Mobile Phone Work Group

Selected Use Case Analyses – v 1.0

Abstract

This document describes a set of selected mobile phone uses cases and identifies required security mechanisms in a mobile phone implementation that support these use cases. These security mechanisms are mapped onto existing MTM commands. We also suggest potential specification enhancements.

This document constitutes an annex to Use Case Scenarios v2.7, published in September 2005.

Contents

1 ABOUT THIS DOCUMENT

1.1 Introduction

The TCG mobile phone specifications (Mobile Trusted Module (MTM) specifications) enable trusted computing on mobile platforms. The specifications have been developed with the particular needs and limitations associated with mobile devices in mind. The mobile specification work started with requirements and use case collections. The following mobile phone use case scenarios were identified [1]:

- Platform Integrity
- Device Authentication
- Robust DRM Implementation
- SIMLock / Device Personalization
- Secure Software Download
- Secure Channel between Device and UICC
- Mobile Ticketing
- Mobile Payment
- Software Use
- Prove Platform and/or Application Integrity to End User
- User Data Protection and Privacy

This collection of use cases includes both mobile phone-specific use cases and general relevant TCG use cases. The purpose of this document is to analyze the mobile use cases in order to show how the various mobile phone stakeholders (e.g. users, manufacturers, operators, service providers) benefit from a TCG-compliant mobile platform. To that end, the use cases published in September 2005 have been reviewed and updated where pertinent. Currently this document treats only a subset of the use cases listed above. The selection has been made with prioritization on those use cases with security requirements that we think are typical for a mobile phone.

Platform integrity is a basic requirement for many use cases. The mobile phone working group has put major efforts in defining secure boot mechanisms. Secure boot provides an engine's stakeholder with confidence that certain services were correctly instantiated. Secure boot sequences are provided as part of the mobile trusted module specification. In particular we discuss the mobile phone personalization, software flashing and software version handling aspects of the platform integrity use case.

Device and subscriber authentication is a core feature of cellular systems. A mobile phone stores and uses several different identities and associated secrets. Trying to analyze mobile phone device authentication taking all different device authentication scenarios into account is a very large task. Instead we have chosen one device/subscriber authentication scenario and see how TCG mechanisms can be used to enhance the security for this case. One of the most widely deployed cellular network standards, are the 3GPP standards and we analyze the UMTS device and subscriber authentication mechanisms.

The goal of DRM technology is to create a trusted infrastructure for distribution and consumption of digital content such as music, film, etc. In particular the content owners want to protect their content against non-authorized use and copying. This protection should be maintained even when the content has been delivered to the end-user's device where the content is to be transformed from its digital format into user consumable form. There are many aspects around protection in and architectures for DRM solutions. In the context of this document we consider the OMA DRM 2.0 architecture, [9], but the general reasoning applies with necessary modifications for other DRM solutions as well.

SIMLock is about ensuring that a mobile device remains locked to a particular network (or network subset, service provider, corporation or (U)SIM) until it is unlocked in an authorized manner [5]. SIMLock fraud costs mobile network operators (MNOs) a large amount of money yearly, and it is one of the major mobile phone security design challenges. We illustrate how mobile phone SIMLock security can be enhanced using the TCG technology.

Current mobile phones have an open unsecured interface between the mobile terminal and the UMTS subscriber identity module on the UICC. 3GPP has released a specification that allows secure establishment between a mobile terminal and an UICC [6]. This enables such things as a secure PIN entry path, secures message display on a terminal screen, the UICC as a control point for device management and increased usage of the UICC for data protection. We analyze how TCG mechanisms can be used to increase the security of the terminal end of the UICC to terminal secure channel.

Mobile ticketing is an electronic ticketing service where the proof of access or usage right to a particular service is controlled by a mobile phone. One can distinguish between *two* major different mobile ticket systems [12]:

- One where the actual ticket just is a record in the issuer server and redemption of the ticket consists of user authentication from the mobile phone to the server.
- The ticket is an electronic object *stored* protected in the mobile phone.

The first type of mobile ticket scenario is actually a user authentication use case and we will *not* analyze that scenario but instead the second one, i.e., the electronic ticket object scenario.

Many different mobile payment systems have been deployed and much effort has been put into the development of mobile payment schemes during the last ten years. However, the expected large boom for mobile phone based payments has not come. This is due to several different reasons and there exist many analyses of models, market, fragmentation among stakeholder, and trends. Mobile transactions can either be remote such as payment of images, ringing tones, goods etc. or local such as parking, toll, vending machine, fast food or retail payments. With the introduction of NFC, local payment schemes might finally find the way into mobile phones or perhaps NFC in credit cards will be the dominant solution. As NFC based payment is evolving, we have chosen to analyze several NFC based payment schemes in relation to the TCG mobile phone working group new specifications.

1.2 References

This document refers to the following documents.

Editors Note: These are not properly ordered yet.

Reference	Description and Location
[1]	Trusted Computing Group, *Mobile Phone Work Group Use Case Scenarios,* Specification Version 2.7, 2005.
[2]	Trusted Computing Group, *TCG Mobile Reference Architecture,* Specification Version 1.0, 2007.
[3]	Trusted Computing Group, *TCG Mobile Trusted Module,* Specification Version 1.0, 2007.
[4]	V. Niemi and K. Nyberg, *UMTS Security,* Wiley2003
[5]	3GPP, *Personalisation of Mobile Equipment (ME); Mobile functionality specification,* TS 22.022 version 6.0.0
[6]	3GPP, *Key establishment between a UICC and a terminal,* TS 33.110 version 7.0.0
[7]	Open Mobile Alliance, http://www.openmobilealliance.org.
[8]	Interoperable Digital Rights Management Platform, technical specification v3.0, Digital Media Project website, http://www.dmpf.org
[9]	OMA DRM Architecture, Version 2.0, March 2006, http://www.openmobilealliance.org/release_program/drm_v2_0.html.
[10]	OMA DRM Specification, Version 2.0, March 2006, http://www.openmobilealliance.org/release_program/drm_v2_0.html
[11]	MPWG Use Case Scenarios – Phase 2, v2.4, July 2005, https://www.trustedcomputinggroup.org/apps/org/workgroup/mobilewg/documents.php?&sort_field= d1.submission_date&sort_type=DESC#folder_119.
[12]	MeT White Paper on Mobile Ticketing, MeT 2003.
[13]	MOBEY forum: http://www.mobeyforum.org/
[14]	Mobile device security element http://www.mobeyforum.org/public/pressreleases/ Mobey%20Forum%20Security%20Element%20Analysis%20Summary%202005.pdf
[15]	TPM Main Part 1 Design Principles, Specification Version 1.2, Revision 103, 9 July, 2007
[16]	TPM v1.2 Specification Changes, October 2003

1.3 Terms and Abbreviations

This document uses the following terms and abbreviations.

Term	Description
AIK	Attestation Identity Key
CA	Certification Authority
CEK	Content Encryption Key
CI	Content Issuer
CMLA	Content Management License Administrator http://www.cm-la.com/
CRTM	Core Root of Trust for Measuring
CRTV	Core Root of Trust for Verification
DAA	Direct Anonymous Attestation
DCF	DRM Content Format
DMP	Digital Media Project
DoS	Denial of Service
DRM	Digital Rights Management
EMV	Europay Mastercard Visa
ETSI SPC	European Telecommunications Standards Institute – Smart Card Platform
EU	European Union
GSM	Global System Mobile
HW	Hardware
IC card	Integrated Circuit card or smart card
IMEI	International Mobile Equipment Identity
IMSI	International Mobile Subscriber Identity
ME	Mobile Equipment
MeT	Mobile Electronic Transactions organization. See http://www.mobiletransaction.org.
MLTM	Mobile Local-Owner Trusted Module
MNO	Mobile Network Operator
MRTM	Mobile Remote-Owner Trusted Module
MTM	Mobile Trusted Module
MPWG	Mobile Phone Workgroup
NFC	Near Field Communication
OEM	Original Equipment Manufacturer
OMA	Open Mobile Alliance http://www.openmobilealliance.org
OS	Operating System

OSGi	Open Services Gateway Initiative http://www.osgi.org
OTA	Over the Air
PCB	Printed Circuit Board
PDA	Personal Digital Assistant
PIN	Personal Identification Number
PKI	Public Key Infrastructure
POS	Point of Sale terminal
PRMVA	Primary Run-time Measurement Verification Agent
REK	Right Encryption Key
Repurpose	Modifying a Device to perform differently than intended (e.g. Linux on Game Console).
RI	Rights Issuer
RIM	Reference Integrity Metric
RO	Rights Object
ROM	Read Only Memory
RTM	Root of Trust for Measuring
RTR	Root of Trust for Reporting
RTS	Root of Trust for Storage
RTV	Root of Trust for Verification
RTVI	Root of Trust for Verification Identifier
RVA	Root Verification Authority
RVAI	Root Verification Authority Identifier public key configured in the CRTV for the platform root security authority.
SIM	Subscriber Identification Module
SoC	System on Chip: primary platform host execution engine
SRMVA	Secondary Run-time Measurement Verification Agent
SW	Software
TCG	Trusted Computing Group http://www.trustedcomputinggroup.org
TIM	Target Integrity Metric
TPM	Trusted Platform Module
TSS	TCG Software Stack
TTP	Trusted Third Party
UICC	UMTS Integrated Circuit Card
UMTS	Universal Mobile Telecommunications System
USIM	Universal SIM
3GPP	3rd Generation Partnership Project http://www.3gpp.org

1.4 Actors

In the document the following terms are used to describe certain actors in the use cases.

Actor	Description
Content Provider	The legal owner of content that requires protection using DRM.
Corporation	An enterprise that may support mobile devices as a means to access corporate data and networks.
Device	An entity comprising a platform with one or more Mobile TPMs for which attestation data may be provided
Device Owner	The legal owner of the Device. The Device Owner is entitled to customize the platform. The owner may be a consumer, an IT Administrator for a Corporation or some other entity.
Device Manufacturer	The manufacturer or brand of a Device – typically an OEM
End User	The user of a Device. The End User may or may not be the Device Owner.
Employee	An employee using a mobile phone to access applications, data and networks of the Corporation employing him or her.
Service Provider	An entity that wishes to discover properties of a trusted mobile phone platform and provide services to that Device.
Network Provider	An entity that provides cellular communications functionality to the platform (e.g., a 3GPP network provider.)
Application Provider	An entity generating and/or selling user applications to be executed on the platform
Attacker	A person or organization trying to circumvent some policy of the Device, the Service Provider, the Application Provider or the Network Provider

2 USE CASE ANALYSIS: DETAILED EXAMPLES

2.1 Introduction: Platform Integrity

2.1.1 A basis for use cases

Platform integrity is an essential ingredient for most, if not all, use cases that will be described in the sequel of this document. The concept of platform integrity goes back to the root of TCG itself and it addresses those requirements and usage scenarios with which the stakeholders want to achieve a trusted environment for applications. Thus the use case description that goes along with platform integrity is essentially the one behind the MTM architecture and specification itself. The notion of platform integrity has two important aspects. One is that of being a platform that can be relied upon to provide the promised services and the other aspect is that of being able to keep delivering the services when being attacked. Furthermore, depending on the use case one can distinguish two very different perspectives for platform integrity. The most obvious perspective is that of the user of the platform wanting the previous two mentioned aspects of integrity to be realised. Or, in other words, the user and the platform share the same interests and the user has no interest to attack the platform. This is most often the case but there are other situations where the platform and user have different interests. Such situations are usually more challenging when it comes to realize the intended services and often more than one stakeholder's interest need to be addressed. In this document we see such a situation when describing the DRM use case. Another example of such a situation is that of surveillance equipment or sensors operating in a, in principle, hostile, environment. As we will see the MTM allows us to realize technical solutions that can achieve many different practical needs for protection and trusted operation of a device. The MTM specification [3] and Mobile Reference Architecture [2] describe how the protection and trusted operation can be realized and how the MTM can and/or should be utilised.

