2015 IEEE Trustcom/BigDataSE/ISPA

On Rehoming the Electronic Id to TEEs

Sandeep Tamrakar
Aalto University School of Science
Email: sandeep.tamrakar@aalto.fi

Jan-Erik Ekberg
Trustonic
Email: jan-erik.ekberg@trustonic.com

Pekka Laitinen
Väestörekisterikeskus
Email: pekka.laitinen@vrk.fi

***Abstract*—Government Electronic IDs (EIds) are digital credentials issued to the citizens. In Europe, EIds are distributed in the form of identity cards or passports that allow for identity verification towards government and private services in the digital domain. This paper provides a reference design and implementation examples for Trusted Execution Environment (TEE) based EIds. Especially, the paper highlights the role of attestation during enrolment, a requirement that is not present in legacy EIds.**

I. Introduction

Governments around the world have been issuing digital credentials or Electronic Identity (EId) to its citizen for over a decade. Typically, a government-issued EId is in the form of a smart card that holds the credential of a citizen inside a secure chip and other details about the citizen, such as a picture, name, date of birth etc., printed on the surface of the smartcard. Thus it can be used for both online and physical identification of its holders. However, only in recent years we are experiencing a concrete shift towards citizen services moving online in volume — e-Government systems, wherever applicable, significantly increase both the efficiency and ease-of-use of citizen-government interaction. Meanwhile, there has been a rapid shift in the personal computing devices; Mobile devices (smartphones and tablets) have become the personal computing device of choice among general population. Enterprise domains have been quick in tailoring their services to adapt to the mobile environment. E.g. 40% of the younger age group in the US is already using online banking from their mobile phones [1].

The traditional smart-card based EId is not readily usable in conjunction with the contemporary mobile devices. However, the current mobile ecosystem comes loaded with a complementary security technology called trusted execution environments (TEE). These are available in most of the medium to high-end smartphones today. We postulate that the security level of TEEs on end-user systems (mobile devices) is sufficient for hosting the EId system, at least for end-user authentication. However, integrating a TEE to the government-PKI and EId systems is significantly different from what is customary in the smart card adaptation, especially for identity issuance and assurance.

In this paper we present software and a network architecture that enables EId to be deployed on TEEs as secure endpoints for the authentication ecosystem. We highlight and describe the role of remote device authentication and attestation in the context of citizen EId enrolment, which is one of the main contributions of this paper. We also present real-world implementations of the end-user authentication, based on the Finnish EId specifications, using the two most widely deployed TEEs that allows third-party program provisioning.

The paper is organized as follows: Section II introduces the state of the EIds and mobile EIds in Europe. Section III provides an overview of related works on EId implementation. Section IV outlines the requirements for the EIds and threats associated with it. In Section V, we describe TEEs available for the smartphone platforms. We discuss the TEE attestation mechanisms in Section VI and our architecture in Section VII. We provide implementation details and measurements in Section VIII and conclude in Section IX.

II. EIds and Mobile EIds

EIds for citizens have been widely deployed in some European jurisdictions, especially in Estonia where they are actively used for digital signatures, online voting and even as public transport tickets [2]. EIds have also been used with some success in the Scandinavian countries, Belgium, Austria and Spain. The legal status of these identities is based on the electronic signature directives (1999/93/EC) and the data protection directives (2002/58/EC) issued by European parliament and the council of the European Union (EU) [3].

EU directives use the term *electronic-signature products* (ESP) to describe the trusted endpoint that carries out an EId transaction. An ESP is a logically isolated security token that produces digital signatures on behalf of a user to prove his identity. The user controls the ESP and it often resides with him or in his device. To date, the function of the ESP has been successfully integrated on smart cards as well as on SIM (subscriber identity module) cards. Typically, national laws defines the regulation and procedure required to issue physical Ids. These laws are designed in such a way that they are applicable to EIds with little or no modification [4]. In the EU, the legal status of EId is strongly regulated in both national and European level law. E.g. Finnish national legislation includes the law on Strong Electronic Identification and Electronic Signatures (617/2009) [5]. Similarly the European Parliament and Council are finalizing a proposal on Electronic Identification and Trust Services for Electronic Transactions (eIDAS), which allows EU citizens to verify their identity across borders with their EIds [6].

Currently deployed citizen EIds are predominantly ISO 7816 smart cards that correspond to the PKCS#15 interface standards. Most of the EIds contain two signature keys and provide two forms of signatures — the legally binding "qualified" signature key for document signing is separate from the "authentication" key that is used in more interactive settings. The authentication key can also be used to decrypt data that has been encrypted with the corresponding public key. PINs are

978-1-4673-7952-6/15 $31.00 © 2015 IEEE
DOI 10.1109/Trustcom.2015.356

IEEE computer society

used to authorize the key usage. There are also EId systems that take an extra step to authenticate the service providers. E.g. the ECC-based German EId cards can only be used with a PKI-based live authorization that validates the origin of the Id request [7].

A typical use of EIds is to access e-Gorvernment services such as tax declarations, or health care records, or to digitally sign application forms to receive state benefits etc. Since EIds provide strong authentication, they can also be used to prove entitlement to services in public or private domains: Use case range from being used as transport tickets [2] to driving license substitutes, e-banking tokens as well as physical access control tokens for libraries and swimming pools [4]. Furthermore, the PKCS#15 capable EId cards effortlessly integrate with desktop email clients to provide end-to-end S/MIME e-mail encryption and digital signing.

