Credential Remote Management

Laura Marcia Villalba Monne
Helsinki University of Technology
marcia.villalba@gmail.com

Abstract

Software credentials are vulnerable to many attacks and existing approaches to address secure storing and usage fall short of providing an appropriate solution to these problems. User memorizable passwords suffer from bad usability and security. By contrast, dedicated hardware tokens provide better security but they are costly to produce and distribute and also suffer from bad usability. General purpose secure hardware, like TPM and M-Shield have become available recently in many commodity devices. These platforms allow a strongly isolated TrEE and allow the deployment of credential platforms as On-board Credentials. This paper describes how to extend the On-board Credentials to support credential remote management.

Keywords: ObC, Credential Platforms, Trusted Computing, TrEE, Credential remote management

1 Introduction

Credentials can be used for different security relevant tasks such as authentication or providing confidentiality. For example, in information systems credentials are used to authenticate the users to the system. A classic authentication mechanism is the combination of user name and password. The password authentication mechanism is popular because it is easy to use, inexpensive and flexible. Software only authentication mechanisms, such as passwords, do not require any extra device or hardware to function, therefore they are cost efficient to implement. Furthermore, passwords are a flexible authentication mechanism, they can be implemented in almost any service and users are familiar with them.

However, passwords have some drawbacks: they suffer from bad usability and they are vulnerable to phising and malware. Instead of passwords, it is possible to use hardware token authentication mechanisms, which are a physical token used to prove one's identity electronically. These tokens are more secure than passwords, because they provide two-factor authentication. Two-factor authentication means that to authenticate a user to the system, the user has to provide two different qualities about himself. These two qualities have to belong to this list: something that the user knows (e.g. password), something the user has (e.g. bankcard) or something the user is (e.g. biometric).

However, hardware tokens have disadvantages. Firstly, the logistics of manufacturing and distributing are expensive, making this unattractive to the service providers. Secondly, these tokens are inconvenient for the users, because typically there is one hardware token per service (multiple applications per token exist but they are not popular), so the user has to carry one token for each service.

In the last two decades several types of general purpose secure hardware have become available in many commodity devices, such as TPM [15], MTM [4], M-Shield [5]. These platforms enable a strongly isolated secure environment. These platforms enable a strongly isolated secure environment and are called Trusted Execution Environment (TrEE).

Due to the existence of TrEE, credential platforms became possible and attractive. A credential platform is a framework used for storing and using credentials such as, TEM [2], ObC [7], etc.

A secure authentication mechanism provides trust to both sides of the communication; to the user it gives the security that no one can impersonate him, and to the service it gives the security that the user is who he says he is. One example of the usefulness of authentication mechanisms is in using online banking. One way to use online banking currently is using *one time passwords (OTP)*. OTP are passwords that are only valid for a single authentication session. Nowadays, some banks give their customers a paper card or a device to generate OTP. These OTP combined with a pincode (two-factor authentication), are used every time the customers log into the system.

Even though online banking using these authentication mechanisms is secure, it is still possible that the customer's paper list or device can be stolen or lost. Also, these mechanisms are not easy to use, because customers have to carry them whenever they may need to use the service. Furthermore, they are expensive, because the bank has to print and mail all of the paper cards at least two times a year or distribute the device to all of its customers. Therefore the problem is how to make online banking more secure, user friendly and cost efficient?

One solution is to use OTP but in a TrEE. Therefore, instead of receiving a paper list of OTP or a device, the customer will receive in his computer / mobile phone (equipped with a TrEE) a program to create the next OTP or a virtual list with a number of OTP.

The above means, the bank can send to their customers, more cheaply an algorithm or list, and the user can run it in his mobile phone or computer, freeing the user from carrying unnecessary gadgets. However, this solution creates a new problem for the banks, how to manage all of their customers' credentials? How to register, revoke or modify them? Usually banks have thousands of clients and managing them is hard to do manually. Therefore, the system needs to manage of all these credentials in a remote and automatic way.

Credential management is a fundamental issue for this type of system, because the credentials that the devices are using were provided by a third party, and the provisioning entity (in this case the bank) has to be able to manage its customers' credentials. This procedure of remotely managing customer credentials, in this paper will be called credential remote management.

The main functionalities that a credential remote management system should include are: provisioning of the credentials, renewal of the credentials, managing of the credentials and revoking of the credentials. Beside these functionalities it has to be possible to grant new rights to existing credentials or to have a system that allows the migration of credentials to other devices. All of these activities have to be done remotely, and if possible automatically.

This paper describes an extension to a credential platform, On-board Credentials, to support credential remote management. This was a research made by Villalba [11]. Section 2 and 3 provide background information about different TrEE and credential platforms. Section 4 explains how the scheme was designed and a security analysis. The last section provides some conclusion for this research.

2 Background

2.1 Trusted Execution Environments

This paper uses the definition from Kostiainen et al. [7] for general purpose secure environment or *Trusted Execution Environment (TrEE)*. TrEE is defined as a system with the following features:

- **isolated secure execution**, this means that is possible to execute trusted code isolated from untrusted code executing in the same device,
- **secure storage**,this means that it must be possible for the trusted code to store persistent data in order to maintain the confidentiality and integrity of the data,
- **integrity of TrEE**, this means that there is a way to ensure the integrity of the TrEE.

In the last decade different types of TrEE have been incorporated into the computing devices and are starting to be widely deployed. These include Trusted Platform Module (TPM) [15] and Mobile Trusted Module (MTM) [4] both of which were specified by TCG. In addition, there are other platforms like M-shield [5] and ARM Trust Zone [16] for mobile devices.

- **TPM**: One of the most popular TrEE is *Trusted Platform Module (TPM)* specified by TCG. TPM is a separated hardware module with its own processor. In other words, TPM implementation is a chip that is attached to the motherboard and controlled by the operating system. TPM provides cryptographic operations with asymmetric key such as, generation, decryption, encryption, signing and, migration of keys between TPMs. Furthermore, it enables secure storage and authenticated boot. However, TPM TrEE has to rely on the operating system to provide secure execution [7]. TPM supports Authenticated boot, it is a process by which integrity measurements are reliable measured and stored securely but not checked, this is a passive method. This provides to a third party a proof of the configuration initialization via attestation [14].
- **MTM**: On mobile platforms TPM is called *Mobile Trusted Module (MTM)*. Even though MTM specification is closely tied to the TPM, MTM differs mainly from TPM in these issues [4]:
  1. **Implementation of MTM:** This means that MTM can be implemented in software as well as in a physical implementation of hardware.
  2. **Support of parallel MTM:** The new specification support several parallel MTM instances in the same device.

