OMTP

ADVANCED TRUSTED ENVIRONMENT: OMTP TR1

VERSION:	v1.1
STATUS:	Approved for Publication
DATE OF PUBLICATION:	May 28th 2009
OWNER:	OMTP Limited

CONTENTS

The information contained in this document represents the current view held by OMTP Ltd. on the issues discussed as of the date of publication.

This document is provided “as is” with no warranties whatsoever including any warranty of merchantability, non-infringement, or fitness for any particular purpose. All liability (including liability for infringement of any property rights) relating to the use of information in this document is disclaimed. No license, express or implied, to any intellectual property rights are granted herein.

This document is distributed for informational purposes only and is subject to change without notice. Readers should not design products based solely on this document.

Each Open Mobile Terminal Platform member and participant has agreed to use reasonable endeavours to inform the Open Mobile Terminal Platform in a timely manner of Essential IPR as it becomes aware that the Essential IPR is related to the prepared or published specification. The declared Essential IPR is publicly available to members and participants of the Open Mobile Terminal Platform and may be found on the “OMTP IPR Declarations” list at the OMTP Members Access Area.

The Open Mobile Terminal Platform has not conducted an independent IPR review of this document and the information contained herein, and makes no representations or warranties regarding third party IPR, including without limitation patents, copyrights or trade secret rights. This document may contain inventions for which you must obtain licenses from third parties before making, using or selling the inventions.

Defined terms and applicable rules above are set forth in the Schedule to the Open Mobile Terminal Platform Member and Participation Annex Form.

1 INTRODUCTION

1.1 DOCUMENT PURPOSE

This document follows the successful completion of the 'OMTP Trusted Environment: OMTP TR0' [1] recommendations. In preparing the TR0 recommendations, OMTP took a pragmatic approach in terms of working closely with leading operators, hardware vendors (silicon manufacturers as well as intellectual property suppliers), software vendors and Handset manufacturers to produce a set of recommendations that are solid and achievable in already existing silicon platforms. The development of the Advanced Trusted Environment (TR1) followed the same pragmatic approach aiming at delivering a more comprehensive security roadmap. Such a roadmap is aligned with current Threats, business opportunities and the overall industry objectives of achieving hardware-backed security, Defragmentation, a flourishing ecosystem of Applications and wide economies of scale.

TR1 can be seen as a continuation of the work done in TR0 by defining new technological enablers and applying an extended Threat Model. The mechanisms described within TR0 remain applicable for Devices that also implement TR1.

TR1 complements the role of the UICC to allow the UICC to securely access the Mobile Equipment facilities, securing I/O and making available to UICC the processing power of the main Mobile Equipment processor. We note that the UICC remains the primary operator Asset in the UE and provides a trusted execution environment (similar to an OMTP TR1 Trusted Execution Environment) and secure data storage facilities (similar to an OMTP TR1 Secure Storage facility) of its own. Since the UICC defends against most Threats considered in this document, the UICC provides a higher level of security than the defined TR1 enablers and therefore will be the operator preferred location for the most sensitive Applications and code on the UE when possible. Nevertheless, the Terminal security defined in this document is necessary for securing Terminal-centric Applications and services and also for securely facilitating (U)SIM-based Applications (e.g. when they require a user interface).

OMTP anticipates that the TR1 recommendations document can be used by all those participating in the mobile value chain as a tool to:

- Highlight technology platforms and frameworks already available to the market
- Guide Terminal requirements specifications processes
- Assist in roadmap definition

More importantly, the objective of these recommendations is also to deliver a set of requirements that operators can use to foster hardware Defragmentation and create broader alignment in the industry around commonly deployed hardware frameworks and associated software and APIs, ultimately leading to greater economies of scale to all those participating in the industry, including end-users. This approach makes particular sense when dealing with security requirements as they are frequently the aspects of a platform that require careful design attention and where interoperability and Defragmentation can deliver the maximum value without compromising overall platform differentiation, which is also necessary to motivate creativity, new features and user benefits.