III. Related Works

The Electronic Simple European Networked Services (e-SENS) [8] is a large-scale EU project that consolidates, improves and extends technical solutions fostering electronic interaction with public administrations across the EU. The project primarily promotes EIds for cross-sector authentication and establishing secure channels. Similarly, the Secure idenTity acrOss boRders linKed (STORK) [9] project aims at establishing a uniform platform for a Pan-European identity. The platform allows Europen citizens to access public services from other member EU states using their EId issued by their home countries. STORK 2.0 expands the STORK mandate by also considering private sector use cases in addition to e-governmental services. The work done in STORK projects is now being realized as the eIDAS legislation.

Implementing EId on a TEE is not a novel concept, Brasser et. al. [10], implemented a PKCS#15 based software token using TEE on PC platforms (Intel TXT and AMD SVM). Similarly, Vossaert et. al. [11] developed a prototype for the Belgian EId infrastructure using TEEs available on PCs. Several other publications have proposed implementing software tokens on smartphones TEE. Tamrakar et. al. [12] implemented a PKCS#15 based Finnish electronic identity card (FinEID) system on a smartphone using a TEE available on the phone. The smartphone exhibits features of a physical smartcard reader with a FinEID card that can be accessed over USB.

BSI TR-03112 eCard API framework [7] and BSI TR-03124 EId client specifications [13] allowed desigining and developing service infrastructures as well as client applications for the German national EId cards. Morgner et. al [14] presented a concept of using NFC-enabled smartphone as a card reader to authenticate to e-services using German EId cards. Meanwhile, there has been numerious open source projects that allow users to access service designed around German EId infrastructure from PCs and smartphones e.g. PersoApp [1], OpenPACE [2] and nPA Smart card library [3].

The above mentioned works all emulate EIds in different TEEs as well as allow accessing EIds from the PC and smartphones. We build from this, and focus on the enrolment problem of EIds on TEEs. This need was identified by e.g. Nyman et. al. [15], in their work for mapping EIds to TPMv2s, however, they do not provide any system integration of attestation with enrolment.

[1] http:///www.persoapp.de

[2] http://sourceforge.net/projects/openpace

[3] https://frankmorgner.github.io/vsmartcard/npa/README.html

IV. Requirements and threats

Dimitrienko et. al. [16] identified the security requirements for hosting an EId in a TEE. These are the *confidentiality* of the identity key, and the *code isolation* and *integrity* of the identity verification algorithm operating on the identity key. The isolation must hold against all unrelated operations in the terminal device, but also with respect to other credentials that may be stored and used in the same token. The *access control* to the credential, with respect to unauthorized users as well as unauthorized code (malware) in the terminal device must also be guaranteed.

The ENISA project has explored privacy and security threats for the online use of smartcards and e-Identities including national EIds [17]–[19]. They classified the threats according to their severity: threats that are mitigable using existing technologies, medium or high. The threats include the inability of the citizens to selectively disclose information for different purposes, such as proving their age to a service which does not require their identities. Similarly, they mention the threat of binding attacks where independent service providers collude and collect a coherent database of all activities that EId customers undertakes. Although advanced technical solutions do exist to alleviate some of these privacy threats, in many jurisdictions they are addressed by regulatory means.

Identified security threats stem on from the weaknesses in the ESP, the host-device, servers as well as the protocols used with EId. Since EId authentication is often performed only in one direction, man-in-the middle attacks and privacy infringement during ESP use are sometimes possible. Weaknesses in host-devices may also shows up due to the weaknesses in PIN protection. Similarly, the lack of a universal out-of-band authentication for confirming a transaction may disconnect the perception of an end-user towards what he authorizes and what he actually approves.

The TEE-based ESPs provide isolation in a manner similar to smartcards, and their isolation properties can be evaluated using third-party audits. However, new threats may arise from the ESP provisioning model in TEEs. TEEs are multi-purpose environments, and it can not be expected that EIds are enrolled into the TEEs at the time of device manufacturing. Instead, a service that enrols the EId keys remotely must be designed in such a way that it ascertains the trustworthiness and the correctness of the remote endpoint TEE in a similar manner as the very same service ascertains itself of the identity of a citizen at the time of his or her credential enrolment. We examine this functionality in Section VI.

V. Trusted Execution Environments

A Trusted Execution Environments (TEE) is the combination of a hardware platform that provides **isolation**, and a software and operating system residing within the security domain defined by that hardware. Additionally, the hardware is capable of running programs launched into that environment. ARM

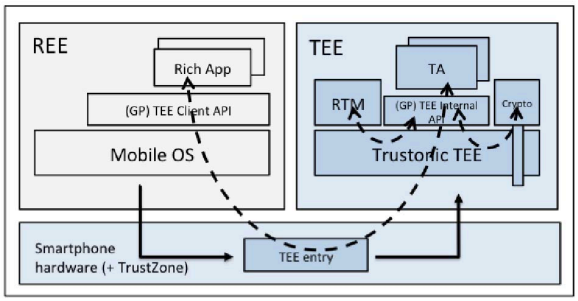

Fig. 1: Trustonic TEE architecture overview

TrustZone [20] is the most widely used TEE that provides a processor with a secure environment and a system architecture for enabling hardware isolation needed for TEEs. TrustZone is available in most contemporary smartphone hardware platforms. With appropriate configurations ARM TrustZone can isolate memory and memory access, CPU tables, interrupts and DMA such that the device can isolate the new **secure world** TEE from the traditional **rich world** OS and applications.