  MTM secure boot procedure is different form the TPM. Secure boot is used to enforce integrity protection, each time the device boots, the boot sequence is measured and aborted if an non-approved state is reached [4].
- **M-shield**: M-shield was developed by Texas Instruments and it is an example of a mobile TrEE. The main difference between M-Shield and TPM is that in M-shield is possible to execute arbitrary code while in TPM only cryptographic operation are available. M-Shield is contained in the main CPU and has two modes of operation: normal mode and secure mode. Additionally, M-Shield has a secure storage and it ensures the integrity of the TrEE, only trusted (signed) code is executed within M-shield [5].

  In M-shield the code is called protected application, the execution of it is isolated from the operating system side because there is on chip ROM and RAM. However the code and stack size is very limited, around 10-20 kB.

  Nowadays Nokia mobile phones with M-Shield hardware exist. This is the platform used by the On-board Credential [7] (See Section 3).
- **ARM TrustZone**: ARM TrustZone is a security extension to the ARM processor core. In TrustZone is possible to execute arbitrary isolated code, like in M-Shield. TrustZone introduces the concept of "secure world" and "non-secure world". This means that there are registers, interrupts flags and other system control registers that exist in a separate secure version, totally inaccessible from the non-secure world [16]. Therefore, the security in TrustZone is achieved by partitioning the hardware and the software resources in two worlds, however these two worlds execute in one physical processor core using time-slicing. The context switch between the two worlds is called the monitor mode.
- **Java Cards**: A smart card is a portable and tamper-resistant computer, that carries processing power and information. The Java Card contains a version of Java virtual machine that is split into two parts: one that runs off-card and the other that runs on-card. The main design goals of the Java Card technology are the portability and security. Java Card was developed for securing sensitive information stored on smart cards. It

allows secure and isolated execution of applications, it also supports commonly used cryptographic algorithms. Objects in Java Card are stored in persistent memory and the objects stored by one application are not available to other applications [1].

2.1.1 Comparison of TrEE

Table 1 summarize the main features of the different TrEE platforms.

2.2 Credential Platforms

A credential platform is a framework for storing and using credentials. It is implemented using one of the TrEEs described before and can also utilize some OS platform security features.

- **TEM:** The *Trusted Execution Module (TEM)* [2] is a credential platform, that was designed for low - resource environments of inexpensive commercially-available secure chips. TEM can execute any type of credential and it guarantees the confidentiality and integrity of both the computation process and the information it consumes and produces. Furthermore, TEM does not trust the authors of the programs it runs, therefore a malicious closure cannot impact negatively on the TEM and on the other closures that executes in the same system. This results that there is no need for certificate the third party closures.

  A closure is the execution primitive of TEM together with the binding of the variables that were in scope when the closure was defined. To attest that the platform offers a security platform, when the device is manufactured a unique asymmetric key is installed in the device, signed by the manufacturer that is the CA. The private part of the key is generated inside the TEM and never leaves the TrEE, and the public part is included in an endorsement certificate.

- **TruWallet:** TruWallet is a wallet based secure web authentication credential platform, this means that TruWallet only deals with web login credentials. This solution according to Gajek et al. [3] provides: (1) Strong protection for user credentials and sensitive data with TPM. (2) Automated login procedure where the server is authenticated independently from SSL certificates. (3) A secure migration protocol for transferring data to other protocols. TruWallet architecture is based on a secure kernel, which is a small trusted computing software layer, providing trusted services and isolated compartments.

- **Flickr:** Flickr architecture that isolates sensitive code execution using a minimal TCB. This means that Flickr can run arbitrary code isolated, however the size of the code is minimal. Flickr can execute at any time and does not require a new operating system, this means that the platform for non-sensitive operations remains unchanged. Also, none of the software executing before Flickr begins can monitor or interfere with Flickr code execution, and all traces of Flickr code execution can be eliminated before regular execution resumes [12].

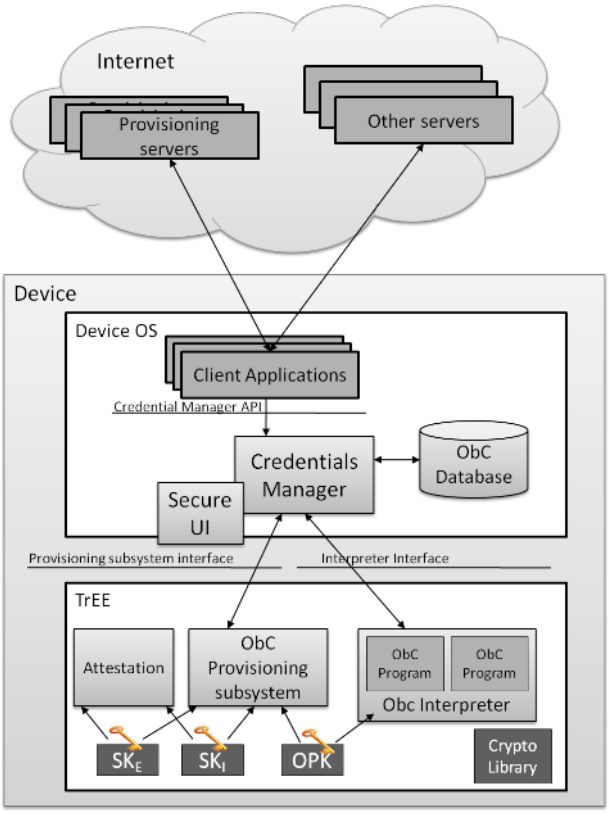

Figure 1: ObC Architecture [7]

- **SKS:** The *Secure Key Store (SKS)* is a credential platform, enhanced for smart cards and optimized for online provisioning of cryptographic keys, this means that SKS is limited to execute standard key pair operations. One of the main characteristics of SKS is the capability to emulate a set of independently issued and managed smart cards using only online communication. The main goal of SKS is to establish two-factor authentication as a viable alternative for any provider [13].

2.3 Comparison between credential platforms

Table 2 summarize the main features of the different credential platforms.