2.1.2 Threats

It is obvious that numerous threats come from shortcomings in the implementation of the MTM. Proper design procedures backed up with assurance providing procedures and processes like those given by, for example, Common Criteria and related protection profiles should be utilized to minimize the risks. Also proper testing in form of conformance and penetration tests are good instruments in the process to counter the threats. Other threats come from low-level ASIC debug and/or hardware analysis means. Such means are often a “must have” in devices to pin-down the cause of malfunctioning devices when dealing with field returns. Careful design should take into consideration the threats that come from debug and analysis means and one should deploy appropriate counter measures.

Furthermore, threats could also come from the user of the device. Especially when different stakeholders have interest in the integrity of the platform there can be conflicting interests and one may land up in a situation that from a given stakeholder the device could be physically operating in the territory of the enemy. It is obvious that physical protection is then a must to stand against attacks.

2.2 SIMLock / Device Personalisation

2.2.1 Use case description

The 3GPP SIM personalization or SIMLock functionality is specified in [5]. The personalization features work by storing information in the Mobile Equipment (ME), which limits the SIMs with which the ME will operate, and by checking this information against the SIM[1] whenever the ME is powered up or a SIM is inserted. If a check fails, the ME enters the "limited service state" in which only emergency calls can be attempted.

There are five personalization categories:

- Network category, controlled by the Network Control Key (NCK)
- Network subset category, controlled by the Network Subset Control Key (NSCK)
- Service Provider (SP) category, controlled by the Service Provider Control Key (SPCK)
- Corporate category, controlled by the Corporate Control Key (CCK)
- SIM/USIM, controlled by the Personalization Control Key (PCK).

The personalization categories are independent in so far as each category can be activated or de-activated regardless of the status of the others. The ME can be personalized to one network, one network subset, one SP, one corporate entity, one SIM/USIM or any combination thereof.

The core feature of the SIMLock functionality is to compare the different possible locks with the subscriber identity field, the IMSI, stored on the SIM. A SIMLock protection scheme consists of *three* major parts, protection of the SIMLock control keys, protection of the SIMLock data and protection of the SIMLock and network software execution environment.

The SIMLock control keys are used to lock and unlock the SIMLock personalization. They need to be stored in the ME in non-volatile memory such that they can be used by the SIMLock controlling software. The SIMLock codes do not need to be stored in *clear text* but can be encrypted or one can choose to store one or several *one-way* hash values of the SIMLock codes. Independent of which option that is chosen, either the SIM lock codes or hashes should be stored in shielded storage or integrity protected in ordinary non-volatile memory such as a flash memory (see also Section 2.2.2 below).

The SIMLock data describes the individual personalization of the phone and it must be possible to *dynamically* reconfigure the lock data. This implies that also the SIMLock data must either be stored in *shielded* storage or integrity protected storage.

The SIMLock software that performs the SIMLock data checks, must run in a protected environment. Similarly, the network software that handles the communication with the network and the SIM card must be protected from manipulations.

[1] For simplicity we use here SIM to cover both the USIM and the SIM application.

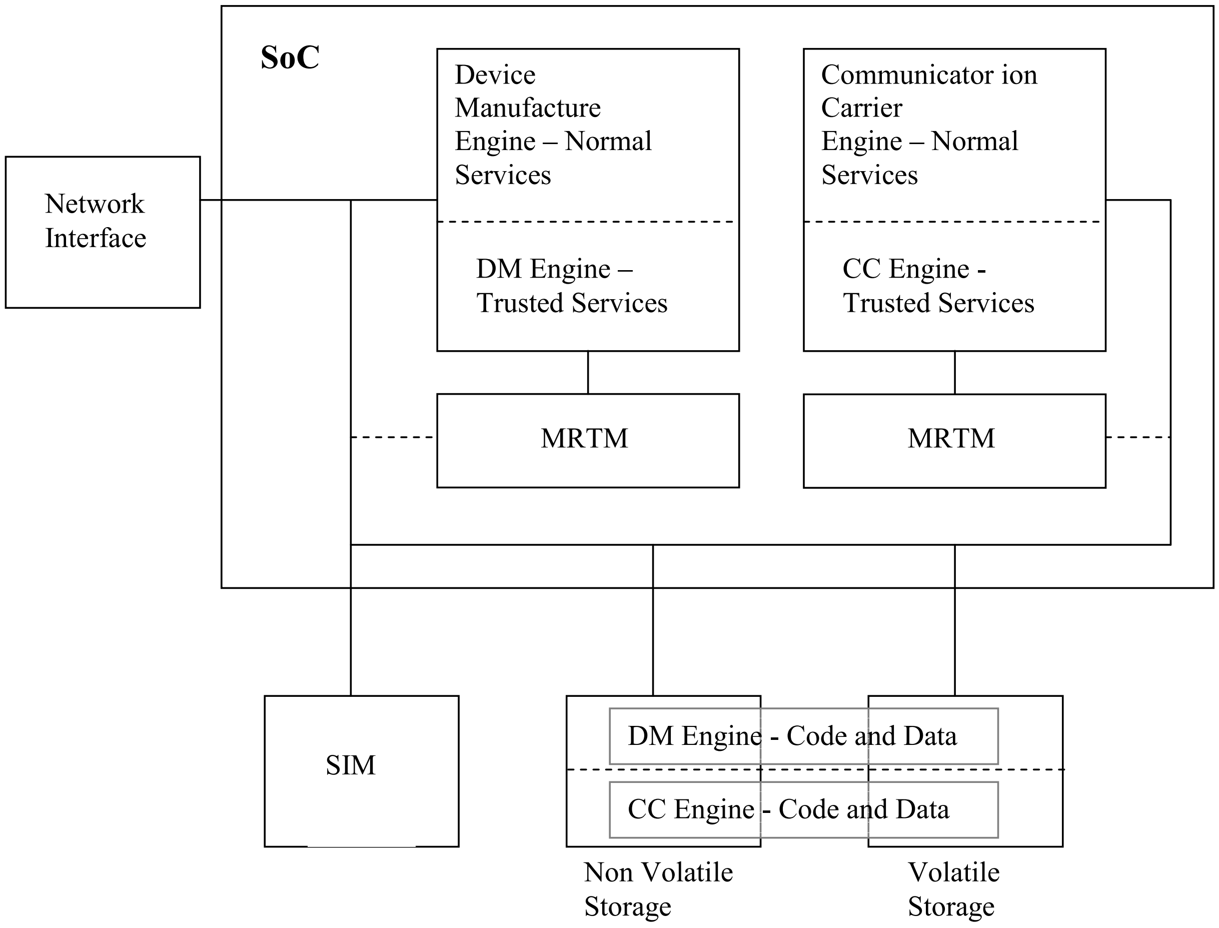

SoC
Device Manufacture Engine – Normal Services
DM Engine – Trusted Services
Communicator ion Carrier Engine – Normal Services
CC Engine - Trusted Services
Network Interface
MRTM
MRTM
SIM
DM Engine - Code and Data
CC Engine - Code and Data
Non Volatile Storage
Volatile Storage

Figure 1: A Mobile System on a Chip (SoC) TCG mobile platform imitation example (MRTM built from allocated resources).

Figure 1 illustrates an example SoC according to the TCG Mobile reference architecture [2]. The SIMLock control key data and SIMLock data might be stored in the *external* non-volatile storage medium. The SIMLock software typically will run as a trusted service in the device manufacturer engine and the network software will run as a service in the device manufacture engine. Using this example architecture we discuss the threats against a SIMLock realisation and the usage of the MTM commands.

2.2.2 Threats

When we analyse threats against SIMLock implementations it is not enough to just consider software attacks. As it is the phone user him/herself that is the potential attacker, we must assume that the attacker has physical control of the device. Consequently, it is as important to analyse different hardware attacks. In this document we will not give a complete security analysis on different SIMLock threats but concentrate on a couple of typical threat scenarios. In particular, we will discuss the following attacks:

1. Attacks on the SIM
2. Attacks on external non-volatile storage
3. Attacks on external volatile storage
4. Attacks on the SIMLock software
5. Attacks on the Network software
6. SoC side channel attacks

2.2.2.1 UICC

Successful hardware attacks on SIMs are very uncommon as the card manufacturers have spent many efforts on making the UICC that hosts the SIM tamper resistant. If it would be possible for an attacker to get hold of the SIM secret values etc., the major threat is a *clone* of the SIM. A successful clone would allow the attacker to use the SIM credentials in another SIM or phone. However, this we do not consider a SIMLock threat but rather a general security threat against SIM. The goal of a SIMLock attacker is instead to get *another* SIM to be accepted by the *phone.* Hence, the SIM itself is not the major attack target.

Another threat on the SIM is the SoC to UICC *interface.* The goal of the attacker is to replace a “legal” SIM (according to the SIMLock rules) with a non-legal one. If the attacker is in control of the UICC interface, he/she might be able to add a *second physical* UICC slot to the device where he/she places the non-legal SIM. If the attacker at the same time has the possibility to switch between the two slots the SIMLock protection might be broken. This is especially true if the SIMLock checks *only* take part at start-up or SIM removal as mandated by the 3GPP standard [5]. We will discuss different mobile trusted platform countermeasures in Section 2.1. A secure UICC channel as discussed in Section 2.3 would considerable reduce this threat.

2.2.2.2 External Non-volatile storage

External non-volatile storage can be attacked in several different ways. One way is to physically *remove* or manipulate the non-volatile storage medium. Depending on the system design this might or might not be hard to do in practice without destroying the device. Another option is to attack the *interface* to the storage medium. Again, depending on the system design, this might be easy or very hard. The third option is to install hacker software on the phone that reads and/or manipulate data stored on the non-volatile storage medium. A good SIMLock implementation must take any of these threats into account. Hence, the design should be resistant to any attack against SIMLock data stored on non-volatile storage.

SIMLock or device personalization requires protected storage of several different parameters:

- SIMLock control keys
- SIMLock unlock attempt counter
- SIMLock data

The SIMLock control keys are used by the SIMLock software to validate a valid SIM unlocking operation. Obviously, if the control keys are stored *in clear* in external non-volatile memory both the non-volatile memory and the interface to the non-volatile memory are very sensitive to attacks. There are two major options of how the control keys can be protected. One option is to store the control keys *encrypted* and *integrity protected* on *non-volatile* storage. Another option is to *not* store the control keys as such but instead just store integrity protected secure *one-way hash values* of the SIMLock keys in non-volatile memory. A third option is not to store the keys in external non-volatile memory at all but instead in protected storage on SoC level for example. If the control keys are stored protected on external non-volatile memory, an attack on the non-volatile storage medium or the interface to the non-volatile storage will only succeed if the encryption and/or integrity protection algorithm can be broken.