In addition to the isolation property that the hardware provides, the following features further characterize the software and persistent data that constitutes a TEE:

Secret(s): *Trusted Applications* (TAs) which operate in a TEE isolation domain should have exclusive access to derivatives of device secrets provisioned by the chip manufacturer, device integrator or operator. Such secrets are used for secure storage and device authentication.

Provisioning of code and application secrets: For the EId use case to be viable in scale, third-party provisioning of TAs into the TEE is needed.

Cryptographic APIs: Basic cryptographic primitives such as hash algorithms, encryption agorithms and a random data source should be available to the TAs from within the TEE.

We have implemented our ESP as TAs on two different commercially deployed TEEs. In the next subsections we provide brief introductions to these TEEs:

A. Trustonic TEE

The Trustonic TEE is an evolution of a microcontroller operating system (OS) which adheres to the OS principles put forward by Jochen Liedkte and his L4 kernel in the 90's [21]. In this design paradigm, the kernel handles only the *scheduling of threads*, *inter-process communication* and *MMU configuration* as efficiently as possible. To complement the kernel, the *Run Time Manager* (RTM) authorizes and installs new tasks at run-time, serving as the common exception handler for most components in the running system. It is also responsible for memory mapping for all other trusted OS tasks and also maintains a "session management registry". Furthermore, device drivers also reside in the Trusted Computing Base (TCB), since they have access to system memory management. A *cryptography driver* performs all authentication and encryption needed within the TEE. It also shields the use of a hardware-specific device key, which is used for storage and attestation. Meanwhile, the *storage driver* provides persistent storage to trusted applications.

Figure 1 depicts the setup of the Trustonic TEE. Isolation between tasks, or TAs is enforced by the TCB: Isolation is achieved by controlling hardware, i.e. the processor MMU, DMA, privileged and user-mode contexts and register banks, just like it is done in any REE OS. True multi-tasking between TAs in the TEE is therefore possible without security degradation, but overall memory available to the TEE does limit its scope. TAs written for the Trustonic TEEs are native ARM binaries (compiled with e.g. GCC), linked against a dedicated system library.

To make the overall setup a TEE, the whole Trustonic TEE OS and its TAs are isolated from the REE using ARM TrustZone hardware functions. The remote provisioning mechanism for third-party TAs to a Trustonic TEEs architecturally resembles that of GlobalPlatform smart cards. When the device is manufactured, a factory resident Hardware Security Module (HSM) executes a secure provisioning protocol with the manufactured Trustonic TEE. During this process a new random *TEE binding key* is generated and provided to the TEE. The same key is also eventually transported to a back-end server farm together with the device's identity. This shared key conceptually is the (root) security domain and the trust root under which TA provisionining, attestation and revocation for the some 300 million already deployed Trustonic TEEs takes place.

B. Nokia (Microsoft) OnBoard Credentials

On-board Credentials (ObC) is a TEE architecture available in Nokia Windows Phone 8 and Symbian phones. The ObC architecture is illustrated in Figure 2 and further elaborated in [22].

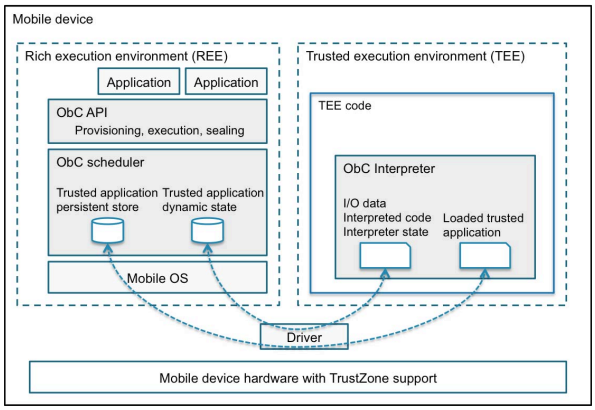

Fig. 2: On-board Credentials (ObC) architecture: TAs are executed within the TEE by the *ObC Interpreter*. The *ObC scheduler* maintains trusted application persistent storage and handles execution scheduling. REE applications use ObC services through the *ObC API*.

In the ObC architecture, a small virtual machine called *ObC interpreter* provides all the necessary TA isolations. Although this interpreter is not a complete operating system, it provides the runtime isolated environment for execution of TAs that originate from developers.

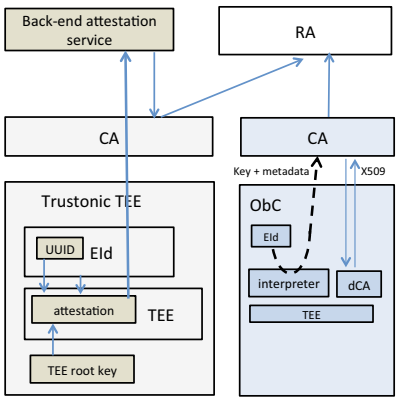

Fig. 3: Attestation mechanisms in Trustonic TEE and ObC

ObC TAs are developed either in BASIC or using a bytecode assembler. The ObC interpreter is implemented as a set of interacting TrustZone code components. Depending on the underlying TEE hardware and software architecture these components may permanently reside inside the TEE or can be loaded on-demand. In some environments they constitute the only available "trusted OS" in the device whereas in other deployments the ObC interpreter components are themselves TAs of another TEE.