3 On-board Credentials

On-board Credentials (ObC) [7] is an architecture for provisioning, storage and execution of credentials, that uses secure general purpose hardware (TrEE).

Before starting with the architecture is important to define what a *credential* is for ObC, it consists of *credential secrets* such as keys, and an algorithm that operates on these secrets known as a *credential program* [7].

One of the main goals in the ObC deployment is that the system should be open, this means that is possible for anyone to develop and deploy a new credential type or ObC program,

	TPM	MTM	M-shield/TrustZone	Java Card
Hardware Support	Separate hardware	Software and hardware	Normal CPU	Removable hardware
Boot	Authenticated boot	Secure boot	Secure boot	None
Execute TrEE code	No	No	Yes	Yes
Storage	Secure Storage	Secure Storage	Secure Storage	Secure Storage
Attestation	Yes	Yes	Device specific propietary solutions	No

Table 1: Comparison of TrEE

	TEM	TruWallet	Flicker	SKS	ObC*
TrEE	JavaCard (or similar)	TPM	TPM	JavaCard	M-shield (or similar)
Secure Storage	Yes	Yes	Yes	Yes	Yes
Attestation	Key	Key	Code	Key	Key
Secure execution of arbitrary code	Yes	No	Yes	No	Yes
Provisioning	Yes	User inserts	No	Yes	Yes
Credential Migration	Yes	Yes	No	Yes	No

Table 2: Comparison of credential platforms (* Obc is presented in Section 3.)

or provision secrets to existing ObC programs without having to obtain permission from a third party. In addition, it is important that the openness do not compromise the security of the ObC system [7].

Nowadays ObC is available in research phones and Nokia N8 will be the first phone in the market that support ObC. It will be available in the majority of Symbian 3 based Nokia phones and newer.

3.1 Architecture

Figure 1 shows a high-level overview of the ObC architecture, where it is possible to see two very different environments: the device OS and the TrEE. The main components in the TrEE are: the ObC Interpreter and the ObC platform key. While in the device OS the Credential Manager is its most important component. These two environments are inside the device, additionally in the Internet it is possible to find provisioning and other kinds of servers.

The *ObC Interpreter* is the core of the ObC platform and it isolates credential programs from the TrEE resources. The Interpreter is a simple byte code Interpreter that only uses less than ten kilobytes of runtime memory. This Interpreter executes a modified subset of Lua [8] and assembly scripts. The trust of the Interpreter is based in code signing and it provides a virtualized environment where the credential programs can be executed. Furthermore, the Interpreter provides isolation of the secrets, sealing and unsealing of the secure data, and common cryptographic primitives.

The *ObC platform key*, (OPK) is part of a group of device keys, these keys are used by ObC for key generation and attestation. OPK is the device specific master key, it is a symmetric key that was installed in the device at the moment of its manufacture. The ObC Interpreter has access to it and this key will never leave the TrEE.

In addition to OPK the other two asymmetric device keys are: the *internal device key* (PK_I and SK_I) and the *external device key* (PK_E and SK_E). The first of these key pairs is used by the Interpreter to sign data that originates within the TrEE or which semantics can be verified within the TrEE and the public part of this key pair is certified by the manufacturer as the internal device key. The second one is used by the Interpreter to sign data that originates from outside of the TrEE, this key usage is limited within the Credential Manager using mechanism provided by the OS security, furthermore the public part of this key pair was certified by the manufacturer as the external device key.

The *Credential Manager* is a trusted operating system level component that belongs to ObC platform, it controls how client's application accesses the TrEE. The Credential Manager also maintains the ObC database and it provides a simple API towards applications to hide complexity of the executions inside the TrEE.

3.2 Provisioning

As was mentioned before, one of ObC main objectives is to allow openness, this means that any entity can provision secret data to a group of credentials or programs on the device. In other words, any provisioner can provision credentials secrets to a group of credential programs on a device.

A *family* is an important concept in ObC, a family is a group of programs and the secret data that they share. Credentials programs belonging to the same family may share sealed and persistent stored data, in addition all the programs and secrets are provisioned using the same *family key (FK)*, a random 16-byte key that is created at the provisioning side.

For example, is possible that two credential programs (prog1 and prog2) want to use the same secret (sec1), to achieve this prog1, prog2 and sec1 must belong to the same family. The provisioning server will be the one determining this.

Sensitive data (secrets and confidential programs) is stored outside TrEE encrypted in the Credential Manager database.