The 3GPP SIMLock and device personalization standard [5] recommends that the device only shall accept a limited number of failed attempts to de-personalize the device. Hence, there is a need to store, properly protected, a (failed-unlock) attempt counter in non-volatile storage. If the attacker can reset this counter or replace it, he/she will be able to run an unlimited amount of unlock attempts. Hence, also this counter must be stored integrity protected on flash. If the SIMLock unlock attempt counter is stored integrity protected, an attack on the non-volatile storage medium or the interface to the non-volatile storage will only succeed if the encryption and/or integrity protection algorithm can be broken

The SIMLock data determines the phone SIMLock status and contains personalization rules. This is an obvious attacker target. Similar to the storage of an unlock attempt counter, integrity protection of the SIMLock data prevents unauthorized modification of the SIMLock data.

A mobile device might contain an execute in-place external non-volatile storage medium. This is another potential attack threat but this threat is eliminated if no SIMLock or device engine security critical code is executed direct from the non-volatile storage.

2.2.2.3 External volatile storage

Similar to the external non-volatile storage, the external volatile storage can be attacked using both hardware and software based attacks. The attacker has several different possibilities:

- Read SIMLock secret data transferred to and from the volatile storage medium
- Read SIMLock secret data direct from the volatile storage medium
- Modify execution code transferred to and from the volatile storage medium

As we discussed in Section 2.2.2.2, integrity protection and optional encryption of critical SIMLock data parameters is required if the SoC does not contain protected storage for these parameters. This implies that one or several symmetric secret key must be available on the system. If these keys are temporarily stored on external volatile memory, they are sensitive to physical attacks on the SoC to volatile memory interface. Similarly, if the SIMLock unlocking keys or hash values of unlocking keys, attempt counters or SIMLock data are temporarily stored on volatile memory; these are also subject to an attack on the volatile memory interface. An attack on the SIMLock data transferred over the volatile memory interface is only of limited use to the attacker though as that only will give a temporary de-personalization of the device.

An even more powerful attack would be to get direct access (physically or through software) to the volatile storage medium. The threat picture is the very same as for the interface attacks regarding SIMLock secret and data attacks. Software attacks on the volatile temporary storage might be possible if the attacker can circumvent the protection mechanisms of the device manufacture engine such that the security critical parameters can be retrieved at run time. Another threat that does not require a break of the device manufacturer engine security mechanisms is the following: the attacks succeeds with halting the execution at a certain step and then restart the device into a non-trusted state while keeping the external volatile memory content intact and then read out the secret parameters.

Another attack target is software executed from external RAM. In a mobile trusted platform SIMLock controlling software and/or network software might be executed on external RAM. If this is the case, it might be possible for the attacker to modify/replace execution such that for example non authorized de-personalization can be performed.

2.2.2.4 SIMLock software

A SIMLock can be broken by taking control of modifying the SIMLock controlling software. This can be done in several different ways:

- Manipulation of the stored software or attacking the interface to the storage medium
- Software attacks targeting the SIMLock execution environment

The threats against the SIMLock code at external non-volatile storage are rather similar to the SIMLock data threats and the same type of counter measureless are needed. Threats on SIMLock software executed on external RAM were briefly discussed in Section 2.2.2.3.

The most severe threat against the SIMLock software is software attacks against the SIMLock execution environment. A trusted boot process will hopefully prevent attacks against the SIMLock software at start up of the device, but that will not prevent any attack on the execution environment at a later stage. An attack might for example disable all calls to the SIMLock controlling software or replace them with alternative routines. Such an attack would disable any SIMLock checks during run time.

A mobile trusted platform with a secure boot and with SIMLock software as a trusted service considerably reduces the SIMLock execution environment threats, from both storage attacks and execution environment attacks. However, the protection level will to a large extent depend on the security strength of the device manufacture engine execution environment and that is outside the scope of the TCG specifications. The secure boot as such does not protect against the pristine boot attack with wiped SIMLock data, hence additional security measures are needed.

2.2.2.5 Network software

It is not enough protecting the SIMLock controlling software. The main SIMLock mechanism is to make a mapping between SIM personalization rules and the IMSI number in the SIM. However, if the network software that use the IMSI over the network interface accepts or use a different IMSI, the SIM locking mechanism is circumvented. One example is an attack that introduces a *second* SIM to the system (using a suitable available phone external interface) and modifies or takes control of the network software such that the second SIM is used for network connections instead of the primary one that is controlled by the SIMLock controlling software.

Similar to SIMLock software attacks, network software attacks can target the software storage or the attacker can take control of the network software execution environment. In order to protect the network software one option is to place it as a trusted service in the device manufacturer engine. This might give a good protection of the network software execution environment. However, the network software in a mobile phone typically contains a large code base with many dependencies to different other software components in the system. Hence, from a general security perspective one should try to avoid to large trusted components as that can jeopardize the whole engine security level.

2.2.2.6 SoC side channel attacks

Another attack target is the SoC itself. This includes side channel attacks such as high or low temperature exposure, radiation exposure, grounding of pins, short time pulses to supply voltage, power analysis, probing of signals etc. Such attacks can be used trying to modify the security behaviour of the SIMLock or network software or trying to get hold of secret SIMLock keys or parameters.

2.2.3 Mapping to MTM concepts

We assume a setup where the MTM is implemented in software and runs as a special trusted process in the engine. The MTMs have **no** internal non-volatile memory. There is internal volatile memory to store data structures and calculation results which no other processes have access to.

Each MTM has a chip-unique symmetric key (at least 128 bits long that is stored using e.g. e-fuses) which is configured at chip manufacture. Firstly, we need to be able to store and retrieve the MTM_PERMANENT_DATA and the TPM_PERMANENT_DATA information to and from the external non-volatile memory. We can use a combined encryption and verification algorithm .e.g. AES in Galois Counter Mode. These permanent data structures can then be read by the MTM at power up and stored internally in volatile memory for quick access. At the customisation step in the production, the permanent data structures are calculated externally (in a production computer) and encrypted and integrity protected using the chip unique keys and stored in the non-volatile memory at a pre-defined address.

The following data (at least) must be initialized during customisation:

1. MTM_PERMANENT_DATA->aik (public part, DM specific, same in both engines)
2. MTM_PERMANENT_DATA->internalVerifiationKey (chip unique and engine unique)
3. TPM_PERMANENT_DATA->srk (chip unique and engine unique)
4. TPM_PERMANENT_DATA->tpmProof (chip unique and engine unique)

Additionally, the implementation of the MTM must ensure that MTM_STANY_FLAGS->loadVerificationRootKeyEnable always is evaluated to FALSE.

The SoC ROM area is the Root of Trust which is responsible for loading and measuring the operating environment of the engines. Since the MTMs are implemented in software, we cannot directly at boot time rely on the secure boot mechanisms of the MTM but need something external to measure and verify the MTM software image. After the MTM is loaded and executed, the MTM should take responsibility for the rest of the boot process verification and especially the verification of the operating environment.

After normal power up, we assume that we have a set of PCRs which characterize the trusted configuration and have been verified using RIM_Certs during the boot.

2.2.3.1 Protection of SIMLock software

After the secure boot process, the correct SIMLock software is running in the engine. The first thing the SIMLock software should do is to start an OIAP-session with the MTM process. If all the keys in the MTM used by the SIMLock software are bound (through usageAuth) to the SIMLock software, we can minimize the risk of an attack to the executing environment where hostile code is infecting a different process and starts to ask the MTM to perform duties normally restricted to the SIMLock software.

2.2.3.2 Protection of Network software

Also for the protection of network software, we suggest a secure boot process to ensure the sanity of the network software One could indeed imagine an additional security feature where the network software somehow signs the IMSI code with which it is about to setup a call and then asks the SIMLock software to verify that it is the same IMSI code as the one the SIMLock software has approved. If the IMSI code is the same, the call may proceed.

2.2.3.3 Protection of SIMLock control keys

We consider a solution, where the actual lock / unlock keys are not stored on the platform at all, instead hash digests of the keys are stored. The hash digests are sealed to the PCR values for the trusted boot. In this way the hash digests are only available to the software if the platform has booted to a trusted state. Even if a hostile program has managed to inject itself into the boot, it will not be able to export the hash digests for an offline attack since the PCR values will not match.

When a legitimate service centre is going to unlock a ME, the unlock keys are entered using e.g. the keypad or sent to the ME via SMS. This SIMLock software should then ask the MTM to unseal the hash values of the keys and compare against the hash of the entered values. The unseal will only be successful if the PCR values are the expected and the engine is in a trusted state. Integrity protection of the hash values are provided by the TPM_Seal / TPM_Unseal commands by including a SHA-1 MIC of the data which is encrypted together with the data.

2.2.3.4 Protection of SIMLock attempt counters and SIMLock data

The SIMLock data consists of the IMSI patterns for the five different locks and flags for indicating which locks are active. The SIMLock data and the attempt counters are sealed and stored in non-volatile memory. Whenever a new SIM card is inserted into the phone, the SIMLock software must unseal the lock flags and IMSI patterns and check them to ensure that the SIM is allowed to operate on the ME.

The SIMLock attempt counters are similarly sealed and stored in the non-volatile memory. When the SIMLock software is about to check the SIMLock control keys, the counters are unsealed and verified. Then the SIMLock software must increment the counter, re-seal and write back to flash before any attempt is made to compare the entered keys to the hash values. In case of a successful unlock, the SIMLock software must clear the flag for that lock and may also reset the counter value. Similarly to the control key hashes, the SIMLock data and attempt counters are integrity protected by the TPM_Seal / TPM_Unseal commands.

2.2.3.5 What is achieved?

The MTM provides us with a secure boot, once the MTM software is running in the processor. The sealing of the SIMLock data, counters and control key hashes ensure that only if we are in a trusted configuration we can access the SIMLock data.

Furthermore, the key used for sealing does not need to be available to any other software running on the processor apart from the MTM. The SIMLock software needs to know the key.usageAuth to be able to execute to TPM_Seal and TPM_Unseal commands, but the key itself never leaves the MTM.

UICC (SIM) Card

If the network software can securely ask the SIMLock software before each call if the SIM that is used to setup the call is the same as the one verified by the SIMLock software we might be able to protect against the SIM interface attack. This is however very much dependent on the system design in large.

External Non-volatile storage

The permanent data holding the MTM keys is protected with a symmetric key which is chip unique. This prevents the physical removal and exchange of the non-volatile memory since it is tightly bounded to the chip. An attacker which is able to inject hostile software during runtime cannot access or modify the data stored in the non-volatile memory. However, if an attacker manages to insert code very early in the boot process, the code might be able to access the chip- unique symmetric key. In the use case example realization we rely on the SoC ROM code to prevent this attack.

External volatile storage

The current use case assumes that the MTM uses internal volatile storage both for active data and for executing software code. However, the external volatile storage may store encrypted MTM software code that will be verified and decrypted within the internal volatile storage. Furthermore, if either data or software code is swapped out from internal volatile storage to external volatile storage, it will be encrypted and have a (MTM specific) check value associated with it when writing to external volatile storage; when the previously swapped-out data or software code is swapped back in it will be decrypted and the associated check value verified when writing to internal volatile storage. If the mobile phone moves to sleep mode either data or software code in internal volatile storage may be swapped out to powered memory and swapped back in from powered memory using the method described above. This may be one implementation of the TPM_SaveState command. In addition through the use of, for example, one of the additional monotonic counters provided by the MTM specification it is possible to protect against replay attacks, see also subsection on replay protection for DRM in Section 2.4.3.