ObC scheduler that runs in REE is responsible for scheduling between the ObC TrustZone components, as well as managing the encrypted state of the currently interpreted TA. Applications in REE access the ObC system through the *ObC API* (see Figure 2). To execute a TA, the scheduler loads the locally encrypted representation of the TA. Additionally, it loads possible inputs and stored data sealed by previous invocations of the same application instance. The ObC interpreter then executes the TA bytecode using on-demand code page loading of the bytecode. Back at the REE, the ObC scheduler re-invokes the same or different TA, using the same or different TrustZone component, attaching possible temporarily stored data or interpreter state. The bytecode execution continues in this manner until the TA completes.

The ObC platform supports an open provisioning model in which any developer, with the permission of a device user, can deploy TAs without requiring additional permission from a centralized authority, such as the device manufacturer or the OS provider. A device-specific public key, certified by the manufacturer, provides the basis for remote provisioning of TAs (and the secrets they need to operate on) to devices already in the field. Isolation between security domains inside the TEE is guaranteed by interleaving execution of the different security domains and the implementation of the local storage with distinct encryption keys for each domain.

VI. Platform Trust and Attestation

The trust in both of the examined TEE architecture is based on the isolation properties of the ARM TrustZone hardware as well as on the immutability of the TEE code itself. This is guaranteed by the secure booting of the platform, governed by the OEM (original equipment manufacturer). Both explored architectures also leverage OEM-provided trust roots to derive persistent, device-specific private key material. Therefore the confidentiality and integrity of platform keys (as well as the TEE-provided secure storage) ultimately ties back to OEM. Furthermore, the correctness of the possibly audited TEE code, by the TEE provider, affects the security level of the platform. In short, this part of the trust formation, sometimes called the "roots of trust", can be evaluated as a platform property. Therefore, the security level of the ESP constructed out of a TEE can then be compared to other forms of ESP, such as smart cards or network-based solutions.

Additionally, the ESP hosted on a TEE will require an extra set of trust anchors and protocols. Typically, a trustworthy manufacturer provisions physical smart cards on behalf of the token issuer. In contrast, TEEs are manufactured and put to market in a trust domain that normally does not include any priori relation to the service issuer. Thus, the trust between the ESP issuer and the TEE must be established at service provisioning time, rather than at manufacturing.

At large this is a problem of endpoint *attestation*, i.e. the service issuer must remotely authenticate the target TEE at the endpoint as well as verify the hardware and software setup accordingly. This includes e.g. the TEE version in use, its revocation status etc. Furthermore prior to data provisioning, the issuer must confirm that the secure channel it established with the endpoint is actually the correct service TA running inside TEE and not some other code running in the TEE or elsewhere at the end-user device.

Attestation protocols and related trust relationships can be constructed in many ways, and it is illustrative to note that our two targeted TEEs significantly differ in this respect. Since the attestation for service provisioning is crucial for the reliable deployment of EIds to TEEs, we present the two attestation mechanisms for our selected TEEs as depicted in Figure 3 in Section VI-B. But before that we will look at how TA identites are managed in the respective TEEs.

A. Program identity and separation

The separation of TAs in Trustonic TEE is based on a UUID (universally unique identifier) of a program (TA). The license to run a TA is provisioned by a Trusted Service Manager (TSM), and the uniqueness of UUIDs for programs is guaranteed by the provisioning TSM. Recent versions of the TEE also support origin-bound UUIDs. In this variant the UUID contains an RSA public key fingerprint of the issuer (origin) key, and the provisioning protocol (as enforced by the TEE) carries an additional attestation over the TA digest, signed with the UUID key. In essence, the mechanism implements same origin policy / authentication for UUIDs.

With the Trustonic TEE, the provisioning also includes a target-device specific license provided by the TSM. This license embodies the right to run the TA on the device. The license binds the TA UUID (either variant) to a TA-specific encryption and integrity protection key required for running the TA in the TEE. Therefore only the TA issuer with the knowledge of the encryption key can package and provision applications for the given UUID, and using these keys the integrity and confidentiality of the provisioned TA

binary is guaranteed. However, all storage and IPC (inter-process communication) separation in the TEE is based on the uniqueness of the UUID alone.

In ObC, the program identifier is the digest of the service's byte-code. All TA context separation – for storage, run-time and data provisioning – can be based based on this value, which is recomputed at every program load. This mechanism provides full TA-specific isolation, but there are cases where this isolation is too stringent. For storage and IPC, an alternative security context (selected by the TA programmer in a case-by-case manner) binds the isolation to an issuer-provided "family key", which was provided to the TA during provisoning. In this second security context, data delegation and TA-to-TA IPC can be done in an origin-controlled fashion. The first-mentioned program digest based separation always also includes the origin-key based aspect, which allows the same binary to operate in the same device on behalf of different issuer without a risk of information leakage.

B. TEE authentication and attestation

The authentication procedure in ObC is based on an X.509 device certificate, provided by the OEM. The OEM certifies a device-specific signature key during device manufacturing. The certificate also provides necessary device / TEE identifiers. The signature key is securely stored in the TEE, and governed by a "SubCA" entity. The ObC interpreter (and a few other TEE services) has access to the device SubCA through an internal attestation API. For TAs, access to the X.509 certificate generation by the SubCA is limited to an attestation interface, which restricts attestation to RSA key objects generated by the TA calling the attestation interface. In ObC the TA does not have access to the private RSA key component, only full usage rights. The private key object contains metadata primarily used for local access control. Parts of this information is also duplicated into the generated "subject key attestation evidence" (SKAE) structure. Among other information, the SKAE structure contains the code digest / identifier of the TA that requested the generation of the key, as well as the hash of the public key generated by the TA. The SKAE digest is then included in an extended certificate attribute during device SubCA-certificate generation.