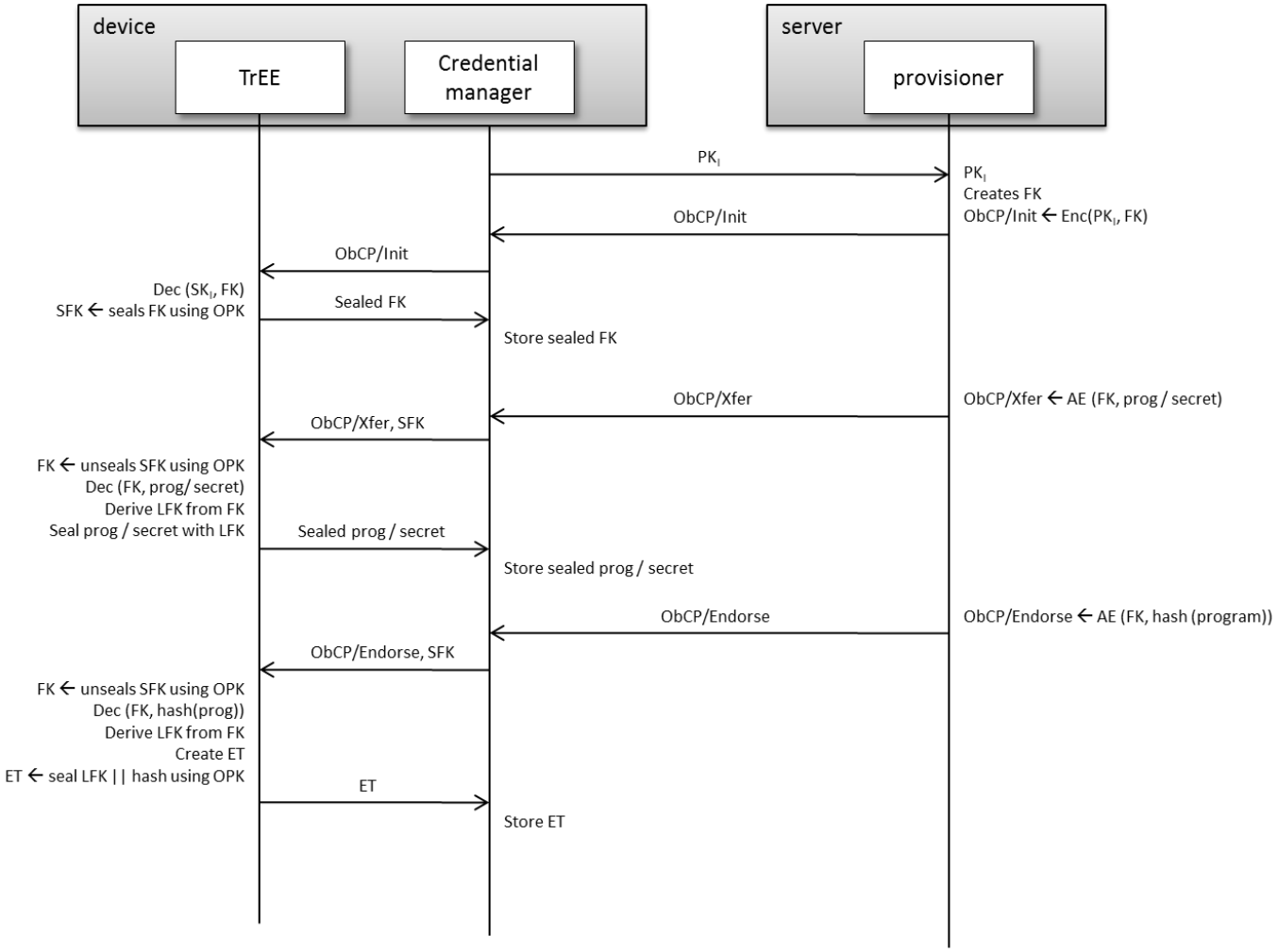

Figure 2: Provisioning in ObC

The operation of converting the data to the local storage format is called sealing and the complement operation is unsealing. To seal and unseal the sensitive data the OPK is used.

The provisioning process can be seen at Figure 2, it starts with the device sending the PK_I to the provisioner server. Then the provisioner server picks the FK randomly and encrypts the FK with the PK_I, and the result is the *ObCP/Init* message. This message is used to begin the family provisioning. The goal of this message is to transfer the FK from the provisioning server to the device.

When the FK arrives to the device, the Credential Manager sends this package to the TrEE, and there using SK_I decrypts the FK. The TrEE contains a limited storage capacity, so the FK has to be stored in the insecure environment, for this the TrEE seals the FK using the OPK.

The FK is used as a symmetric key between the device and the provisioner. The provisioner performs an authenticated encryption to all the future packets with the FK. The secrets and confidential programs are provisioned with the transfer messages, denoted as *ObcP/Xfer*. These packages contain a secret or a program encrypted with the FK.

When the device receives the ObcP/Xfer package, it sends it to the the TrEE and there using the FK it will be decrypted it. To store a secret or a program in a secure way, the *LFK or local family key* is derived from the family key, and used to seal this program / secret.

In the context of a family, a provisioner can authorize a credential program to be able to access provisioned credential secret. To do this, the endorse package is used. This package contains the hash of the byte code of the program that is endorsed, and the endorsement will apply to the family whose family key the endorsement package is encrypted with. The endorse packet is denoted as *ObcP/Endorse*.

The device when it receives the package will decrypt it using the FK and will create an *endorsement token (ET)*, this token will be sealed using the LFK from the secret, so it means that this program is authorized to access this secret.

It is also possible that the program and the secret come from different sources. To solve this, the provisioner of the secret, endorse the program using an endorsement package. After sending this package to the TrEE the program is able to use that secret.

This protocol does not provides user authentication, this must be handled separately, for example using TLS.

3.3 ObC API

The ObC Credential Manager not only provides an architecture but also provides an API to create applications in Symbian C++ using the ObC features. The information of this section was taken from [6], [9] and [10].

The ObC API provides three types of functionality:

- **Device initialization:** The ObC platform requires an initialization phase before it can be used to its full extent. The initialization means the creation of the device keys and the execution of the device key certification protocols.
- **Symmetric credentials:** In ObC architecture a symmetric credential consists of two separate parts:
  - Functions for adding provisioned secrets and programs.
  - Creating credentials and running them.
- **Asymmetric key pairs:** The ObC API also supports user created asymmetric key pairs. The key pairs are always generated locally and the private part never leaves the TrEE. It is possible to do some operations with the key pairs such as, sign message, verify signatures, decrypt and encrypt text.
- **Local access control policy :** This policy is used to determine which applications can use the credential, this is implemented using the AppAuthKey. This parameter can be shared with other applications, so they can access the credentials.
- **User access control policy:** This policy is used to determine when ever the user has to authenticate to the system for example by writing a secret PIN.

4 Credential remote management design

Credential Remote Management (CRM) is a scheme that allows third party entities to remotely manage credentials. This means that a third party entity can provision, revoke, update, etc. credentials that they own in their customers devices over any remote connection. The following section is extracted from Villalba's thesis [11].

Designing an CRM for the provisioning secrets from a server, registration of key pairs, updating secrets and deleting of credentials is not useful. This is because the provisioning credentials is already designed and implemented in credential platforms, the update of keys is basically a new provision or a new registration of the keys and the deletion of keys is not needed because deleting the access of the credential in the relaying server performs the same task. In addition, this analysis revealed that there is a need for updating the credential parameters.

Therefore, this section details how the CRM of credential parameters can be achieved. Nowadays, there is no complete tool that provides management of credential parameters in a remote way. There are some credential parameters that are relevant to the owner entity of the credentials to keep their credentials updated and to be able to manage them in an automatic way.

To explain in a clearer way how the CRM scheme works, this section presents an example that shows how the CRM scheme is applied to a real life scenario. Then this section explains in detail the protocol designed.

4.1 Example case

This example illustrates a possible usage for CRM. In this example there are two entities: provisioning entity and CRM server that are at the **bank server** and, a client who owns the **customer's device**.