SIMLock software

The secure boot process will prevent any attacks to the SIMLock software stored in non-volatile memory, but the current solution does not prevent all attacks to the SIMLock executing environment. If we can isolate the SIMLock software process in the processor, we can prevent other processes to use the MTM since we have bounded the keys using the usageAuth secret.

Network software

Similarly to the SIMLock software we can measure parts of the Network software at boot, but we have no possibility to prevent attacks to the executing environment. The attack of introducing a second SIM can be prevented but that depends on if there exists a secure channel between the network software and the SIMLock software.

SoC side channel attacks

The use case example realization does not directly address any side channel attacks.

2.2.4 Alternative solution

Another solution is to store the SIMLock data signed using a public/private key pair. The private signing keys for each of the five personalization categories are protected by the MTM by using the SIMLock control keys as authData for the keys. When the SIMLock data needs to change, it is re-signed, which requires use of the authData (hence entry of the control keys, or otherwise sending them to the platform such as by SMS). Each change in a pattern is also reflected by an increase in a monotonic counter, whose new value is signed along with the new IMSI pattern. The SIMLock software contains the public key(s) needed to *verify* the pattern signatures when checking the IMSI at boot time. Those public keys are protected by a RIM for the SIMLock software. For example, there is a general RIM for the code of the SIMLock software, which is common across devices, and an additional device-specific RIM for the hash of the public keys that are specific to that given device. As this RIM is device-unique, it has to be created by MTM_InstallRIM after the initial pristine boot.

The most critical operation of the SIMLock software is the boot-time check of the IMSI number to the SIMLock patterns. So if the boot-time RIM certificates are validated true the software is unlikely to be successfully attacked during boot. The attacks that appear possible would be:

a) After boot, create a rogue signed IMSI pattern which is to be used during the next boot.

b) After boot, wind back to an old signed IMSI pattern which is to be used during the next boot.

c) After boot, create a rogue signed IMSI pattern with the “wrong” private key, and persuade the SIMLock software to verify the pattern at the next boot using a “wrong” public key.

d) After boot, otherwise tamper with the SIMLock software, and create a rogue RIM for the tampered software so that the device still starts up at the next boot.

For a) to work the attacker needs to have knowledge of the relevant SIMLock control key since it is the authData for the relevant signing key. Similarly for c) and d), the attacker needs the have knowledge of the verificationAuth data to be able to create a new RIM certificate. For b) to work the attacker needs to rewind a monotonic counter protected by the MTM.

The protection of the SIMLock software and other data are as same as the preceding solution and hence the attacks and protection for those are the same.

2.3 Secure Channel between Device and UICC

2.3.1 Use case description

A secure channel is needed to achieve end-to-end protection for the communications between a smartcard a terminal. Many applicative use cases need a secure channel, and example use cases can be found in ETSI SCP TS 102 412.

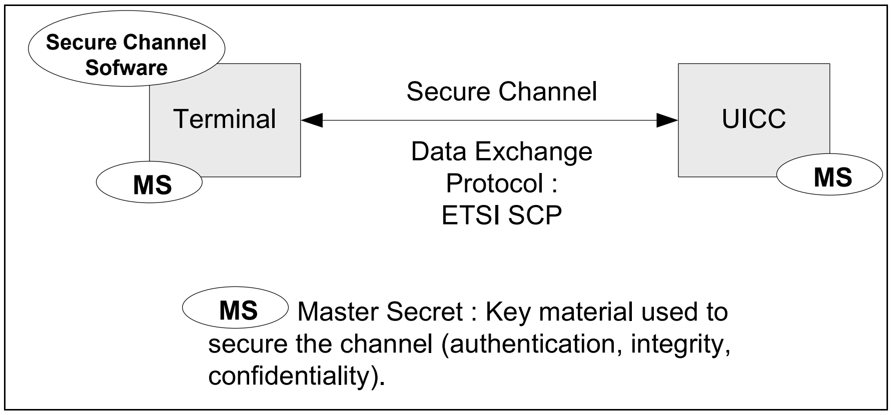

Secure Channel Sofware
Terminal
MS
Secure Channel
Data Exchange Protocol : ETSI SCP
UICC
MS
MS Master Secret : Key material used to secure the channel (authentication, integrity, confidentiality).

Figure 2: Secure channel architecture between terminal and UICC.

A secure channel implements solutions to provide:

- Mutual authentication between a UICC and a terminal end point.
- Integrity and/or confidentiality (encryption) protection of the communication between a UICC and a terminal end point.

Security is achieved only if the system provides:

- A secure end on the Smart Card [UICC] side (this is true by assumption; the Smart Card is a tamper resistant device).
- A secure end on the terminal side. This is reachable with a MTM as we will see in the next sections.
- A secure Master Secret set-up procedure prior to the establishment of a secure channel, and secure procedures for handling the Master Secret prior to, during, and after the secure channel establishment

For a Master Secret to be set up, a mechanism is needed to share key material between the two endpoints. ETSI SCP defines four mechanisms for key agreement that can be used irrespective of which secure channel type is used:

- Strong Pre-shared Keys - GBA.
- Strong Pre-shared Keys – Proprietary Pre-agreed keys.

- Weak Pre-shared Keys – Proprietary Pre-agreed keys.
- Certificate exchange.

We focus on the two non-proprietary solutions: GBA and Certificates. For all pre-shared key agreement mechanisms the Master Secret is denoted Ks_local. A Master Secret set-up is a rare event, which occurs only when the mobile and the SIM are sold to the customer or when the customer changes his/her subscription while keeping the same mobile phone, or conversely, changes his mobile phone while keeping the same subscription.

GBA is specified by the 3GPP TS 33.110: Key establishment between a UICC and a terminal. The Annex of this section gives an overview of the detailed procedure used to establish the shared key (i.e. Ks_local) between the UICC and the terminal.

Alternatively, the Master Secret set-up process can be managed by using public key cryptography. In this case, the USIM and the terminal each has its own key pair and a certificate. It needs to be mentioned that regardless of whether the Master Secret (Ks_local) is established by GBA using pre-shared keys or by use of public key cryptography, a series of pre-requisite processes need to take place before either of these methods can establish the Master Secret. The first of the pre-requisite process is a prior GBA process of GBA_U type involving an 'enhanced' capability UICC. The other, subsequent pre-requisite process is that of Security Association. The figure 2 in Annex 2.4.5.1 depicts the sequence of processes that culminate in the Secure Channel establishment. The GBA_U process involves the UICC, the UICC-holding mobile phone terminal, and a Bootstrap Server Function (BSF), and establishes a GBA key (denoted Ks) between the UICC and the BSF. The GBA key (Ks) is a concatenation of CK and IK, where CK is the ciphering key and the IK is the integrity key, both 128-bit long. The subsequent Security Association process involves the UICC, the mobile phone terminal, the BSF, and a Network Application Function (NAF). The process establishes a NAF-specific shared key called Ks_int_NAF between the UICC and the NAF, and another related NAF-specific shared key called Ks_ext_NAF between the mobile phone and the NAF. Only after these two prior processes (GBA_U, and Security Association) are securely completed can the process of Master Secret set-up take place. Therefore, in order to secure the Master Secret set-up process, one must also secure the GBA_U and the Security Association processes, the secret keys and other sensitive information involved in these processes. Examples of such sensitive information are the various IDs (for UICC, Terminal, NAF, UICC application, Terminal application, the Transaction ID), Key Lifetime and Counter Limit parameters, and possibly even nonces. These keys and other sensitive information can thus be considered as the 'requisite, intermediary secrets' that need protection by the MTM and the trusted mobile phone. In cases where public key cryptography is used for the Master Secret set up, the private keys should also be additionally considered as one of the intermediary secrets that are needed to be protected, in order to secure the set up process for the Master Secret Ks_local. In addition to protection of these "intermediary secrets", it is also needed to protect the integrity of the software that handles procedures related to the establishment of these intermediary secrets.

We note that here that in some cases, such as when the public key pair used in the HTTPS tunnel process is pre-provisioned and sealed to pristine platform integrity by the manufacturer, the process of protecting the secure channel process, especially the Master Secret set up process, can be simplified by forgoing some of the protective measures for some of the 'intermediary secrets' referred to in the above.

Secure channel establishment happens when requested by an application. The key used for confidentiality and integrity of messages is derived from the Master Secret. This derivation is out of scope of this document. The requirements for the Master Secret set up are the following:

- The terminal and the home network shall be able to authenticate each other.
- The home network shall be able to control whether this terminal has a genuine OS (through MTM attestation) and is authorized to establish a shared key with the UICC.

- The Master Secret shall be stored in a trusted way in non-volatile memory of the terminal (through the MTM seal mechanism) and accessed only by the secure channel software.
- Secret keys derived in previous GBA_U processes, PKI private keys, and any other credentials, which are needed to derive the Master Secret (Ks_local), shall be stored in a trusted way in non-volatile memory of the terminal (through the MTM seal mechanism) and accessed only by the secure channel software.

The requirements for the secure channel establishment are

- The Master Secret set up has been previously performed in the terminal and in the UICC.
- The terminal OS is genuine (and verifiable using the MTM PCR mechanism).

The OS offers the secure channel service to relevant applications in the terminal which use this protection during message exchange with the UICC. Section 2.3.3 gives some details on possible implementations, focusing on the TCG aspects, i.e., the MTM functions which have to be used. Several implementations are possible, using different MTM functionalities. The interoperability requirements are that secure channel establishment on terminal or UICC side is independent of the type of terminal and the type of UICC. This aspect is out of the scope of this note; it is achievable through compliance with standards from ETSI, 3GPP and GP.

2.3.2 Threats

When we analyse threats against different kind of applications, it is not sufficient to consider only software attacks. Some use cases for the secure channel (i.e. DRM) confer strong incentives for the user of the phone to break the secure channel. This means that the device owner or user is a potential hacker. Being in possession of the device, he can attempt hardware attacks.

In the next section, we will investigate the following attacks:

1. Attack on the UICC-terminal link
2. Attack by insertion of malware in the terminal

Others attacks described below are considered as physical attacks, needing specialized tools and adequate manpower, and are considered out of scope.

3. Attack on the external non-volatile storage
4. Attacks on external volatile storage
5. Attacks on the Secure Channel software
6. SoC side channel attacks

2.3.3 Mapping to MTM concepts

After normal power up, we assume that we have a set of PCRs which describes the trusted configuration and have been verified using RIM-Certs during the boot. We consider that secure channel software is a part of the OS software, thus protected within the secure boot process.

2.3.3.1 Verification of terminal- OS integrity by a server

In both solutions (GBA and Certificates) it is necessary for a server to verify the integrity of the terminal and of the OS. MTM attestation functions ("TPM-quote" command) give this possibility. But in this case, the server has to be aware of mobile brand, type, etc in order to know what are the PCR relevant values, since these values with input data are signed by the "quote" command. In other words, the server has to be part of a "TCG-aware" infrastructure. Thus, an additional key-pair for signing input data with its public half signed by an Attestation Identity Key (using TPM_CertifyKey) can be generated in a primary phase and sealed to the platform state.