OMTP has links with standards bodies and initiatives that are active in the area of mobile Terminal security. Although not all of the work of these bodies have been considered within the TR0 and TR1 recommendations, it is important to note the work of 3GPP, EICTA, ETSI Smart Card Platform, GlobalPlatform, GSMA Security Group, OMA BCAST, OMA DM, OSGi and the TCG Mobile Phone Working Group.

1.2 BUSINESS RATIONALE

In order to address the more complex use cases, enhanced security is one that stems from traditional Robustness principles (such as roots of trust and system stability) which in turn are motivated by strong business drivers.

Uniform provision of future hardware security capabilities will benefit the whole value chain:

Silicon Platform Providers

Silicon platform providers can commit to providing these capabilities knowing that there is wide industry support for them and minimal risk of the capabilities not being valued or used.

Terminal Platform Vendors

Terminal platform vendors can commit to providing capabilities defined in this recommendation knowing that there is wide industry support for them and minimal risk of capabilities not being valued or used. It helps increase roadmap confidence and helps Terminal platform vendors to align capabilities across multiple platforms in their portfolio.

Terminal Vendors

Terminal vendors are able to provide capabilities without being restricted to certain platform providers. This gives more roadmap flexibility and motivates their suppliers to differentiate features.

Terminal Platform Middleware Providers

Middleware providers can develop middleware providing and/or utilising the security capabilities knowing that there is wide industry support for them and minimal risk of the capabilities not being valued or used.

Operators

Operators can refer to or request the security capabilities described in TR1 knowing that these requirements can be supported where necessary across their Terminal portfolio. Operators can launch services that require these capabilities across much of their Terminal portfolio. Examples of such services are discussed in Part II of the document.

Users

Both consumer and corporate users can enjoy the new services thereby enabled. Economies of scale can be achieved by hardware Defragmentation. The richness of Applications enabled by these security capabilities will also benefit users with lower costs and wider choice.

1.3 ENABLERS AND CONSIDERED USE CASES

The TR1 document is comprised of a number of components. Overall, the document is divided into two parts. In Part I, the TR1 enabling technologies and their requirements are defined (hereafter called TR1 enablers). The following TR1 enablers are described:

- **Trusted Execution Environment**: an Execution Environment with security capabilities and meeting certain security-related requirements; an Execution Environment that makes certain programs executed as written, resisting a set of defined Threats.

- **Secure Storage**: is a facility that aids Trusted Execution Environment handling of Sensitive Objects, storing them as long as is required, in such a way that it maintains their security properties when resisting a set of defined Threats.

- **Flexible Secure Boot**: implements the process that ensures the Integrity of the ME code base at boot time with the possibility of modifying the code base via an update code base image which is obtained via an over-the-air download or via some other connection.

- **Generic Bootstrapping Architecture**: defines a secure GBA method for using the existing security relationship between the mobile network operator and the (U)SIM of the user for other Applications within the ME. This function is primarily used for establishment of security on the 3GPP or GSM bearer layer.

- **Run-Time Integrity Checking**: determines and maintains the Integrity of principal hardware and software components within the ME. Run-time monitoring and detection of unauthorised changes of state provides reaction capabilities against a defined set of Threats.

- **Secure Access to User Input/Output Facility**: assures secure interaction with the user, guaranteeing the secure and trusted user input and output from the Trusted Execution Environment and defending information against a defined set of Threats.

- **Secure Interaction of UICC with Mobile Equipment**: assures secure communications and interaction between the UICC and the ME, allowing the UICC to ratify the trustworthiness of the ME and securely access ME facilities. It establishes a Secure Channel between TEE and UICC Applications.

In Part II we consider example requirements for implementations of potential operator use cases using TR1 enablers.

The different use cases can independently use the different enablers and the corresponding Profiles defined in section 1.5 in order to meet the needed security requirements depending on value of content, liability models, cost of the solution and operator needs. Table 1 depicts how the different use cases can make use of different enablers.