As a concrete example, consider the following scenario: Alice is a customer in a bank and wants to use her mobile phone to log in into her online bank. The bank, therefore sends to hers device a credential program and a credential secret, in order for Alice to generate OTP, this set will be called a credential.

This credential contains some parameters such as the expiration date of the credential, the user authentication policy (UAP), a list of applications that can access this credential and the usage for it. These parameters will be managed by the CRM.

One example of management that the CRM server can do is to modify a customer UAP. The UAP for new customers may be that the customers have to write their PIN code every time they need to access an OTP. After some months of using the service, the bank may decide that the customer does not need to identify himself every time he wants to use the service.

Another possible example is the management of the application list, for example if the bank creates a new application for its customers. The customers will want to access that application using their old credentials, therefore the credential's application list has to be updated to include the new application identifier.

4.2 Terminology and assumptions

This section provides the definition for the terminology used in this chapter. In addition, it provides the reason why the parameters were chosen to be part of the CRM scheme.

All of the credentials in ObC platforms can have parameters associated such as, name, owner, expiration date, user authentication policy, application authentication policy, etc. The parameters that a credential owner may be interested in updating and being able to manage are the following: the expiration date, an applications list identifying the applications that can access the credential (applist), the user authentication policy (UAP) and the application authentication key (appAuthKey) and if the credential can be modified. These parameters are related to the ObC credential platform (see chapter 3). In addition, some of these parameters will be maintained and enforced by the Credential Manager in the device and others will be maintained by the TrEE of the device.

- **Expiration date**: This date is the expiration date of the credential, in general all credentials have an expi-

ration date, and the owner of the credential may want to change this parameter after setting it. This parameter is maintained by the Credential Manager because the TrEE does not possess a clock.

- **AppList**: This parameter contains the list of applications that can access that credential. This is a list of unique identifiers for the applications. It is maintained and enforced by the Credential Manager because this feature uses the unique identifiers provided by the underlying OS security platform.
- **UAP**: The user authentication policy, describes how the user authenticates to the device in order to be able to use that credential. For example, the owner of the credential may require that the user inputs a PIN code every time that the credential is going to be used. This parameter is enforced by the Credential Manager.
- **AppAuthKey**: Credentials can be bound during their creation to an AppAuthKey. The appAuthKey depends on the application authorization policy, which is used to determine which applications can use the credential. There are three possible options to access the credential: only the creator, using the appAuthKey or open to all applications. Therefore, if an application wants to access a particular credential that has the AppAuthKey policy, it should share this key. This parameter is enforced by the TrEE, and it controls if this parameter applies.
- **Modify:** The owner of the credential may not want to modify this credential later. When this parameter is set that the credential should be not modified, it is not possible to change this value. This parameter is enforced by the TrEE.

In addition to the different parameters, it is important to define the different entities that are participating in the CRM scheme:

- **Provisioning server**: The provisioning server is a server that provisions the symmetric key. Provisioning in this case means that the server creates and then sends the credential to the device so that the device can use it. (See section 3.2)
- **CRM server**: The CRM server is the server that manages the credentials. It does not matter where the credential is created (either in the device or in the provisioning server), the CRM server will be the owner of the credential and will have the rights to update the credentials in the device.

  The provisioning server and the CRM server usually are the same entity.
- **Relaying server:** The relaying server is a server where the credentials are used, e.g. the bank where the OTP are used.

It is assumed that the CRM server and the device are, both are equipped with a hardware-based TrEE that provides protected execution of trusted code and secure storage. Likewise, it is assumed that the operating system in the device and in the CRM server only allows trusted applications to communicate with the TrEE, however the operating system cannot be trusted completely.

In addition, the provisioning server and the CRM server can be the same server, in the event that a provisioning server is needed. However, the CRM and the provisioning server must always belong to the same party. The CRM can only modify the credentials created by that provisioning server or the credentials that were registered to the CRM server, in other words the CRM server is the owner of those credentials.

4.3 Threat model

The following assumptions are the capabilities of the attacker:

1. **Network communication control:** The attacker is able to read, modify and replay any network traffic between the user's device and the CRM server.
2. **External media control:** The attacker has access to any data stored in insecure storage media.
3. **TrEE control:** The attacker cannot read or modify any processing that takes place within the TrEE, or read or modify any secrets. The attacker is not able to tamper the TrEE with hardware or mount side-channels attacks.
4. **Operating system controls:** The attacker can install or remove any OS application while in physical possession of the device. The attacker might reinstall the OS, or reset it. The attacker can also erase the secure database.

4.4 Security requirements

The following requirements are security requirements that this design must enforce:

1. Only the same entity that provisions the secrets or where the key pair is registered, it is the one that will be able to update the parameters.
2. There are some parameters that must be kept confidential from other applications, these parameters must not leak during the remote management.

5 Design of credential remote management scheme

This section describes the proposed design for CRM scheme. It is divided into three subsections: asymmetric key creation and registration, symmetric key provision and credential update.

It is important to notice, that this protocol as ObC does, does not provide user authentication, this must be done separately.