2.3.3.2 Protection of Secure Channel software

After the secure boot process, the correct Secure Channel software is running in the engine. This software provides the confidentiality and integrity services requested by higher level applications to communicate with the UICC.

The Master Secret used by the Secure Channel software may be bound (through usageAuth) to the Secure Channel software which has to execute the TPM_Seal and TPM_Unseal commands, to mitigate the risk of an attack of the executing environment, although it can be also be considered that this binding is implicit trough isolation capabilities provided by the OS and prevention of access by other applications to the MTM commands acting upon the Master Secret.

2.3.3.3 Protection of the Master Secret

We consider a solution, where the Master Secret (MS) is stored in the non volatile memory. The MS is sealed to the PCR values for the trusted boot. In this way the key is only available to the secure channel software if the platform has boot to a trusted state and execute the genuine Secure Channel software. When a legitimate application requests secure channel services, the MTM is asked by the secure channel software to unseal the MS and provide it to the Secure Channel software. The unsealing operation will only be successful if the PCR values are verified

2.3.3.4 Protection of the Intermediary Secrets

We also consider a solution where the intermediary secrets that are needed to set up the Master Secret are stored in the non volatile memory. The intermediary secrets that are protected may include:

- Phone Terminal ID (denoted Terminal_ID)
- Terminal application ID (denoted Terminal_appli_ID)
- UICC ID (denoted ICCID)
- UICC application ID (denoted UICC_appli_ID)
- NAF ID
- Bootstrap Transaction ID (denoted B-TID)
- IP Multimedia Private ID (denoted IMPI)
- GBA Keys set up from the Security Association process (denoted Ks_ext_NAF)
- Key Lifetime values for Ks_ext_NAF
- Counter Limit values (denoted Counter Limit)

- Nonces (there are several nonces that are either generated from the mobile phone and transferred out of it, or received by it, including those denoted RAND and RANDx, depending on which process the nonce is generated)
- Authentication information generated by the Authentication and Key Agreement (AKA) algorithm (denoted AUTN)
- Private key of a public key cryptography key pair that is needed to establish a subscriber certificate, which in turn is used to establish a HTTPS tunnel, whereby the Master Secret is transferred from the NAF to the Mobile Phone Terminal.

In our solution, all or a selected set of the above intermediary secrets (where the selection satisfies a certain security requirement or a policy) are sealed to the PCR values for the trusted boot. In this way these intermediary secrets are only available to the software that needs to perform the GBA_U process and the Security Association process if the platform has boot to a trusted state and executed the genuine such software. Note that the software needed for the GBA_U and the Security Association could be implemented in the same unit as the software needed for the Secure Channel establishment, or they could be separate sets of software.

Note that the sealing of the intermediary secrets can be done either separately from the sealing of the Master Secret, or it could be combined in one sealing operation. The PCR values that could be sealed with the intermediary secrets could be the OS, the software for the Secure Channel, the software for the GBA_U and Security Association, or any appropriate combination thereof.

When a legitimate application requests secure channel services, the MTM is asked by the secure channel software to unseal the intermediary secrets, and provide the secrets to the Secure Channel software. The unsealing operation will only be successful if the PCR values are the expected ones.

2.3.4 What is achieved

OS integrity distant verification:

The MTM with the attestation capabilities provides a means for a distant server to control this integrity

Secure storage of the Master Secret (Ks_local) and intermediary secrets

The MTM provide us with a secure boot, once the MTM software is running in the processor. The sealing of the Master Secret (MS) and intermediary secrets ensures that a mobile phone can access Secure Channel Master Secret only if the phone platform is in a trusted configuration.

Software integrity

Integrity of the Secure Channel software as well as any software that is responsible for generating the intermediary secrets

This can be achieved either through the secure boot process where the software for Secure Channel and the software for the set up of intermediary secrets are implemented as part of the OS or separately protected after the boot process.

2.3.5 Annex: GBA and Certificate options

These options are described in 3GPP/SA3 and ETSI/SCP documents, and summarized here.

GBA option

3GPP/SA3 has defined a standard procedure (Generic Bootstrapping Architecture) called GBA for provisioning session keys between the UE and a NAF (Network Application Function), cf. 3GPP TS

33.220. GBA gives a solution for key establishment which doesn't need public key cryptographic capabilities in the UICC.

An enhanced version called GBA_U (TS 33.220 [3]) is used to provision a master secret between a UICC and a Terminal (i.e. Ks_local, as noticed in this standard). The GBA_U key Ks_int_NAF is used by the UICC and the NAF to derive Ks_local. The NAF securely delivers Ks_local to the Terminal through a TLS tunnel, which is established between the NAF and the Terminal.

The GBA_U procedure allows session keys to be established between the UICC and the NAF.

- in a UICC : Ks-int-NAF, and Ks-ext-NAF from data including a challenge coming from the home network, and with the authentication key of the UICC
- Ks_local is sent securely to the terminal through an HTTPS protocol.

The procedure is summarized in the drawing below.

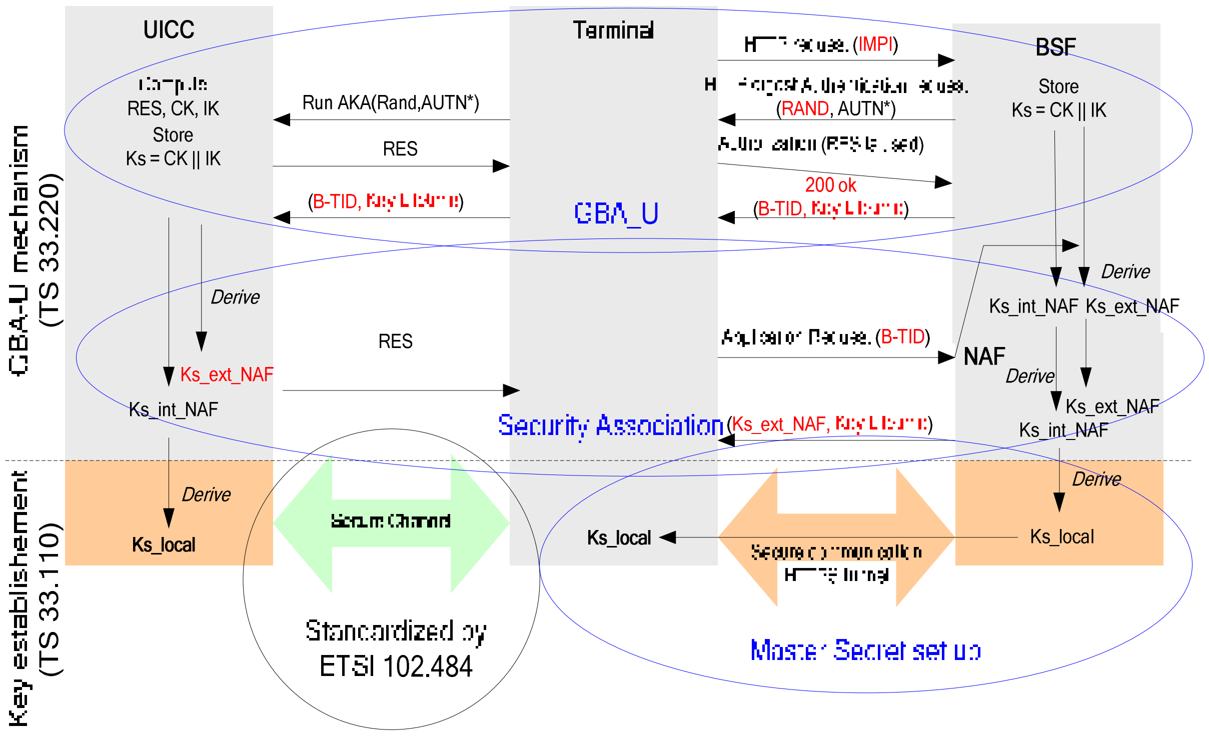


Figure 3: Secure Channel Process and prerequisite processes for it

NAF server has to perform terminal OS integrity check before sending Ks_local. This check (based on attestation functions offered by the MTM) has to be performed between the Terminal and the NAF server entity which has the necessary information to verify the relevant measures.

Note the above diagram shows some (but not all) of the “intermediary secrets” described in Sections 2.3.1 and 2.3.3. Note that in this diagram only IMPI, RAD, B-TID, Key Lifetime, and Ks_ext_NAF are depicted, but some of the other intermediary sensitive data are also handled by the Mobile Phone at various stages of the three processes, i.e., the GBA_U, Security Association, and Master Secret (Ks_local) set up.

Certificate option

ETSI SCP102484 describes this option considering that the certificates in the SIM and the Terminal exist prior to the handshake protocol establishing a master secret MS.

We consider here that prior to this phase, Terminal-OS integrity check (based on attestation functions offered by the MTM, and using the same key as for handshake) has to be performed between the Terminal and a Server entity The server has the necessary information to verify the relevant measures and the required keys to compute this certificate and send it to the terminal. The MS establishment is performed through a classical PKI based handshake for key establishment.

2.4 Implementing an Open Mobile Alliance DRM system

2.4.1 Use case description

Digital Rights Management (DRM) is a technology that enables the controlled consumption of digital media objects such as movies, games, music clips, ring tones, photos and streaming media. Analyzing a DRM system is highly complex as it consists of many functional entities which interact via a network. Several standardization bodies address DRM technologies [7] [8]. This analysis focuses on the OMA DRM perspective. OMA has released a number of DRM-related documents, including DRM requirements, DRM architecture and the specifications of protocols and mechanisms to implement a DRM system in a mobile environment.

The OMA DRM architecture document defines a functional architecture for DRM [9]. Of the architecture's functional entities, three are very important for implementing the architecture described:

- DRM Agent: This entity is a trusted element of the user's device that is responsible for enforcing permissions and constraints associated with DRM content, and for controlling access to DRM content. Each DRM agent owns an asymmetric key pair and a digital certificate that identifies it.
- Content Issuer (CI): This entity is in charge of the distribution of DRM content.
- Rights Issuer (RI): This entity issues Rights Objects (ROs) to OMA DRM-conformant devices. RO consists of a collection of permissions and constraints that are linked to DRM content.