Table 1: TR1 Enabler Profiles

Use Case Enabler	Profile	Use case 1	Use case 2	Use case 3	Use case ...	
Trusted Execution Environment	1					
	2					
Secure Storage	1					
	2					
Flexible Secure Boot	1					
	2					
Generic Bootstrapping Architecture	1					
	2					
Run-Time Integrity Checking	1					
	2					
Secure Access to User Input/Output facility	1					
Secure Interaction of UICC with Mobile Equipment	1					
	2					

Three use cases are described in Part II of the TR1 document. Other use cases may exist, but are not described in this document. The use cases in Part II of this document are:

- **Broadcast Service Protection**: the OMA BCAST Smart Card Profile protection schema
- **Trusted Device Management**: managing Trusted Environments using the OMA DM mechanism
- **Mobile Commerce**: A UICC-based proximity payment Application with user interaction

These use cases provide:

- Likely requirements should operators choose to adopt these use cases

- An indication of how the use cases can be achieved by implementing requirements from TR1

1.4 DOCUMENT STRUCTURE

In Part I, each enabler is covered in a separate chapter. For each enabler chapter, we consider the following:

- The enabler definition and introduction
- Stakeholders of the enabler (Actors)
- Key Assets of the enabler
- Requirements for the enabler

Enabler requirements are divided into core enabler requirements and enabler specific requirements. Core requirements consider the level of security required for enabler Assets (i.e. what types of Attack on enabler Assets should be resisted). Enabler specific requirements consider functional requirements for that enabler.

A wide variety of potential Attacks on mobile Devices (Threats) were gathered in ‘Security Threats on Embedded Consumer Devices’ [2], and these were used as a reference for determining required levels of security and generating Core Requirements. In order to simplify requirements, similar Threats were grouped together into generic Threat groups, which are defined in Chapter 2 Trusted Environment Threats and Best Practices: Threat Groups and Core Requirements. All Core Enabler Requirements are defined with respect to these Threat groups, and Core Requirements that are common to all enablers are also captured in section 3.2 TR1 Core Requirements.

Requirements for operator use cases are dealt with in Part II and each use case is covered in a separate chapter. The structure of these chapters is very similar to the enabler chapters in Part I.

1.5 PROFILE DEFINITIONS

Since not all Applications require the same levels of security, and since not all technologies provide the same levels of security, two different levels of security are defined within TR1 (Profile 1 and Profile 2) for each enabler. Levels of security are defined in terms of the types of Threats that must be defended against.

The two Profiles are defined as follows:

- Profile 1 enablers require defence against software Attacks, some basic hardware Attacks and some hardware probing Attacks for confidential key material. Profile 1 enablers can be used to support basic security services such as an Application Security Framework as described in the 'OMTP Application Security Framework' [3] or to manage I/O in security-demanding use cases (as for instance those defined in Part II).

- Profile 2 enablers require defence against the Attacks considered in Profile 1 as well as more sophisticated hardware Attacks such as probing on external memory. Profile 2 has to be considered from a security point of view as a superset of the Profile 1. Profile 2 enablers would typically be used to protect or provide security facilities for high value services such as m-commerce.

Most functional requirements apply to both Profiles, but certain functional requirements are only applicable to higher security enablers (Profile 2 enablers).

Unless stated otherwise, when an enabler is referenced in a TR1 requirement and the TR1 requirement is for a particular Profile, then the enabler Profile required to fulfil the requirement must be at least as high as the Profile of the requirement. For example, consider the requirement: "Asset X SHALL be protected by a Trusted Execution Environment (TEE)". If this requirement is a Profile 1 requirement, then a TEE of at least Profile 1 is required to satisfy the requirement. If the requirement is a Profile 2 requirement, then a TEE of Profile 2 is required to fulfil the requirement; a TEE of Profile 1 will not suffice in this case.

Although a Terminal is likely to integrate enablers of the same Profile, it is also possible for a Terminal to implement enablers of different Profiles. For example, a Terminal may contain a TEE of Profile 1 as well as a TEE of Profile 2.