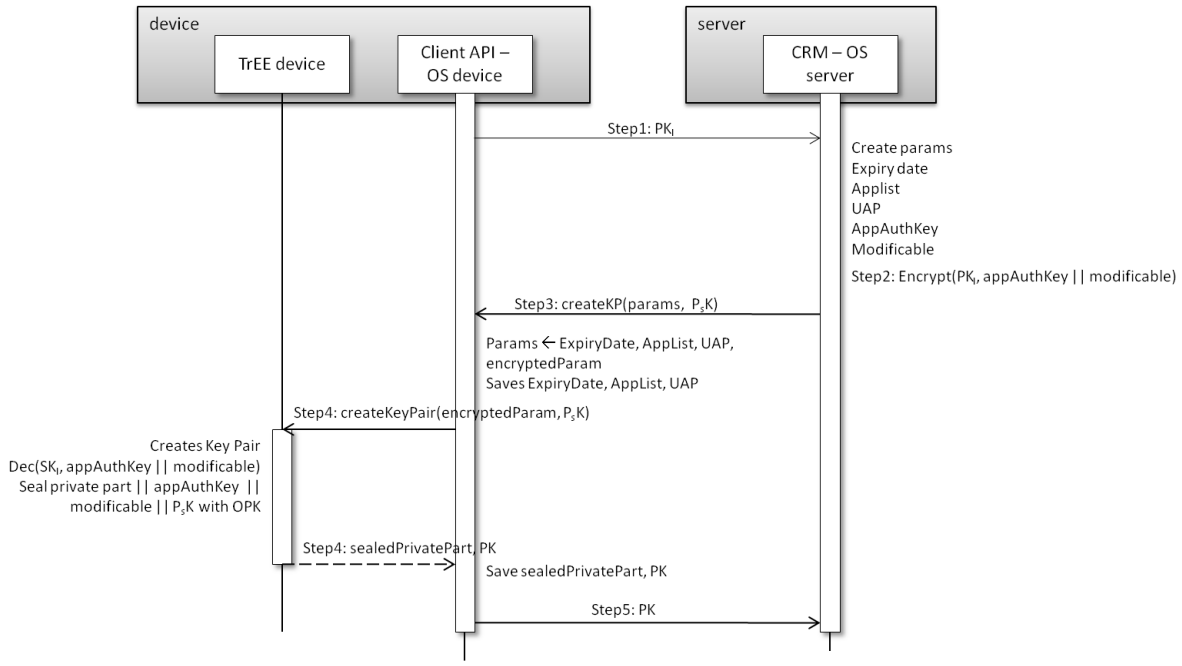

Figure 3: Asymmetric key creation

5.1 Asymmetric key creation and registration

Key creation is the first step in the CRM scheme. Key creation results from the request of a CRM server to a device to create a new asymmetric key, and the registration of that key to the server. This process will also create the parameters associated with this credential.

Figure 3 shows how new asymmetric key is created and how the parameters of that key are assigned.

The steps shown in Figure 3 are described below:

Step 1: The client sends its public device key to the CRM server, this is needed in order to the CRM server can encrypt the parameters enforced by the TrEE of the device. When the CRM server receives the public key of the device, it creates the parameters associated with the credential: expiration date, applist, UAP, appAuthKey, and the parameter if the credential is modifiable. Only the required parameters will be created.

Step 2: The appAuthKey and the modify parameters have to be communicated to the device's TrEE to process them. Therefore, in this step the server encrypts the parameters using the public key of the device.

Step 3: The CRM server sends to the device the parameters, the encrypted parameters and its public key.

Step 4: The Credential Manager sends the encrypted parameters to the TrEE. The TrEE creates the key pair and then decrypts the encrypted parameters using the secret key of the device. Then it seals the private part of the key pair with the decrypted parameters using the OPK, and sends them to the Credential Manager, to be stored in the secure storage.

Step 5: The Credential Manager sends the public key of the just created key to the CRM, in order to prove that the creation process was successfully finished.

5.1.1 Symmetric key creation

The key creation is the first step in the CRM scheme. The key creation results from the request of a CRM server to provisioning a new symmetric key to the device. This process will also create the parameters associated with this credential.

Figure 4 shows how a new symmetric key is created and how the parameters of that key are assigned.

The steps shown in Figure 4 are described below:

Step 1: The client sends its public device key to the CRM server, this is needed in order to the CRM server can encrypt the family key (FK). The FK is shared between the CRM and the TrEE.

Step 2: This step is the same that the ObC provisioning.

Step 3: The CRM creates the parameters for this symmetric key: expiration date, applist, UAP, appAuthKey, and modify. Only the required parameters will be be created.

Step 4: Then the XFER message is created like the normal provisioning, but the AppAuthKey and modify parameters can be added to the seal.

Step 5: The CRM server sends to the device the INIT and XFER messages its public key.

Step 6: The TrEE adds the credential secret. The Credential Manager sends the INIT and XFER message and the public key of the server to the TrEE. The TrEE extracts the secret from the package and all the parameters. Then it seals the secret with the decrypted parameters and the server public key using a derived version of the FK, and sends them to the Credential Manager, to be stored in the secure storage.

Step 7: The Credential Manager creates a key identification for that credential and stores the secret with the other parameters. When it completes the creation process, it sends the key identification to the CRM server, in order to prove that the creation process was successfully finished.