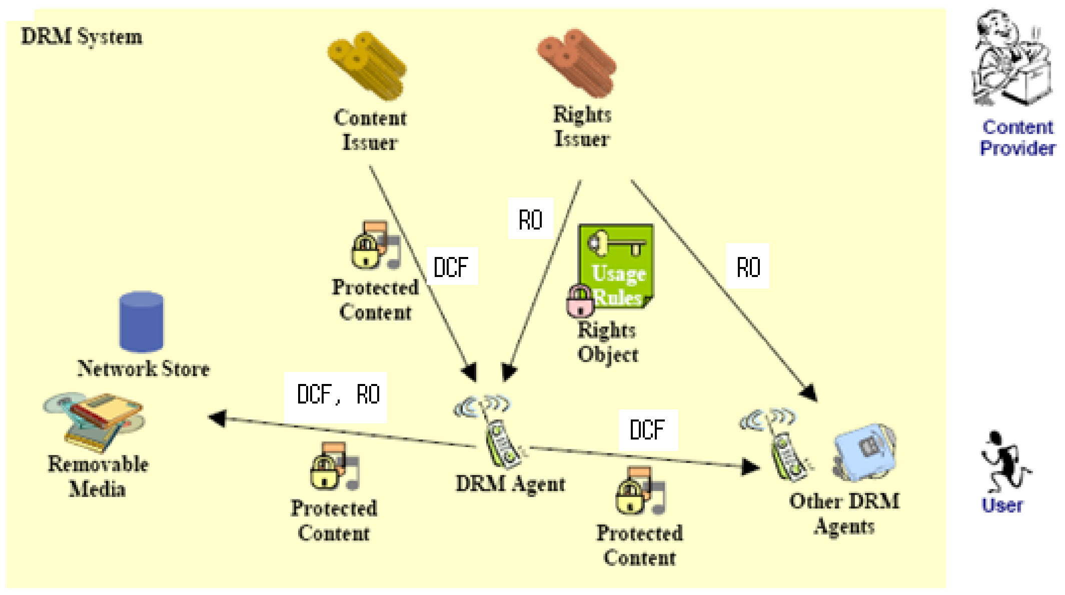


Figure 4: OMA DRM System

In Figure 4, distribution of DRM content generally consists of five basic steps:

1) content packaging,
2) DRM agent authentication,
3) RO generation,
4) RO protection, and

5) content delivery.

Before distributing content, the Content Issuer encrypts the content package with a Content Encryption Key (CEK), which is a symmetric key to protect it from unauthorized access. It is not possible to decrypt the content without the CEK. When a user wants to play the DRM content, the user's DRM agent contacts the Rights Issuer which generates a RO for that content. The RO contains the CEK used to encrypt the content and a list of permissions and constraints. The RO is encrypted by the RI with a Rights Encryption Key (REK) which is itself encrypted with the public key of the user's DRM agent. The DRM agent, using its private key, can decrypt the REK and decrypt the RO. If the user respects the terms of use contained in the RO, the DRM Agent decrypts the content with the CEK stored in the RO.

2.4.2 Threats

According to the OMA DRM specification [10], OMA DRM is built upon a PKI to support a "trust model". Using classical PKI, the DRM agent and the RI authenticate to each other using public key cryptography and digital certificates. For a DRM Agent and RI implementation, OMA DRM suggests applying various security mechanisms concerning confidentiality, authentication, integrity protection and key confirmation.

These mechanisms are used to secure the protocol; however, there can be potential threats when DRM is weakly implemented. Particularly, vulnerabilities in the communication between a DRM Agent and a RI may cause threats to occur.

Potential threats are listed as follows, in referring to the use case scenarios document [11]:

- Keys related to DRM can be extracted from the device and used to decrypt protected content in an unauthorized manner.
- The private key used to authenticate the DRM agent to service providers can be extracted and used to clone the agent.
- The data contained in a RO stored on the device can be manipulated in an unauthorized manner.
- The software relating to DRM on the device can be manipulated in an unauthorized manner or additional unauthorized software can be added so as to allow unauthorized access to protected content.
- Attribute values that relate to certain conditions in Rights Objects may be manipulated in an unauthorized manner. Prime examples are time/date and location. Such data is potentially subject to tampering.
- Decrypted DRM content can be extracted from the device in an unauthorized manner.

There are other potential threats to the DRM system, such as threat scenarios that deny service or compromise communication between DRM entities such as, for example, between the digital content provider and the RI. But these threats are not considered in this document. We only focus on the DRM Agent and attacks against it.

2.4.2.1 Compromised execution of the DRM Agent

An attacker may attempt to compromise an entity of the DRM system. In particular, a compromise of the DRM Agent may result in the disclosure of its private key, REKs, CEKs, and decrypted DRM content. This situation may also result in the loss of integrity protection of the replay cache of the DRM Agent and loss of the Rights protection stored in it. Additionally, a compromised rendering application in the DRM Agent may also result in the loss of DRM content.

If execution of the DRM Agent is not secure, an attacker may be able to spy on the DRM Agent or disturb it during its execution. Depending on the implementation of the DRM Agent, an attacker can mount several attacks by targeting various elements including volatile memory (if he/she has a physical access to it or if he/she can bypass the memory space isolation enforced by the OS between processes), the data bus between the memory chip and the CPU, or even the CPU itself (although this attack is quite sophisticated). If an attacker can spy on the DRM Agent, he/she can easily retrieve sensitive data or decrypt DRM content, therefore bypassing subsequent restrictions.

2.4.2.2 Disclosure and manipulation of sensitive data

The DRM Agent owns a public/private key pair to authenticate itself to Right Issuers and which is used by Rights Issuers to distribute the REK securely to the DRM Agent. If an attacker obtains the private key of a genuine DRM Agent (for instance if the attacker has a physical access to the non-volatile memory where it is stored, or if the attacker can snoop the data bus that connects the non-volatile memory to the CPU where the cryptographic operations are performed, or if the attacker can bypass the access rights enforced by the operating system on the non-volatile memory space, etc.), the attacker can use this knowledge to retrieve the CEK (by decrypting the REK) and therefore decrypt the contents without respecting the usage restrictions. The attacker can also spoof the identity of the DRM Agent in future communications with Rights Issuers.

Rights Objects can also contain usage counters used to limit the number of times the content can be used. If an attacker can manipulate the non-volatile memory, it can roll back the counters of Rights Objects to an old state and thus it can bypass the limit restriction. Similarly, the clocks in standard computing platforms are relatively unprotected. An attacker may be able to manipulate the clock by, e.g., resetting it, or by causing a drift, and thus can circumvent time/date related restrictions in usage rights. The OMA DRM requires use of trusted time information in several of its protocols. Hence, there is need for the mobile phone to protect time information obtained from both the mobile phone’s internal clock and trusted external sources (e.g. OCSP server) for a robust implementation of DRM.

If the code of the DRM agent is stored in the non-volatile memory and the attacker can manipulate this memory, it can for instance also modify the code of the DRM agent in order to bypass usage restrictions.

2.4.3 Mapping to MTM commands

The threats analyzed in the previous section can be mitigated by embedding an MTM into the mobile device. Attestation, secure storage and secure boot are useful functionalities that MTM can provide for robust DRM implementations. In this section, we assume that the DRM Agent and its asymmetric key pair are loaded onto the platform at the same time as the operating system during the manufacturing process.

2.4.3.1 Secure execution of the DRM Agent

A correct instantiation of the DRM Agent can be guaranteed using the secure boot process. The operating system is in charge of the enforcement of the isolation between processes to guarantee that a process cannot spy on or modify the memory space of the DRM Agent. The OS is securely booted owing to the MTM. As mentioned earlier, a compromise of the DRM Agent or rendering application may also result in the disclosure of its private key, REKs, CEKs and decrypted DRM content.

To mitigate the risk of run-time modifications of the code of the DRM Agent or rendering application (due to a hardware attack or a bypass of the isolation between processes), the DRM Agent can be monitored periodically by a Secondary Runtime Measurement Verification Agent (SRMVA). Alternatively, the monitoring can be triggered by particular events, so that the integrity of the DRM Agent and rendering application can be verified according to one or more of the following timings; at the access to the Rights

Objects, at the check if the Rights Objects for the content package has rights to play, at the start of decryption of the content package, and after decrypting every prescribed number of encrypted frames in the content package.

In the OMA DRM architecture, the UICC can be used to authenticate a user if the usage restrictions and permissions bound to some DRM content are associated with a user identified by the UICC. In this case, the secure channel between applications and the UICC (described in another use case) can be applied to prevent an attacker from targeting the link between the DRM Agent and the UICC.

2.4.3.2 Data protection

Sensitive data (e.g., private key, CEK) can be maintained securely by storing it in the protected storage. If a DRM Agent's private key is stolen, it can be used by an attacker to decrypt the REK and therefore decrypt the CEK. If the public key pair is generated by the MTM and secured in protected storage, the possibility of attack by stealth can be greatly reduced.

To further mitigate the risk of theft of the DRM Agent's private key, the key can be sealed with the TPM_Seal command to the correct instantiation of the platform and of the DRM Agent. Thus, an attacker cannot decrypt the private key even if he/she has access to the non-volatile storage. Only the genuine TPM Agent running in a genuine configuration can retrieve it with the TPM_Unseal command.

2.4.3.3 Protection against replay attacks

Stateful rights objects stored in the DRM Agent non-volatile memory can be protected against manipulation with one of the monotonic counters provided by the MTM specification. However, the MTM specifications require only three monotonic counters which cannot be used to mitigate these replay attacks, and other counters are optional. If the MTM provides the platform with only one of these additional monotonic counters, and if the maximum value of this counter is sufficiently high, the operating system would be able to provide applications with as many virtual monotonic counters as they need. With these monotonic counters, the DRM Agent would be able to protect the Rights Objects against rollback attacks. The virtual monotonic counter mechanism can be easily implemented by the operating system. The operating system maintains a list containing the values of the virtual monotonic counters. Each time a virtual monotonic counter is incremented, the operating system increments its MTM monotonic counter, generates a hash of the value of the new MTM monotonic counter and the values of the virtual counters and seals this hash with the TPM_Seal command. When an application reads the value of a virtual counter, the operating system unseals the hash, verifies its value by computing it again with the current value of its MTM monotonic counter and the values of virtual counters and, if the verification succeeds, returns the value of the virtual counter to the application. However, this solution depends on the implementation because the maximum value of the real monotonic counters can be as low as 4096. In such a case, the operating system may for instance when being shut down store the virtual counters in memory and only then protect them with the real counter.

2.4.3.4 Protection against clock and time manipulation

There are at least three distinct time-information fields defined for use in OMA DRM. They are the DRM Time, the Rights Issuer Time Stamp (RITS), and the "producedAt" field in the On-line Certificate Status Protocol (OCSP). These are all used to provide an accurate measure of date and time to the DRM Agent and to enable protection of DRM content by the DRM Agent. There are two fundamentally different options to support the secure enforcement of rights that relate to certain time statements. The first assumes that the DRM agent has access to a source of real time values on the device. The second makes use of the optional feature of a tick counter provided by the MTM.

The MTM specification by itself does not provide a secure time source which would allow issue of trusted time statements to the DRM Agent or an external entity, e.g., the Rights Issuer. However, the MTM offers an ability to secure the operation of a Real Time Clock (RTC), often found in a mobile phone, and provide a protected storage capability to protect the time information using such commands as TPM_seal and

TPM_unseal. When the mobile phone obtains trusted time information from external sources (such as a NTP server or an OCSP server) in the RITS or the OCSP “producedAt” time-stamp, it can re-synchronize its own internal reference time and such protected time information may subsequently be used by the DRM Agent. Since the MTMs do not have independent power sources, and since the power to the mobile phone itself may be turned off and not made available to the RTC all the time, a DRM Agent on the phone may be required to frequently store available trusted time information in non-volatile storage, using such commands as TPM_seal. When power is restored, the DRM Agent may be required to first retrieve such “last known trusted time”, using such commands as TPM_unseal, and then re-synchronize with an external trusted time source. Only then may it be allowed to proceed with DRM applications and consumption of Content.