The validation of the ObC attestation process includes verification of the certificate chain from the OEM CA to the RSA key generated by the TA (through the on-device "SubCA"). It also examines the SKAE to check if the leaf certificate contains the TA code digest. Further, it confirms that the key being certified is generated and accessible only to the intended TA inside the TEE in the device specified by the SubCAs public key certificate. This is confirmed by comparing the code digest in the SKAE against the code digest of the TA and validating a PKCS#10 proof-of-possession provided by the TA that proves the key access. Attributes for providing attestation freshness can be included in both the PKCS#10 request and in the attestation interface towards the SubCA.

In short, ObC ESP endpoint attestation is rooted in PKI, and therefore does not rely on a live third party connection during the attestation protocol, with the possible exception of checking the revocation list of the OEM CA, whether the particular SubCA certificate has been revoked.

The trust model for the Trustonic TEE is based on a network protocol, which involves a network-facing attestation server. We earlier learned that a potential attacker could not mislabel a TA in terms of UUID. The attestation process in the Trustonic TEE is a service interface provided by the system, which any TA can use to provide any data for attestation. Additional information such as the UUID of the TA requesting attestation as well as internal identities needed for identifying the end device to attestation servers are signed along with the data using symmetric device root keys. Furthermore, the attestation record is encrypted for the cloud endpoint [4] to provide privacy. The signed and encrypted blob is returned via the TA to the attestation service, which has the necessary key material to decrypt and validate the blob.

In the Trustonic attestation model, a service that wishes to attest a key generated in a TA provides a public-key digest to the attestation function. As the UUID is unique in the Trustonic TEE, this identifier (which will be part of attestation) will map the program to its origin, and to the verifier this must translate to a trust that the secret key component of the key being attested is properly handled by the TA.

In our ESP setup, the attestation blob is sent to the TA , which in turn shall transport this blob to a back-end network service to translate the attestation for the ESP program into a platform + TA + key X.509 certificate. The key that is being certified this way in the ESP TA will in turn be used to attest the ESP key during ESP enrolment.

VII. Architecture

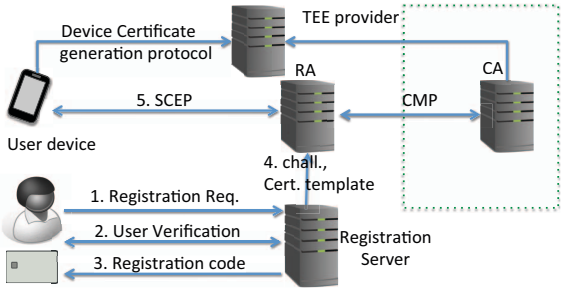

Fig. 4: EId registration process

In Figure 4 we identify the five entities involved in the mobile e-ID enrolment transaction: The end user, the point-of-registration server, the TEE provider, the registration authority (RA) and certification authority (CA). In our model, the user holds a device with a TEE of sufficient security qualifications to be acceptable by the identity system. The user must also hold a verifiable identity, such as a government-issued EId smartcard, or in case of a physical visit at a local registration point (e.g. police station), other necessary documents to prove his or her identity. The task of the point-of-registration server/service (remote or physical) is to verify the user identity

[4]The used back-end is provided by the TEE provider, and validates the attestation blob using key material that exists for all Trustonic TEE enabled devices in the field. A successful attestation proves the existence of the TEE in the endpoint and indirectly also that the attestation did take place within the confines of the TEE rather than elsewhere in the device.

and provide, to the RA, and authorized statement of this fact. In this paper we focus on remote registration. In this mode, the point-of-registration identifies the user remotely, and will provide one or several session credentials to the identified user over an out-of-band channel. The RA will expect these credentials to be provided from the mobile device in the next processing step. In our implementations we use QR codes for convenience – the user authentication will mainly happen with a smart card connected to a laptop or PC, and if the transaction succeeds, a QR code with the credential is displayed on the PC screen to be captured by the phone's camera.

The objective of the RA service is to validate the enrolment process. The input from the point of registration asserts the user identity. Next the RA must ascertain the security level of the ESP (TA/TEE) that has generated the to-be certified credential, as well as the Proof-Of-Possession (POP) of the credential itself. For the latter a signed PKCS#10 certificate signing request is used. The former will be composed of a) evidence from the TEE/TA attestation process signed by b) the trust root used for authenticating the TEE (typically the OEM or the TEE provider).

We require that the result of the attestation process can be distilled into a TEE (type) identity, and a related verification blob that the RA by itself, or by consulting a trusted third party, can use to determine and validate protocol freshness, the endpoint identity as well as its security level. In the enrolment protocol, the mobile device sends the POP and attestation evidence in session-key encrypted PKCS#10 attributes inside the protocol request. This way of operating hides the identity parameters used for attestation from network eavesdroppers. In terms of addressing we assume that the TEE or an enrolment application in the mobile device knows the network address of the RA and can initiate the RA session after receiving the session credential (as transfered e.g. with the QR codes) from the point-of-registration server.

When the RA has performed all needed identity verifications and attestation clearing, it will forward the certification request (using CMP) to the CA, that is the trust root that signs the user certificate for the key generated under the protection of the TEE. The user device receives the certificate that is included in the return message of the enrolement.