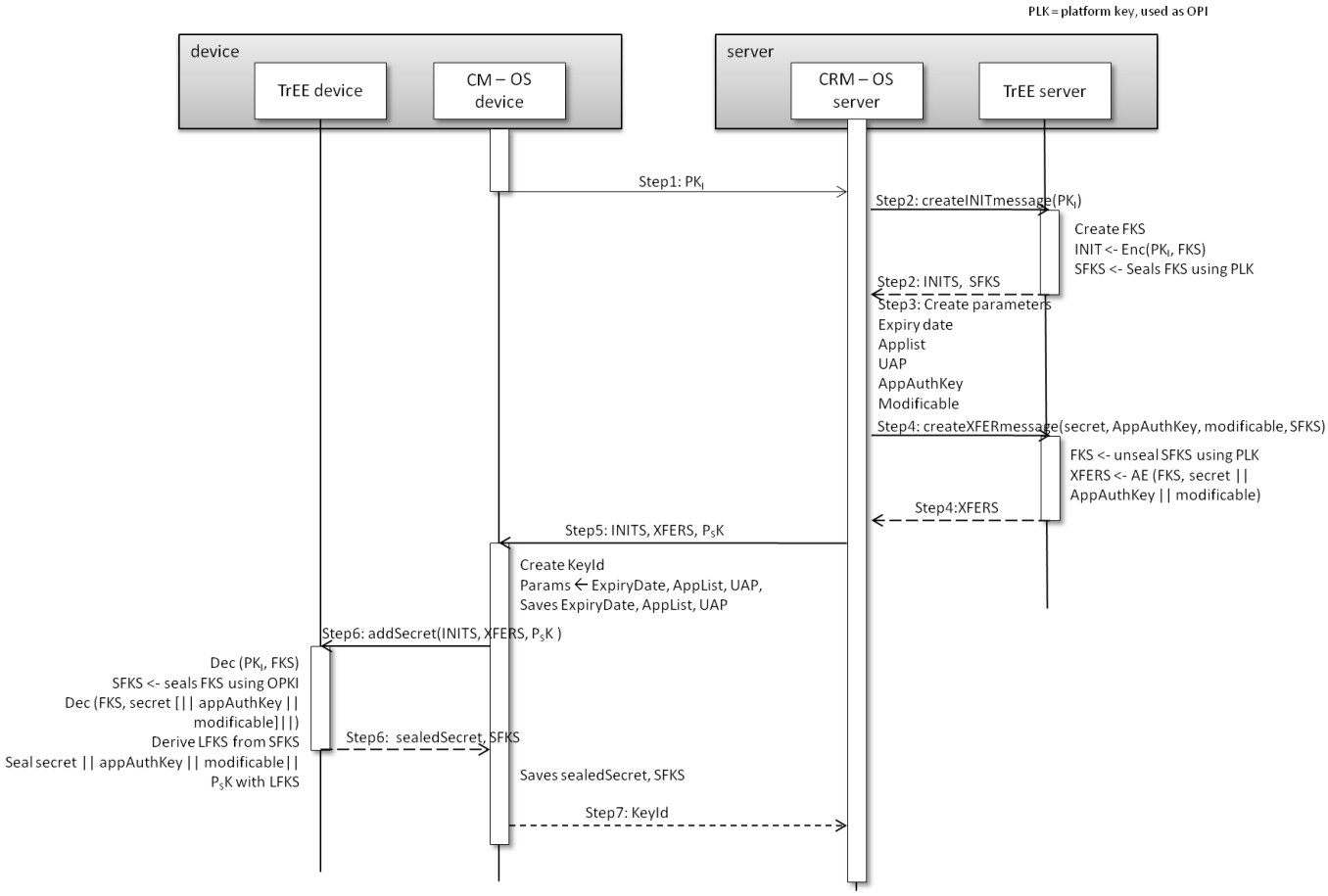

Figure 4: Symmetric key creation

5.1.2 Creating and updating parameters

Using this scheme is also possible to create new parameters and update for the already existing credentials old parameters. For example, a credential can be created without an expiration date, and later the owner of the credential has the option to add that as a new parameter. This operation is called update of the parameters, and Figure 5 illustrates how it is done for asymmetric keys.

To update the parameters in the device:

Step 1: The CRM server looks for the public key that it wants to modify. Then it checks that the credential is modifiable.

Step 2: It the credential can be modified, the CRM modifies the parameters.

Step 3: If the parameter to be updated is appAuthKey or modify, then these parameters have to be encrypted, in order for the TrEE in the customer's device to manipulate them. Therefore this step will encrypt the parameters using the stored public key of the device.

Step 4: The CRM server sends the parameters and the public key of the server to device.

Step 5: The Credential Manager sends the encrypted parameters, and the sealed private key to the TrEE. First the TrEE checks that the credential is modifiable. If so, then the TrEE unseals the private part and updates the corresponding parameters. Finally it seals it again.

Step 6: The Credential Manager sends the public key to the CRM, in order to prove that the creation process was successfully finished.

If the credential is a symmetric key, the update of a parameter is almost the same, except instead of using the private and public part, it will be the FK and the credential secret.

5.2 Security analysis

Based on the assumptions presented in the previous chapter and the threat model, the following section provides an informal security analysis of the proposed CRM scheme.

The design meets the security requirements (see Section 4.4), the first requirement is meet and enforced by signing the parameters with the key of the server. At the moment of creation of these parameters, the public key of the server, responsible for the creation of the parameters, was stored in the device and, at the moment of the update the device will verify that the parameters are signed using the same key.

The second requirement is meet and enforced by encrypting in the server the parameters that must be kept confidential using the public key of the device. In this way, the only entity capable of decrypting them is the device's TrEE.

The design implementation provides a safe way of transmitting all of the messages through the network, even though the attacker can control the network communication, the messages travel over TLS.

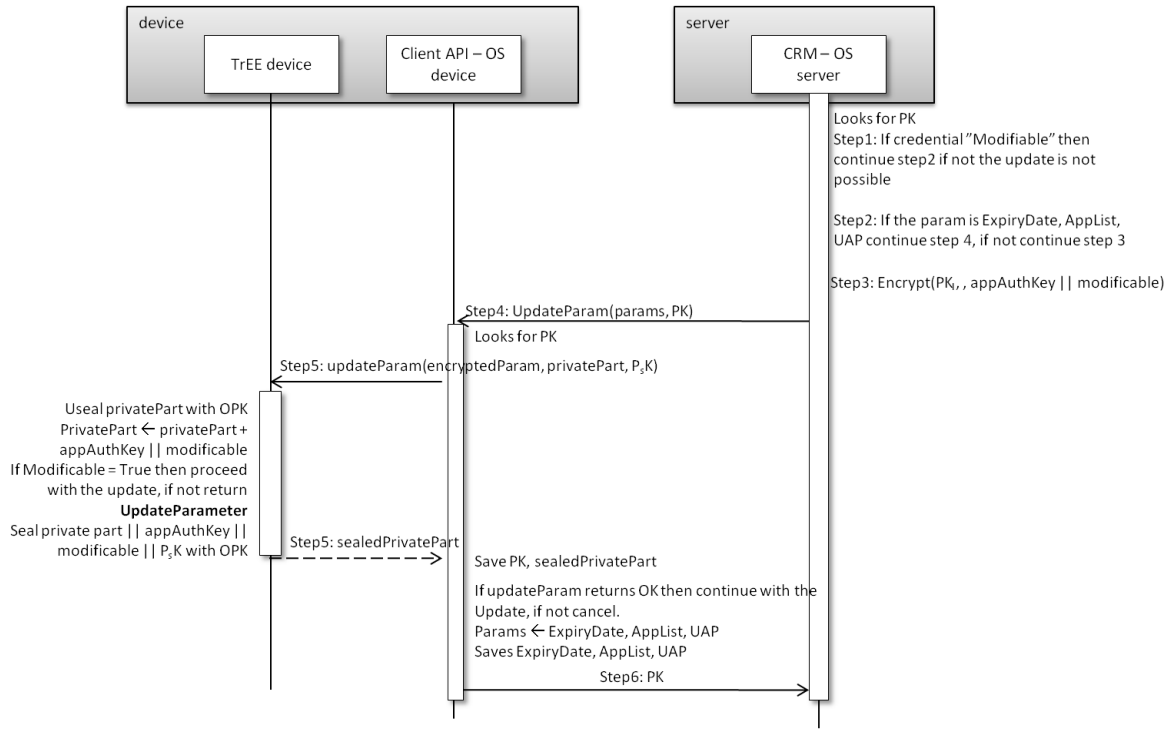

Figure 5: Asymmetric key update of parameters

The parameters that are enforced by the Credential Manager (expiration date, application list, UAP) are stored in the device's un-trusted environment. Even though, these parameters are secured by the OS security features, if the OS has been compromised it is possible that these parameters have also been compromised.

