.7** The platform should support a Trusted Execution Environment. PIN and password entry should occur completely in the Trusted Execution Environment or on a dedicated trustworthy input device, for example the keypad of a certified card reader. The normal OS should not have any control over the keyboard, the keyboard driver or the keyboard buffer during entry. After completion of password/PIN entry, the keyboard buffer should be empty. Relevant data, for example data to be signed, should be displayed in a viewer within the trusted environment. The normal OS should not have access to the frame buffer while the trusted viewer is visible.

### 8.1.4 Security Level 5

**R1.8** The PC-platform must support a secure and tamper-proof storage space, like a smart card or HSM.

**R1.9** The platform should support a Trusted Execution Environment. PIN and password entry should occur completely in the Trusted Execution Environment or on a dedicated trustworthy input device, for example the keypad of a certified card reader. The normal OS should not have any control over the keyboard, the keyboard driver or the keyboard buffer during entry. After completion of password/PIN entry, the keyboard buffer should be empty. Relevant data, for example data to be signed, should be displayed in a viewer within the trusted environment. The normal OS should not have access to the frame buffer while the trusted viewer is visible.

## 8.2 Mobile Device Platforms

For all security levels, all relevant standards and specifications shall be implemented correctly.

### 8.2.1 Security Levels 1&2

For these security levels, no special requirements on the device platform apply. However, a basic set of security software consisting at least of up-to-date virus/malware scanner and firewall and the latest security updates should be installed.

### 8.2.2 Security Level 3

**R2.1** The mobile device platform must support a Trusted Execution Environment.

**R2.2** Password or PIN entry has to occur completely in the Trusted Execution Environment. The normal OS shall not have any control over the keyboard, the keyboard driver or the keyboard buffer during entry. After completion of password/PIN entry, the keyboard buffer must be empty.

**R2.3** Relevant data, for example data to be signed, must be displayed in a viewer within the trusted environment. The normal OS shall not have access to the frame buffer while the trusted viewer is visible.

### 8.2.3 Security Level 4

**R2.4** The mobile device platform must provide a secure storage space, which is at least secure against software attacks. This can be a secure element, a SIM card, a Mobile Trusted Module (MTM) or a Trusted Execution Environment.

**R2.5** The device platform should support a Trusted Execution Environment for secure PIN/password entry. The entry should occur completely in the Trusted Execution Environment. The normal OS should not have any control over the keyboard, the keyboard driver or the keyboard buffer during entry. After completion of password/PIN entry, the keyboard buffer should be empty.

### 8.2.4 Security Level 5

**R2.6** When no external tamper-proof hardware is used for credential storage and verification (e.g. a smart card), the mobile device must provide a tamper-proof secure hardware element like a secure element or SIM card, which is resistant against software attacks, hardware attacks and side-channel attacks.

**R 2.7** The device platform should support a Trusted Execution Environment for secure PIN/password entry. The entry should occur completely in the Trusted Execution Environment. The normal OS should not have any control over the keyboard, the keyboard driver or the keyboard buffer during entry. After completion of password/PIN entry, the keyboard buffer should be empty.

## 8.3 Privacy-related Requirements

**R3.1** Usage of trusted hardware of environments must not come at the cost of loss of privacy for honest users with respect to relying parties. In particular, an honest user's actions across the Internet must not become more easily linkable when using a trusted environment than without such environment. Any signatures, keys, tokens, or nonces created by the trusted environment must therefore not contain any unique identifiers of the user, or be linkable in any other way.

**R3.2** The manufacturer of a trusted environment must not be able either to link different transactions by the same honest user, or be able to inspect the transaction itself. Preferably, once deployed, the manufacturer should not be involved at all with the usage of the trusted environment.

**R3.3** Security Levels 2, 4, and 5 must support usage of Privacy-ABCs as cryptographic secrets. Authentication is performed by deriving presentation tokens from the stored credential(s).

**R3.4** For Security Level 4, the trusted hardware or environment must support a mode where it stores the user secret of a Privacy-ABC system and collaborates with a presentation token generation performed on an external untrusted device for credentials bound to that user secret. The trusted hardware or environment never leaks the user secret itself to the outside world; the interfaces only allow generating presentation tokens.

**R3.5** For Security Level 5 in conjunction with Privacy-ABCs, the entire presentation token generation happens on the trusted hardware or environment, possibly only for a number of carefully selected credentials and/or presentation policies to obtain a more efficient implementation.

**R3.6** The compromise of a limited number of pieces of trusted hardware or environments (and the leakage of all keys contained within that hardware or environment) should not endanger the security of the system as a whole, and should not force all honest users to get their hardware or environments re-issued.

**R3.7** There must be a mechanism to detect "rogue" hardware or environments that are known to have been compromised.

# 9. References

[Brick07] Brickell, E., & Li, J. (2007). Enhanced privacy id: a direct anonymous attestation scheme with enhanced revocation capabilities. *Workshop on Privacy in electronic society.* ACM.

[Brick04] Brickell, E., Camenisch, J., & Chen, L. (2004). Direct anonymous attestation. ACM Conference on Computer and Communications Security 2004: 132-145

[Muel08] Müller, T. (2008). *Trusted Computing Systems.* Springer-Verlag.

[Pash05] Pashalidis, A. (2005). Interdomain User Authentication and Privacy.

[Yoshi06] Yoshihama, S., Ebringer, T., Nakamura, M., Munetoh, S., Mishina, T., & Maruyama, H. (2006). *WS-Attestation: Enabling Trusted Computing on Web Services.* IBM Japan, Ltd., IBM Research, Tokyo Research Laboratory, Yamato.

[CKL12] ABC4Trust Architecture for Developers. Jan Camenisch, Ioannis Krontiris, Anja Lehmann, Gregory Neven, Christian Paquin, Kai Rannenberg. ABC4Trust deliverable H2.1, 2012. https://abc4trust.eu/index.php/pub/149-h2-1

[Goo12] D. Goodin. Why passwords have never been weaker—and crackers have never been stronger. Ars Technica, 2012. http://arstechnica.com/security/2012/08/passwords-under-assault/