All the requirements in this document shall be considered only with respect to the relevant Threats for the associated Profile. For example, a requirement to maintain Confidentiality of keys in a Profile 1 will be considered with respect to the Threats considered for keys in Profile 1.

The two Profiles provide flexibility for mobile operators, giving them the choice between different levels of security for different use cases. In addition, the existence of Profiles acknowledges the fact that different technologies will be available and will achieve different security levels. However by agreeing reasonable security levels at this stage, consistent choices can be provided to operators and the market can be defragmented.

1.6 UICC - SMART CARD

The UICC - Smart Card represents the security Asset trusted both by the user and the operator. The UICC currently represents the most effective way to have homogeneous security properties for a Trusted Execution Environment and Secure Storage on the whole operator Terminal portfolio.

It is a security-in-mind token, resistant to all the Attacks described in ‘Security Threats on Embedded Consumer Devices’ [2], including all hardware Attacks. It is a trusted facility for Execution and storage with very high security properties within the UE. In addition, it is also a certifiable security token. In the UICC all high security parts of the service are handled by the operator.

The TR1 enablers (in particular TEEs) will communicate and interact with the UICC in order to increase security in the access and I/O to and from UICC functionalities. TR1 requirements will complement the role of the UICC by allowing the UICC to ratify the trustworthiness of the ME and securely access ME facilities. This enables the UICC to effectively build higher levels of trust in the ME.

Although the UICC is the primary operator Asset and security resource in the UE, TR1 is dedicated to Terminal (ME) requirements and UICC requirements are out-of-scope in this document. Throughout, the terms (U)SIM, SIM, USIM and UICC are used without distinction unless it makes sense to do so.

TR1 requirements include requirements for ME interaction with the UICC. The presence of TR1 enablers can greatly support the UICC's access to the ME.

1.7 INTENDED AUDIENCE

- Operating System Vendors
- Silicon Platform Providers
- Terminal Platform Middleware Providers
- Terminal Platform Vendors
- Terminal Vendors
- Other tasks within OMTP that will take these recommendations as input

1.8 CONVENTIONS

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in RFC2119 [4].

MUST: This word, or the terms "REQUIRED" or "SHALL", mean that the definition is an absolute requirement of the specification.

MUST NOT: This phrase, or the phrase "SHALL NOT", mean that the definition is an absolute prohibition of the specification.

SHOULD: This word, or the adjective "RECOMMENDED", mean that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.

SHOULD NOT: This phrase, or the phrase "NOT RECOMMENDED" mean that there may exist valid reasons in particular circumstances when the particular behaviour is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behaviour described with this label.

MAY: This word, or the adjective "OPTIONAL", mean that an item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor feels that it enhances the product while another vendor may omit the same item. An implementation which does not include a particular option MUST be prepared to interoperate with another implementation which does include the option, though perhaps with reduced functionality. In the same vein an implementation which does include a particular option MUST be prepared to interoperate with another implementation which does not include the option (except, of course, for the feature the option provides).

The requirements within this document are uniquely identified using the following format:

ATE-XXXX-####, where:

- ATE (Advanced Trusted Environment) is the acronym identifying the subject of this OMTP document
- XXXX is a 3 to 4 uppercase letter acronym identifying the type of requirement. The acronyms used in this document are:
    - COR for Core requirements
    - TEE for Trusted Execution Environment requirements
    - SST for Secure Storage requirements
    - FSB for Flexible Secure Boot requirements
    - GBA for Generic Bootstrapping Architecture requirements
    - RIC for Runtime Integrity Checking requirements
    - SUIO for Secure access to User Input/Output facility requirements
    - SUM for Secure interaction of UICC with Mobile requirements
    - BCA for Broadcast Service protection requirements
    - TDM for Trusted Device Management requirements
    - MCO for Mobile Commerce requirements
- #### is a 4 digit number that identifies the requirement (e.g. 0020) and which is to be unique within the document.