On the other hand, the MTM offers a protected, read-only Tick Counter Value (TCV), initialized at each TPM reset. The TCV may be made useful to DRM Agents handling right objects that have relatively short time durations. The counter session is uniquely identified by the Tick Session Nonce (TSN) a random number generated by the MTM at TCV initialization. Association of the TCV to a UTC time can be achieved as described in chapter 20 in the TPM Main Part 1 Design Principles specification [15]. This operation envelops TSN and a time stamp from an external Time Authority (TA) with two signed TCV values via the TPM_TickStampBlob command. This can be used to produce a digitally signed assertion, an Accuracy Statement (AS), that the UTC time of the time stamp lies between the actual times of the first and second TCV, as described in Section 20.3 of [15]. An AS can be used at two stages in the DRM scenario. In a first variant, the RI may request an AS, and a signed TCV, from the DRM Device during the execution of the ROAP. In this way, the RI obtains assurance that the DRM Agent has access to an accurate, un-tampered time. The RI can then code time limits in usage rights directly in terms of TCVs in the RO. The DRM Agent can then compare the actual TCV to the limits expressed in the RO and exert access control to the content accordingly. In a second variant, the DRM agent can use an AS generated at boot time to make sure that TCVs are bound to an accurate external time, and belong to the same session of the Tick Counter (by comparison of TSNs in the AS and the MTM Non Volatile Memory). Again, the DRM Agent has a valid time-to-TCV association to base access control to content on.

2.4.3.5 DRM Agent in the UICC

Instead of being implemented in software on the mobile device, the DRM Agent can also be implemented in the UICC. In this case, the intrinsic security provided by the UICC can mitigate many of the aforementioned threats (for instance, the integrity of the DRM Agent and its execution cannot be altered, and the critical keys cannot be stolen). However, the DRM Agent must check the state of the platform before giving the CEK to an application running on the mobile device. The transmission of CEKs between the UICC and the applications must also be protected to prevent an attacker who may monitor the physical interface between the UICC and the mobile device via sniffing. These issues can be mitigated by using a secure channel between applications running on the mobile and the UICC (cf. secure channel use case).

2.4.4 Potential enhancements

2.4.4.1 Secure execution environment

The MTM protects the DRM system from any attacks and entity compromises.

2.4.4.2 Secure storage of important data

The MTM provides hardened PKIs via sealing and unsealing of the protected storage.

3 USE CASE ANALYSIS: BRIEF EXAMPLES

In this section, we have collected some of the Use Cases which are not discussed in such detail as the others. The reason for this brevity is the multitude of possible solutions and also that the solutions depends heavily on the prerequisites and the surrounding infra structure assumptions.

3.1 Mobile Ticketing

3.1.1 Use case description

Mobile ticketing is a convenient way of storing and using electronic tickets. A ticket can be seen as a token for access to a particular event or as a token for usage of some resource. As an example of the first case we might consider a cinema ticket or a concert ticket and for the second case we have bus tickets which might be used for a specific area but not bounded to a particular departure time.

Traditionally, an electronic ticket can be issued in two ways[2]; either the token is stored at the event provider's (EP) server in which case the customer only needs to prove his/her identity to the server in order to use the ticket, or the token is transferred to the customer at the time of sale. These different schemes have different properties and which one chosen depends on the needs. The first one can be used in cases where the EP sells "seats" which is non-transferable, e.g. airplane tickets. One additional property with this scheme is that the EP has a list of all the names of the users, which can be good or bad, depending on your standpoint. The second scheme offers transferable tickets as the token need not be bounded to the identification of the customer.

A central issue with the first scheme is that the user needs to prove his/her identity at ticket usage. In many cases the EP will not accept *identification of customer* as equal to *customer knows this PIN-code*, e.g. airline companies. For more relaxed security setting, the EP might accept to sell a ticket to *whoever knows the PIN of this SIM-card*. Then one could build a ticketing system based on secrets stored in the SIM card, but this seems more difficult to launch since it would probably need some operator involvement and an EP would need to make its system compatible with a large number of operators.

For this use case analysis, we will consider the second scheme with a token stored on the ME and one solution is to have a ticketing application (TApp) running in the ME. The TApp will use the MTM to attest to its state and also to store tickets bounded to the trusted state.

The ticketing provisioning and usage will be something like this:

1. The customer buys the ticket from the web using a personal computer or the ME.
2. The ticket is sent to the ME and the customers account is billed.
3. At entrance (or usage in case of e.g. a bus ticket)
    a. The EP needs to verify the ticket.
    b. The EP needs to invalidate the ticket.
    c. The customer wants to be sure that only the correct EP can invalidate the ticket.

[2] Additionally we could solve the mobile ticketing problem using the same techniques as used in pre-paid cash cards available in some countries. Each event defines a new currency and the ME could store units of money in many different currencies. Buying a ticket is then equal to exchanging one currency for another and the exchange rate is the ticket price. This scenario is not considered in the current document.

3.1.2 Threats

Most threats are concerned with the customer trying to cheat on the EP or some attacker cheating both the customer and the EP. The following threats are identified:

1. The customer or a rogue ME tries to use the same ticket twice.
2. The customer or a rogue ME mounts a replay attack.
3. An ME running rogue software mounts a man-in-the-middle attack and exports the ticket to another device leaving the legitimate customer to pay the ticket.
4. An ME running rogue software tries to duplicate the ticket and exports the ticket to another device.
5. The customer or a rogue ME tries to use a different ticket for the event.
6. A different EP invalidates the ticket by mistake.
7. A rogue EP sends an invalid ticket to the ME but still charges the customer's account.

3.1.3 Mapping to MTM concepts

When the ticket is about to be downloaded into the ME, the TApp is responsible for sending an attestation (using the MTM) to the EP server attesting to its trusted state. The server will then know that the TApp and the OS is the ones intended and the ME platform can be trusted.

The ticket consists of

a) An event code

b) A ticket serial number

c) EP public key and certificate

d) A MAC of the ticket signed using the EPs private key.

When the ticket is downloaded, the TApp will validate the EP public key and check the ticket MAC. Then it will store the event code in flash memory and seal the ticket to the trusted state using the MTM. By this, the ticket serial is protected from rogue software loaded into the ME at a later stage.

At usage, the EP connects to the TApp, possibly using NFC, and sends a signed message containing the event code and a random nonce. The TApp checks if the ME has a ticket for that event and asks the MTM to unseal it. Using the public key of the EP, the TApp validates the message from the EP and encrypts the serial number of the ticket together with the nonce and sends it back to the EP. The EP checks the nonce and marks the serial number as "used" and grants access to the customer.

Most of the threats are mitigated by the MTM's capabilities of validating the TApp and the OS environment. Also, the use of ticket serial numbers and random nonces in the protocol will prevent reply attacks and accidental misuse.

Threat 7 is perhaps the most difficult threat to combat, but by validating the EP's public key using a certificate together with the signed ticket, the EP cannot repudiate the ticket issuing later.

3.2 Mobile Payment

Mobile payment is a complex topic, with numerous actors and many realizations all over the world. Numerous alternatives for mobile payment are described for example by MOBEY Forum [1]:

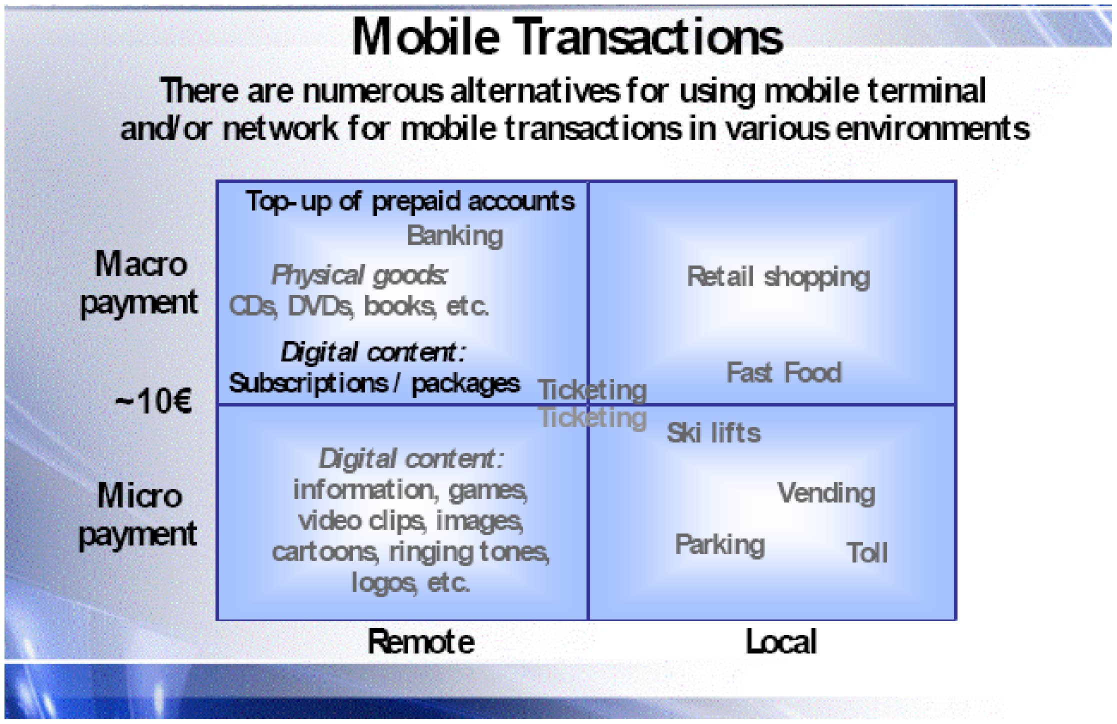


Figure 5: Categories for Mobile Transactions

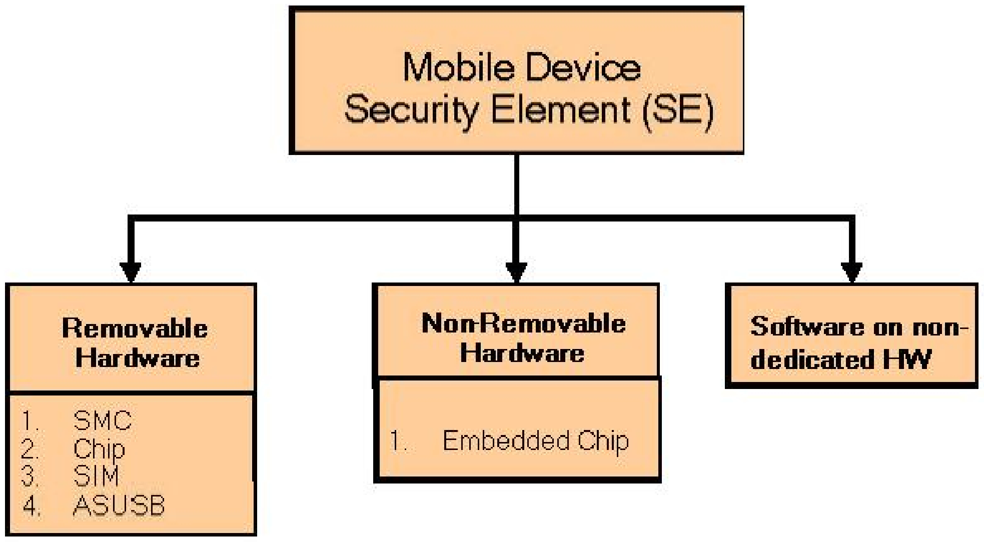
We will concentrate on the alternative which seems to be the most demanding in term of security of the handset: retail shopping, since it concerns higher value payment. This alternative requires practically a means of transmission with the POS (point of sale terminal) which must be simple, fast, and short range so the client can make payments with a wave or a touch on the POS. So, this alternative perfectly fits with the development of NFC mobiles.