A. The EId (enrolment) Trusted Application

For using the EId in his device the user downloads and installs the mobile EId application from an application store. The EId app consists of two parts: a client part and the TA part. The client part runs in the rich world and provides a regular user interface as well as network communication. While TA runs inside TEE and performs all cryptographic operations for ESP. The functionality of the ESP-TA is close to that of PKCS#15 secure element or a PKCS#11 token. On request, the TA will generate an asymmetric keypair, and persistently store the secret part of the key. The TA shall also be able to bind the generated key to PIN and PUK protection. For this, the TAs developed in the context of this paper follow a pattern outline in Section VII-B. Additionally the TA and TEE must be able to support some **attestation** protocol to bind the generated key to the TEE – this is needed for the RA to determine that the key being certified is present and protected by the TEE. For this, we leverage the protocols described in Section VI.

Furthermore, the TA shall provide methods for selectively engaging keys it manages by providing the corresponding public key components on demand. The use of the private key for signatures (and possibly for decryption) is tied to appropriate PIN/PUK protection. Finally, either the TEE or the TA itself should provide key lifecycle support, such as disabling the credential in case the device needs to be serviced or deleted, or if the device is given to another user. In line with contemporary key strength requirements, the RSA keys used should be at least 2048 bit long, whereas a suitable ECC key length is 256 bits.

B. PIN creation and PIN validation

Nyman et. al. [15] list the binding requirements for PINs with ESPs. With our TAs, the REE app configures user PINs for TA operations. During PIN configuration a $PINinfo$ object is initialized. The $PINinfo$ objects consists of the information related to the PIN such as the PIN itself, the PUK, the TA's application UUID, the maximum number of PIN retries before locking TA operations, a list of keys (key hashes) associated with the PIN (the PIN to key mapping is one-to-many) as well as the value of a TEE-specific replay protection counter managed by $PINinfo$.

During $PINinfo$ initialization, the user provides a 4-digit PIN along with a PUK. The PIN configuration process may happen multiple times during the lifetime of the EId TA, but the PUK remains fixed. The maximum number of PIN retries is predefined by the ESP-TA.

During PIN validation, the EId client application (or a Trusted UI, achievable with some Trustonic TEE) supplies the user-entered PIN, the sealed PINinfo blob, the sealed private key along with the selected key identifier to the TA. The TA first unseals the blobs and checks the number of remaining PIN retries. In case if the PIN retry has exhausted the TA refuses to cooperate. Otherwise, the TA continues to match the PIN and verifies the key associated with the PINinfo against the supplied key identifier before performing any operation with the key.

C. Enrolment

We rely on the **Simplified Certificate Enrolment Protocol** (SCEP) [5] for the ESP enrolment. SCEP was originally designed for closed networks to simplify the certificate enrolment process on networking devices such as routers or base stations. As a rule network devices do not have identities that a CA can understand or verify using a public key. SCEP relies on shared secret or "password" for authentication, which is well suited for bootstrapping the mobile ESP enrolment from another identity verification mechanism, such as a smart card authentication. Step 1 to 3 in Figure 4 depicts the bootstrapping process. Additionally, it can equally be used as part of physical identification during a "service point" visit.

Today, SCEP is the most popular and widely available certificate enrolment protocol. The alternatives to SCEP are Certificate Management Protocol (CMP)[6] and Certificate Management over CMS (CMC)[7]. However, these alternatives are

[5]https://tools.ietf.org/id/draft-gutmann-scep-00.txt
[6]https://tools.ietf.org/rfc/rfc4210.txt
[7]https://tools.ietf.org/rfc/rfc5272.txt

complex and do suffer from interoperability problems and lack of industry support.

SCEP supports the following general operations: Certificate enrolment, Certificate renewal/update, Certificate queries and CRL queries. SCEP makes extensive use of CMS[8] and PKCS#10 [9]. The SCEP message syntax is based on CMS, i.e., the PKCS#7 format, in which SCEP attributes are sent as signed attributes and the SCEP payload as encrypted data (the message payload is e.g a PKCS#10 request the certificate response).

A typical SCEP message flow goes as follows: The requester receives a registration code from the RA using out-of-band channel (Step 1-3 in Figure 4). The requester then prepares the certification request (CSR) to the server (Step 5 in Figure 4). It must authenticate the enrolment request either by attaching the registration code or by using an existing valid certificate. Typically a registration code is used, and this is inserted in the PKCS#10 request. The requester then sends the request to the RA. The RA validates the registration code. It is recommended that the user identity be mapped into certificate request template only at the server side to mitigate the threat of requester impersonation attempts (step 4 in Figure 4). Finally, the request is passed to a CA, which issues a certificate according to its policies based on the certificate template. The X.509 certificate is returned to the requester, which will store the certificate for later use. Martinelli et. al. presented a formal analysis of the SCEP protocol in their work [23].

With ESP key enrolement using SCEP, we need a way to attestation evidence to the RA. For this we add a new attribute, "proofOfOrigin", to the PKCS#10 request inside the SCEP enrolment request. If stated in the CAs policy statement, the servers (RA+CA) will evaluate this attribute and assert the CA's trust in the user's TEE before issuing an EId certificate for it.