The Assets defined within this document are uniquely identified using the following format:

- A.XXX.YYYY.# or AC.XXX.YYYY.#, where:
- A is to denote that the term is an Asset name
- AC is to denote that the term is an Asset Container name
- XXX denotes the abbreviation for the enabler or use case chapter
- YYYY denotes an abbreviation for the Asset name
- # denotes an abbreviation for the type of Asset, e.g. C for code, D for data, K for key and H for hardware

  Note: XXX.YYYY is sometimes condensed as a single abbreviation when it is the enabler itself that is considered as an Asset (e.g. A.EE.C is the Execution Environment Code).

A look-up table of Asset names can be found in Appendix 1.

PART I : TR1 CORE REQUIREMENTS AND ENABLERS

In Part I of this document we define the TR1 enabling security technologies and requirements for these enablers. Chapter 2 introduces the Threat groups that are used to define different levels of security through core requirements. More precisely, we define different security Profiles (or levels) based on the types of Threats that are defended against. Chapter 3 defines general Asset groups in TR1 and core requirements that apply to all ME Assets defined in the document.

Chapters 4 to 10 are dedicated to introducing and defining requirements for the different TR1 enablers:

- Trusted Execution Environments
- Secure Storage
- Flexible Secure Boot
- Generic Bootstrapping Architecture
- Runtime Integrity Checking
- Secure Access to User Input/Output Facility
- Secure Interaction of UICC with Mobile Equipment

2 TRUSTED ENVIRONMENT THREATS AND BEST PRACTICES

2.1 THREAT GROUP DEFINITIONS

In order to describe what security level an implementation of this recommendation must have, this recommendation will describe how a particular Asset must be protected against particular Threats.

The following Threat groups are defined and will be used throughout this document. The Threat groups are based on the document 'Security Threats on Embedded Consumer Devices' [2] where more background information is available.

2.1.1 DEFINITION OF THREAT GROUP 1 - HARDWARE MODULES USED FOR ACCESSING MEMORIES

A DMA (Direct Memory Access) module is an example of a hardware module which has "bus master" rights. These rights allow the device to move data independently of the CPU(s) managing the Assets. As such it is not blocked by MMU (Memory Management Unit) level defences and the user/privileged split.

The DMA module might use its capabilities to illegally move data between memory areas, i.e. to read and/or write data to or from memory areas which the entity controlling the DMA is not authorised to access. This leads to unauthorised leakage and/or manipulation of data.

Other "bus master" modules, such as other CPUs, CLCD controllers, Ethernet controllers, or even camera control blocks, may be used to perform similar Attacks.

This Threat group only includes Attacks mounted via modules that are built into the ME design, and not additional modules added by an Attacker.

More information can be found in Threat T.SWO.008 in 'Security Threats on Embedded Consumer Devices' [2].

2.1.2 DEFINITION OF THREAT GROUP 2 – CLCD USED FOR DISPLAYING MEMORIES AND INTERFERING WITH DISPLAYED DATA

This Threat group is similar to Threat group 1 – Hardware Modules Used for Accessing Memories. It is, however, a separate group because the information generated is immediately available outside the device and therefore does not require as much in the way of supporting software.

The CLCD (Colour LCD controller) is the graphics chip in a mobile Device. CLCD controllers are designed to be pointed at memory blocks, which normally contain graphics data. This usually bypasses the CPU's MMU (Memory Management Unit) and any privileges/user protection it may offer.

For games, graphics chips typically have to be pointed at different memory banks by the game code. If no appropriate hardware or software based protection exists then a 'game' can be used to display secrets on the screen. Interpreting the screen may be difficult as it is just 'raw' memory, but it is possible. This can be considered an easier Attack to perform, as graphics processors tend to allow this sort of capability through programmer-friendly APIs. However the translation of the data may be considered more difficult since a moderate knowledge of graphic display formatting is required.

More information can be found in Threat T.SWO.010 in ‘Security Threats on Embedded Consumer Devices’ [2].