Most of the remote usage of mobiles for payment are based on the identification and authentication of the mobile owner by the MNO, and don't generate so much security requirements for the handset.
For retail shopping, situation is totally different, and payment transaction has to comply as much as possible with the already existing payment infrastructures and their protocols. In Europe, EMV is the prevailing standard (a short explanation is given in the annex). In this case, there are much more security requirements since modifying the payment application located in the handset could result in important frauds.

Another aspect may be mentioned: often, for remote payment applications, the MNO plays the role of the Bank, whereas for local payment there is a clear share of responsibility between the MNO and the Bank, thus resulting in strong requirements for the isolation of the respective assets of these two actors, even if located in the same mobile device. This may have a strong impact on how delegation may work.

3.2.1 Use case description

Many implementations are possible, and, as far as payment is concerned, we have to focus on the SE (secure element) which is intended to protect the payment application. Several possibilities are described in MOBEY [2]; since this classification has been made in 2005, it doesn't refer to the MTM explicitly, but it appears clearly that the TCG approach fits with the "software on non dedicated hardware" case since what is called "dedicated hardware" is a piece of tamper proof hardware which is devoted to the payment application(s) and eventually others sensitive applications.



3.2.2 Threats

Examples of threats (referring to the EMV approach given in annex) are as follows:

1. Payment application is loaded OTA on an un-trusted mobile
2. Payment application is attacked by another application loaded on the same mobile, due to a lack of isolation between the two.
3. Mobile owner tries to modify the flow control mechanism, in order to perform transactions without authorization
4. Mobile owner tries to perform a roll back on some data including the counters, in order to perform transactions without authorization
5. Mobile owner tries to get the payment signature keys in order to use it on an un-trusted mobile
6. A virus (loaded on the mobile for example when browsing on the internet) modifies the amount paid (and signed) without modifying the amount displayed on the screen
7. A virus (loaded on the mobile for example when browsing on the internet) tries to catch the PIN when entered by the user, and sends it OTA to an organization

8. Receipt log file is modified after the payment is performed, for example for deleting transactions or modifying the amount paid.

Threat 1 could easily enable attacks 3, 4 and 5 since on an un-trusted mobile, attempting to integrity of software or confidentiality of keys is easy.

Threat 2 would have the same result as 1, and it could be the result of a failure in the mechanisms of delegation which could result in a lack of isolation of sensitive data from different application providers.

The goal of attack 3 or 4 would be to enable a user to pay in a shop whereas his bank account is empty!

The main goal of attack 6 or 7 could be for some organization to attack the Bank in order to fool their organization, and to destroy confidence of consumers in the Bank's means of payments.

Attack 8 could help the client to intend to repudiate a transaction to his Bank.

3.2.3 Mapping to MTM concepts

Scenario 1: software on non dedicated hardware

1) Threat 1 may be countered by using attestation functions or a derivation of this mechanism in case where the server is TCG-unaware (see other use cases)
2) Threat 2 is an issue of trusted OS and quality of this OS for installing different applications and their sensitive data (keys, certificates) and guarantying secure and isolated execution to these applications. Secure boot of the OS is the main TCG concept to be used here.
3) Threat 3 and 5 are a matter of integrity of the applications fully dependent of the OS quality, and of secure storage of their sensitive data. Here also, secure boot and trusted storage sealed to the platform state are the main TCG concepts to be used.
4) Threat 4 may be countered thanks to monotonous counters relying on the functionality offered by the MTM
5) Threat 6 and 7: same as 3
6) Threat 8: is a matter of secure storage where usage of TCG sealing mechanism could be useful.

Scenario 2: dedicated hardware (removable or not)

In this scenario, a large part of the payment application (code, keys, counters…) is embedded in a tamperproof chip; so, some threats disappear by assumption: this is the case of 3 and 4. The others threats remain unchanged, and measures to counter these frauds may be considered as the same. A specific point which can be mentioned is that there is a need of some binding between trustworthiness of the mobile platform and the dedicated hardware which would have to refuse payment transaction if this binding is not possible. See the secure channel use case.

3.2.4 Annex

The purpose and goal of the EMV standard is to specify interoperation between EMV compliant IC cards and EMV compliant credit card payment terminals throughout the world. EMV financial transactions are more secure against fraud than traditional credit card payments which use the data encoded in a magnetic stripe on the back of the card. This is due to the use of encryption algorithms such as DES, 3DES, RSA and SHA to provide authentication of the card to the processing terminal and the transaction processing center. The increased protection from fraud has allowed banks and credit card issuers to push through a 'liability shift' such that merchants are now liable (as from 1 January 2005 in the EU region) for any fraud that results from transactions on systems that are not EMV capable.

Compared to EMV payment with a card, EMV payment with a mobile will change some important aspects for the client: PIN entry will be on the mobile, payment details may be displayed on the mobile display instead of POS display, receipt can become electronic, with storage in the mobile, etc

The payment application on the client side can be described as below, given that many variations are possible, according to the specific requirements of payment organization and/or banks…

1. Get transactions parameters (amount, timestamp, merchant Id, etc)

   Mobile handset may display these parameters, and user, if he agrees, may accept by typing his PIN.
2. Owner authentication (Pin control; if it fails, transaction is aborted)
3. Flow control (i.e. consumption counters compared to ceilings) and decision to accept, or to ask for Authorization, or to refuse.
4. Authorization protocol (i.e. end to end security protocol with the Bank processing center for resetting counters). This step takes place according to the decision taken in step 3 and the decision of the merchant terminal. An authorization message is sent also directly by the Bank to the POS.
5. Transaction execution: increment counter and deliver a transaction with signature to the POS.

   A success status may be displayed on the mobile, and a receipt may be kept in its permanent memory.

POS terminal having received the signed transaction (including account number), verifies it and stores it (for further transaction settlement). Payment is done, and the transaction can terminate.

3.3 User Data Protection and Privacy

3.3.1 Use case description

Today more and more applications are bundled into mobile phones. They are used to make phone calls, to send text messages, to take photos and record short videos, to visit public web sites, to send and receive personal and enterprise email, to watch TV, to buy items (mobile payments), to take the bus (mobile ticketing), to access enterprise intranets, and so on. The mobile has become an indispensable apparatus in the life of its owner (user). However, if all of the user's actions and data be linked together, it would constitute a significant threat to the user's privacy.

This trend is amplified by the introduction of NFC (Near Field Communication) technologies and by Internet browsing, capabilities which are now common to many mobiles. This evolution has bestowed mobile phones with capabilities to be used for many types of transactions, whether by proximity (via NFC technologies) or over long distances (via mobile network technology such as GSM/UMTS).

Hence there is a strong incentive to provide the user with some assurance that his or her privacy is protected by preventing third parties from linking together the user's various actions and data or from linking them to the user's identity. However, there may exist also a need to allow some kind of traceability to counter misuses of a service and to maintain a possibility for revocation in the case of attacks resulting in the leakage of cryptographic material (e.g. keys). . This reasoning is similar to that for DAA as explained the DAA Section "Variable Anonymity" in [16].

Protection of the user's data is essential. The data storage size of the mobile is steadily and significantly increasing; consequently, users can and will store increasingly more and increasingly important data (address book, text messages, emails, photos, payment receipts, electronic tickets, enterprise data, etc.) on their handset. It is important to guarantee that these data will not be stolen if the mobile is stolen or if a malicious application is run on the mobile. Data should also be protected against unauthorized modifications of such applications.

3.3.2 Threats

Several threats can be considered in this use case:

1. Loss of confidentiality: A malicious application extracts (steals) the user data stored on the mobile
2. Loss of integrity: A malicious application modifies the user data that are stored on the mobile
3. Loss of authorisation: A malicious application records all the actions of a user and sends the recording(s) to a third party
4. Loss of authorisation: A third party, needing some private data interacts with the mobile, and links the interaction with data obtained by itself or other third parties
5. Loss of availability: A thief steals the mobile of a user and tries to access the user's data

3.3.3 Mapping to MTM commands

Threats 1&2: To avoid the theft of user data by a malicious application, these data can be sealed (with the command TPM_Seal) to the correct instantiation of the operating system or the application that manipulates them. After these data are unsealed, the operating system (which is supposed to be honest and securely booted thanks to the MTM) is responsible for enforcing the access restrictions between applications so the user data cannot be stolen while being manipulated in memory. In the case where private data is stored and managed in a secure element such as the SIM, it remains the need of trusted user interface for the user to accept or reject information request, and thus the need of a trusted platform with secure boot and trusted UICC/Platform link.

Threat 3: To prevent a thief from extracting user data from a stolen mobile, several solutions exist depending on the storage location of the data. If the data are stored on the SIM card, they can be configured to only be accessible after the PIN code is entered. To protect data stored on the mobile phone or a memory card inserted in the phone, they must be only accessible after the authentication of the mobile owner. To achieve this goal, the data are encrypted and the encryption key is sealed (with the command TPM_Seal) to the correct instantiation of the OS or of an application. In addition, before issuing the TPM_Unseal command, the operating system or the application authenticates the user by asking a password or a PIN code (the password or PIN code, or hash thereof is also securely stored). The data can also be sealed to the presence of the user's SIM card (cf. the SIMLock use case).

Threat 4: Traditionally, the mobile execution environment does not protect very well the privacy of the user. Visited mobile networks may have complete knowledge of the mobile subscriber identity[3], the place (cell) where he is currently, all the calls, etc. However, privacy becomes an important issue with the development of new transactional usages. It is possible, and important, to allow the users to protect their privacy during transactions performed with their mobile, and to minimize the amount of information given in order to prevent further misuse of private information. For instance, proving that "I am more than 18 years old" does not require revealing the exact age.

In a TCG-aware environment, the first way is to use the DAA (Direct Anonymous Attestation) feature, as in PC platforms, to protect the privacy of the user during MPM attestation. The DAA allows proving that an AIK (Attestation Identity Key) was generated by a genuine TPM/MTM without revealing the corresponding EK (Endorsement Key) or using a privacy certification authority. The DAA signature mechanisms could also be used to give a proof that a piece of data is present in the mobile platform, or that a process (like verifying that the age > 18) has been performed in the trusted platform. However, the DAA mechanism is optional in the MTM specifications and so it may not be available in all mobiles. Also DAA processing is split between layers and resides partly in the TSS. Hence DAA processing is not entirely realised in the MTM.

Other access restrictions and cryptographic protocols (based on blind or group signature for instance: "I can prove that I am a member of the group of users that have a genuine trusted mobile containing a piece of data (meaning for example that I have subscribed to this service) without revealing my identity") may be used to achieve privacy. These protocols can be run by applications and the correct work of these applications can be guaranteed by the MTM using the secure boot feature. They can also use the MTM to securely store the information they need to run the protocols.

Threat 5: Through the use of backup copies the availability of user data can be secured, measures like this are however out of scope of TCG. Furthermore, user data may be stored in protected form so it only can be recovered through a user authentication process. Here the MTM can play a role to guarantee the integrity of this authentication process.

[3] In GSM/UMTS networks the subscriber's identity (IMSI) is normally replaced by a temporary one (TIMSI) and it is this TIMSI that normally is used.