In addition, the parameters enforced by the TrEE are also in the ObC database, however these parameters are encrypted with a key that is only available in the TrEE. These parameters cannot be compromised, because the attacker cannot read the key from the TrEE.

The **expiration date** enforcement is provided by the Credential Manager. The Credential Manager is able to control the expiration date of a given credential using the clock of the OS. The expiration date update can be supported simply by modifying this parameter in the Credential Manager. The expiration date is included in the credential information with other parameters.

The **application list** (applist) is enforced by the OS security platform. This parameter is stored in the OS environment and its update is supported by overwriting this list.

The **user authentication policy** (UAP) enforcement is provided by the Credential Manager. The Credential Manager controls this policy and its update.

The **application authentication key** (appAuthKey) enforcement is provided by the TrEE. This key is only visible inside the trusted environment and it can only be updated by the interpreter. Furthermore, the key creator can always be identified via its public key or using a FK.

The **modify** parameter, is enforced by the TrEE. This parameter is only visible inside the TrEE and it can only be updated by the interpreter. Further more the creator can always be identified via its public key or using a FK, like in the appAuthKey.

5.2.1 OS Attacks

If the attacker is able to attack the operating system he will be able to do certain attacks as rollback attack. Rollback attack means that if the TrEE modifies a value in the ObC database, then the attacker can replace that new value with an older version, and the TrEE will not notice the attack.

Another possible attack is that the attacker modifies the parameters that are enforced by the Credential Manager. If the attacker takes control of the OS, those parameters are not safe any more, and can be modified and read by the attacker, because the security of these parameters depends on the OS security platform.

6 Conclusions

This section includes an overview of the conclusions obtained throughout, the benefits that the CRM scheme will provide.

This paper describes the design of a CRM scheme for creating and updating credential parameters in an automatic and remote manner.

The most important aspects of this paper are:

1. **The remote management of parameters was found to be needed**, because none of the studied credential platforms provided this feature. The purpose of the CRM scheme designed in this thesis is to manage credential parameters.

2. The **parameters** that were identified as important parameters to be managed were the: **expiration date** of

the credential, the **application list** of what has access to that credential, the **user authentication policy** in order to use the credential, and the **application authentication key** of the credential. These parameters are important to the owner of the credential, who might want to access them and update them after the creation of the credential.

3. The benefits of the CRM scheme for the owner of the credentials are that this scheme **allows remote management of its customer's credential**. This scheme can also provide an **automatic update** of the parameters, so if the owner has to update all of his customer's credential, he can do it using this scheme in an automatic way. Additionally this scheme provides credential security, during the transmission over the network and in the storage.

References

[1] Z. Chen. *Java Card Technology for Smart Cards: Architecture and Programmer's Guide*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2000.

[2] V. Costan, L. F. Sarmenta, M. Dijk, and S. Devadas. The Trusted Execution Module: Commodity General-Purpose Trusted Computing. In *CARDIS '08: Proceedings of the 8th IFIP WG 8.8/11.2 international conference on Smart Card Research and Advanced Applications*, pages 133–148, Berlin, Heidelberg, 2008. Springer-Verlag.

[3] S. Gajek, H. Löhr, A.-R. Sadeghi, and M. Winandy. TruWallet: trustworthy and migratable wallet-based web authentication. In *STC '09: Proceedings of the 2009 ACM workshop on Scalable trusted computing*, pages 19–28, New York, NY, USA, 2009. ACM.

[4] Jan-Erik Ekberg AND Markku Kylänpää. Mobile Trusted Module (MTM) an introduction. http://research.nokia.com/files/tr/NRC-TR-2007-015.pdf, 2007. [Online: accessed 23/08/2010].

[5] Jerome Azema AND Gilles Fayad. M-shield Mobile security technology: making wireless secure. http://focus.ti.com/pdfs/wtbu/ti_mshield_whitepaper.pdf, 2008. [Online: accessed 23/08/2010].

[6] Kari Kostianen AND Marcia Villalba. Credential manager API documentation. Available from authors, 2009.

[7] K. Kostiainen, J.-E. Ekberg, N. Asokan, and A. Rantala. On-board credentials with open provisioning. In *ASIACCS '09: Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, pages 104–115, New York, NY, USA, 2009. ACM.

[8] Lua. Lua: the programming language. http://www.lua.org/, 2010. [Online: accessed 23/08/2010].

[9] Marcia Villalba. ObC Example. Nokia Research Center. Available from authors, 2009.

[10] Marcia Villalba. ObC Tutorial. Nokia Research Center. Available from authors, 2009.

[11] Marcia Villalba. One-time password implementation and credential remote managment scheme using Onboard credentials, 2010.

[12] J. M. McCune, B. J. Parno, A. Perrig, M. K. Reiter, and H. Isozaki. Flicker: an execution infrastructure for tcb minimization. In *Eurosys '08: Proceedings of the 3rd ACM SIGOPS/EuroSys European Conference on Computer Systems 2008*, pages 315–328, New York, NY, USA, 2008. ACM.

[13] Rundgren, Anders. SKS (secure key storage) - API and architecture. http://webpki.org/papers/keygen2/sks-api-arch.pdf, 2010. [Online: accessed 26/08/2010].

[14] A.-R. Sadeghi, C. Stüble, and M. Winandy. Property-Based TPM Virtualization. In T.-C. Wu, C.-L. Lei, V. Rijmen, and D.-T. Lee, editors, *Information Security*, volume 5222 of *Lecture Notes in Computer Science*, pages 1–16. Springer Berlin / Heidelberg, 2008. 10.1007/978-3-540-85886-7.

[15] Trusted Computing Group. Trusted computing. http://www.trustedcomputinggroup.org/trusted_computing, 2010. [Online: accessed 04/01/2010].

[16] J. Winter. Trusted computing building blocks for embedded linux-based ARM trustzone platforms. In *STC '08: Proceedings of the 3rd ACM workshop on Scalable trusted computing*, pages 21–30, New York, NY, USA, 2008. ACM.