2.1.3 DEFINITION OF THREAT GROUP 3 – BYPASS SECURITY BY REMOVAL OF BATTERY POWER OR REMOVAL OF EXTERNAL MEMORY CARD

In the field, batteries are the only power source on a mobile Device. Depending on Device architecture a Device may have one or more batteries such as a main battery that can be changed by the end user and a back-up battery. Removing the power supplied by any of the batteries during a sensitive operation may leave the UE in an unknown state. For example, in a DRM situation, a song may have been played, but if the rights count is not decremented until the end of the song, then removing the power source could result in the user playing the song indefinitely. In addition, by removing the power source for a sufficient amount of time, any internal clock may halt, and some long term memory may be corrupted. Similarly, if an external memory card is used to hold rights to media, removing the card prior to rights manipulation may allow infinite plays.

More information can be found in Threats T.HWE.004 and T.HWE.005 in ‘Security Threats on Embedded Consumer Devices’ [2].

2.1.4 DEFINITION OF THREAT GROUP 4 - ATTACK BY REPLACEMENT OF FLASH WHEN POWER IS OFF (PRE-BOOT)

Even if all the routes for changing Flash contents are secured - through hacking 'defined' APIs such as JTAG, serial download or hacking software with its own Flash drivers, then it is still always possible to bypass all these by physically replacing the Flash with one containing hacked code.

More information can be found in Threat T.CLO.005 in ‘Security Threats on Embedded Consumer Devices’ [2].

2.1.5 DEFINITION OF THREAT GROUP 5 - EXTRACT SECRET VIA BUS MONITORING (HARDWARE PROBES)

Any data that travels off-SoC e.g. to DRAM or Flash, is vulnerable to being stolen by applying a monitoring probe to the physical bus on the Printed Circuit Board (PCB) that it travels down.

Off-SoC monitoring may be performed without leaving a trace. Some security criteria do not expect the blocking of such Attacks, but do require that there should be some physical evidence left by such tampering.

More information can be found in Threat T.HWT.001 in ‘Security Threats on Embedded Consumer Devices’ [2].

2.1.6 DEFINITION OF THREAT GROUP 6 – MOD CHIP ATTACKS ON DATA IN EXTERNAL RAM

Once data is loaded from Flash to RAM for Execution, it is typically considered only vulnerable to software Attacks such as software bug exploits.

Some Attackers have already used attached devices in order to execute this type of Attack against mobile Terminals and the link between the Terminal and UICC. This is more likely to be attempted when other measures described in this document are implemented, shutting down more accessible and cost-effective Attack vectors. The sort of Mod Chip used to tamper with games consoles could become widely used against mobile phones. These typically consist of a microcontroller with limited ROM containing Attack code. They can take their power from the Device, and are used to interfere with data on the bus between the SoC and the DRAM.

Mod Chips might also be used to Attack data travelling from Flash to SoC with Flashes that may be secure to normal programming/replacement Attacks.

Another Attack that should be considered is one where the Mod Chip intercepts and changes varying code and data being written from SoC to random locations in RAM.

More information can be found in Threats T.CLO.003, and T.CLO.004 in ‘Security Threats on Embedded Consumer Devices’ [2].

2.1.7 DEFINITION OF THREAT GROUP 7 - ATTACK BY REPLACEMENT OF FLASH WHEN POWER IS ON (POST-BOOT)

The post-boot flash substitution requires a little more effort by the Threat Agent than Threat Group 4 - Attack by Replacement of Flash When Power is Off (Pre-Boot).

An example of a method that might be deployed is that of replacing the original Flash using a double size Flash device, with the same physical package outline as the original, allowing it to be placed on the PCB without difficulty. It is then possible to boot from the normal address space with the usual static memory Secure Boot Integrity checks occurring. Then, at some time after completion of the boot process, the address space can be remapped to the upper half of the Flash simply by forcing the state of the highest address pin exposing the system to Execution of unsigned or unchecked code. The Attacking changes will take effect as soon as those code and data blocks are copied from Flash to RAM, for example by a paged memory system.