VIII. Implementation

We have implemented a reference setup consisting of a *registration authority* and a *registration service*. The *registration authority* interacts with mobile clients over SCEP where as the *registration service* bootstraps the user identity from a government-issued smart card EId (ESP) to the mobile ESP in one of our supported TEEs. We have client applications for the Windows (ObC platform) and Android (Trustonic TEE) as well as their respective TAs. In the Windows Phone case (ObC), the attestation evidence for the TEE-hosted ESP is generated locally using the SubCA on the device. It is then sent as part of the certification request to the RA for validation.

The user interface to our ESP enrolment process is shown in Figure 5. The user goes to an enrolment portal, to which he logs in with his government-issued EId card. Once his identity is verified successfully, the portal produces a SCEP registration code in the form of a QR code. The mobile EId application then scans the QR code, generate the ESP keys and, if needed, query the user for a PIN/PUK that will be

[8]https://tools.ietf.org/rfc/rfc5652.txt
[9]https://tools.ietf.org/rfc/rfc2986.txt

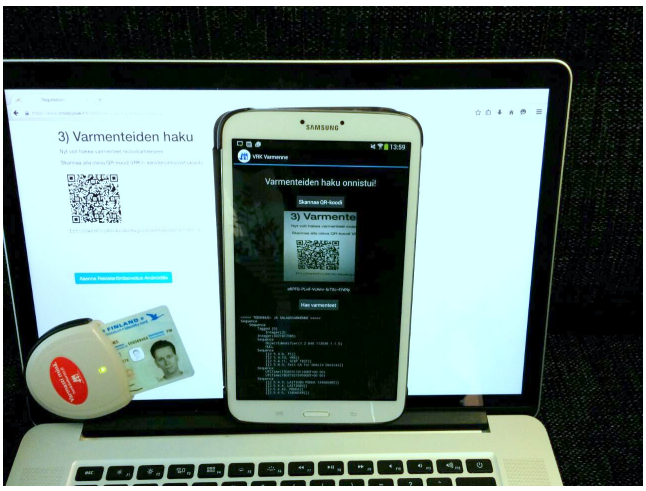

Fig. 5: proposed EId system in use

used to protect ESP operations. Then, the application, with the TEE, performs the attestation transaction as well as the generation of the proof of possession of the key. Finally, it authorizes the SCEP request to the RA with the registration code. The registration service did send the registration code together with the customer certificate details to the RA when the user visited the enrolment portal. These data are kept active in the RA for certain time, e.g. 10 minutes, to allow the user to complete the transaction. Based on this information the RA can securely map the forthcoming SCEP request from the user mobile device to an identity. After the mobile device attestation is verified and accepted, the RA populates required identity information into the CMP (certificate management protocol) request to the CA. In our case, the CA for Mobile EIds is a test CA.

After successful key enrollment, the EId application "becomes" the identity service to other applications in the mobile device. We use the inter-application communication frameworks available in the respective mobile devices to achieve this. While ESP in PCs are typically access via a PKCS#11 interface or OS specfic APIs (CryptoAPI in Windows, KeyChain in MacOS), in Android we emulate the essence of an Android keystore interface (KeyStore) for EId use from other applications (that need to be aware of this TEE based keystore). On Windows Phone, we haven't implemented the actual usage of the ESP scenario but our EId signing interface would resemble a browser MIME type activation. The user experience is much like in the PC – when the application requests for a signature generation, the user is presented with a certificate selection interface. When the user selects a certificate from the list, the PIN is requested from the user before signing the request using the corresponding key.

In Table: I, we compare TEE-based ESP use with traditional smart cards. We note that the transaction speed of our mobile ESPs are significantly better than that of a traditional smart card, a detail that might be relevant for use cases such as ticketing or e-mail signing.

	Avgerage transaction time
Smartcard (Finnish EId)	
external USB reader	3242 ms
Smart card (Finnish Organizational Card)	
external USB reader	1165 ms
Nokia Lumia 1520	
using standard Crypto library	213 ms
ObC API	120 ms
ObC Script (sign + PIN vrf)	260 ms
Samsung Galaxy Tab 3	
AndroidOpenSSL / SC	23 ms
Trustonic TEE	237 ms
Google Nexus 7 (1st gen)	
AndroidOpenSSL / (SW)	29 ms
AndroidKeyStore / (HW)	707 ms

TABLE I: EId signing time on various platforms

IX. Conclusions

TEEs are common place today, and ObC and Trustonic TEE together cover some 0.5 billion deployed devices worldwide. Given that many ESP use-cases, as well as consumer preferences in general are increasingly "going mobile", ESP deployment on handsets and tablets needs to be addressed urgently. This paper provides a practical design for EId deployment on mobile phone TEEs, and with this paper we provide real-world implementations and test results for ESP enrolment and use in on-the market handsets.

A related, highly relevant, design issue is how to devise the API for ESPs in the HTML5 / JavaScript domain. A coherent "EId" interface for web interaction could make ESP use seamless across mobile and PCs, independently of the trust fundament for the ESP. For that to happen, we must also consider attestation as an important element of that very interface, and this is one of the future directions of this work.

References

[1] Board of Governeors of the federal reserve system, "Consumers and Mobile Financial Services Report," 2014. [Online]. Available: {http://www.federalreserve.gov/econresdata/consumers-and-mobile-financial-services-report-201403.pdf}[AccessedApril2015]

[2] T. Martens, "Electronic identity management in estonia between market and state governance," *Identity in the Information Society*, vol. 3, no. 1, pp. 213–233, 2010.

[3] T. Mahler, "Governance models for interoperable electronic identities." *Journal of International Commercial Law & Technology*, vol. 8, no. 2, 2013.

[4] T. Myhr, "Legal and organizational challenges and solutions for achieving a pan-european electronic id solution: or i am 621216-1318, but i am also 161262-43774.11 these are my national registration number in sweden and norway. cf. footnote 11. do you know who i am?" *Information security technical report*, vol. 13, no. 2, pp. 76–82, 2008.

[5] The Act on Strong Electronic Identification and Electronic Signatures (617/2009). [Online]. Available: http://www.finlex.fi/fi/laki/kaannokset/2009/en20090617.pdf

[6] Regulation (EU) No 910/2014 of the European Parliament and of the Council of 23 July 2014 on electronic identification and trust services for electronic transactions in the internal market and repealing Directive 1999/93/EC. [Online]. Available: http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=uriserv:OJ.L_.2014.257.01.0073.01.ENG

[7] Bundesamt für Sicherheit in der Informationstechnik BSI TR-03112 Das eCard-API-Framework. [Online]. Available: https://www.bsi.bund.de/DE/Publikationen/TechnischeRichtlinien/tr03112/index_htm.html

[8] e-SENS (Electronic Simple European Networked Services). [Accessed: April 2015]. [Online]. Available: http://www.esens.eu/home/

[9] The Secure idenTity acrOss boRders linKed (STORK). [Accessed: April 2015]. [Online]. Available: https://www.eid-stork.eu

[10] F. Brasser, S. Bugiel, A. Filyanov, A.-R. Sadeghi, and S. Schulz, "Softer smartcards," in *Financial Cryptography and Data Security*, ser. Lecture Notes in Computer Science, A. Keromytis, Ed. Springer Berlin Heidelberg, 2012, vol. 7397, pp. 329–343.

[11] J. Vossaert, J. Lapon, B. De Decker, and V. Naessens, "Trusted computing to increase security and privacy in eid authentication," in *ICT Systems Security and Privacy Protection*, ser. IFIP Advances in Information and Communication Technology, N. Cuppens-Boulahia, F. Cuppens, S. Jajodia, A. Abou El Kalam, and T. Sans, Eds. Springer Berlin Heidelberg, 2014, vol. 428, pp. 485–492.

[12] S. Tamrakar, J.-E. Ekberg, P. Laitinen, N. Asokan, and T. Aura, "Can hand-held computers still be better smart cards?" in *Trusted Systems*, ser. Lecture Notes in Computer Science, L. Chen and M. Yung, Eds. Springer Berlin Heidelberg, 2011, vol. 6802, pp. 200–218.

[13] Bundesamt für Sicherheit in der Informationstechnik BSI TR-03124-1 eID client Part1: Specification, Version 1.2. [Online]. Available: https://www.bsi.bund.de/DE/Publikationen/TechnischeRichtlinien/tr03124/index_htm.html

[14] F. Morgner, D. Oepen, W. Mller, and J.-P. Redlich, "Mobile smart card reader using nfc-enabled smartphones," in *Security and Privacy in Mobile Information and Communication Systems*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, A. Schmidt, G. Russello, I. Krontiris, and S. Lian, Eds. Springer Berlin Heidelberg, 2012, vol. 107, pp. 24–37.

[15] T. Nyman, J.-E. Ekberg, and N. Asokan, "Citizen electronic identities using tpm 2.0," in *Proceedings of the 4th International Workshop on Trustworthy Embedded Devices*, ser. TrustED '14. New York, NY, USA: ACM, 2014, pp. 37–48. [Online]. Available: http://doi.acm.org/10.1145/2666141.2666146

[16] A. Dmitrienko, A.-R. Sadeghi, S. Tamrakar, and C. Wachsmann, "Smarttokens: Delegable access control with nfc-enabled smartphones," in *Trust and Trustworthy Computing*, ser. Lecture Notes in Computer Science, S. Katzenbeisser, E. Weippl, L. Camp, M. Volkamer, M. Reiter, and X. Zhang, Eds. Springer Berlin / Heidelberg, 2012, vol. 7344, pp. 219–238, 10.1007/978-3-642-30921-2_13.

[17] P. Verhaeghe, J. Lapon, V. Naessens, B. De Decker, K. Verslype, and G. E. Nigusse, "Security and privacy threats of the belgian electronic identity card and middleware," 2008, status: published.

[18] C. J. Dietrich, C. Rossow, and N. Pohlmann, "eid online authentication network threat model, attacks and implications," in *19th DFN Workshop 2012*, 2012.

[19] I. Naumann, "Privacy and security risks when authenticating on the internet with european eid cards," *ENISA, Risk Assessment Report*, 2009.

[20] ARM. (2009, Apr.) ARM security technology – building a secure system using TrustZone technology. [Online]. Available: http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.prd29-genc-009492c/index.html

[21] J. Liedtke, "Toward real microkernels," *Commun. ACM*, vol. 39, no. 9, pp. 70–77, Sep. 1996.

[22] K. Kostiainen, J.-E. Ekberg, N. Asokan, and A. Rantala, "On-board credentials with open provisioning," in *ACM Symposium on Information, Computer and Communications Security (AsiaCCS)*. ACM, 2009, pp. 104–115.

[23] F. Martinelli, P. Marinella, and V. Anna, "Automated analysis of some security mechanisms of scep," in *Information Security*, ser. Lecture Notes in Computer Science, A. Hui Chan and V. Gligor, Eds. Springer Berlin Heidelberg, 2002, vol. 2433, pp. 414–427.