More information can be found in Threat T.CLO.006 in ‘Security Threats on Embedded Consumer Devices’ [2].

2.2 Best Practices

2.2.1 Definition of Software Quality Measures

Software quality measures can be deployed during the software development phase to improve the quality of the software and therefore reduce the number of bugs in the software. The security problems that may arise from bugs in software are described in T.SWO.001, T.SWO.006, and T.SWO.014 in 'Security Threats on Embedded Consumer Devices' [2]. The following is a non-exhaustive list of software quality measures. Vendors must deploy software quality measures when developing security critical code although they are not required to implement the exact measures described in the list:

- Intensify testing, verification or reviewing of security critical code
- Deploy coding standards and documentation methodologies that lead to best practice
- Perform external validation or analysis of security critical code
- Reduce the size of the security critical code
- Use developers who are aware of security and quality development issues
- Provide Execution Environments that guard against flawed code by restricting the developer (e.g. reduced instruction set, controlled data import and export options)

2.2.2 Definition of API Related Coding Techniques

API-related coding techniques are methods that can be deployed during the software development phase to prevent typical security problems that arise due to poorly designed APIs. These security problems are described in T.SWO.002, T.SWO.003, T.SWO.004, T.SWO.009, T.SWO.011, T.SWO.015, and T.SWO.016 in 'Security Threats on Embedded Consumer Devices' [2]. The following is a non-exhaustive list of API-related coding techniques. Vendors must use API-related coding techniques when developing security critical code, although they are not required to implement the exact measures described in the list:

- Separate security API and general API domains
- Isolate security critical code from the general code base, so that the boundary that must be crossed to enter that code is not the same one as that being crossed by general code API usage
- Use Type-Safe APIs so malformed data cannot be sent through them. For example, the inbuilt structural checking of the command data tends to reduce any effects of erroneous events
- Do not mix Type-Safe and non-Type-Safe APIs across the same API domain boundary

- Design internal APIs to consider information hiding and access restriction wherever possible
- Application IDs should only be communicated to those components on the ME that require them

2.2.3 *DEFINITION OF BUFFER OVERFLOW PROTECTION MECHANISMS*

Buffer Overflow protection mechanisms are mechanisms that prevent buffer overflows from leading to security breaches such as the ones described in T.SWO.005 in 'Security Threats on Embedded Consumer Devices' [2]. The following is a non-exhaustive list of Buffer Overflow protection mechanisms. Vendors must use Buffer Overflow protection mechanisms for security critical code although they are not required to implement the exact measures described in the list:

- Use a Type-Safe API; when all data has specific size information, Buffer Overflows are prevented by tracking buffer size. Such tracking has a notable speed overhead
- Use a programming language that does not suffer from Buffer Overflow problems
- Use verification tools to find Buffer Overflows in source code before the code is released
- Prevent Execution of memory used for the stack and the heap - e.g. by having hardware that supports marking parts of memory as non-executable
- Do not make internal use of buffers, stacks, or dynamically growing data structures

2.2.4 *DEFINITION OF EXECUTION ISOLATION TECHNIQUES TO ADDRESS SOFTWARE ATTACKS*

Execution isolation techniques such as provided by system virtualisation technology, operating system structuring (e.g. microkernel) and/or hardware isolation are methods that can be deployed during the software development phase to architect the software in such a way that potential software vulnerabilities are confined inside isolated Execution Environments.

This allows to protect the security properties of sensitive Assets against software Threats described in 'Security Threats on Embedded Consumer Devices' [2] and in particular those exploiting faulty or illegal privileged code extensions such as described in SWM.001, SWM002, SWM.003 and SWM.004, software bugs such as described in T.SWO.001, T.SWO.006 and T.SWO014, API related vulnerabilities such as described in T.SWO.002, T.SWO.003, T.SWO.004, T.SWO.009, T.SWO.011, T.SWO.0012, T.SWO.015 and T.SWO016, Buffer Overflows such as described in T.SWO.005, and abuse of concurrent processes such as described in T.SWO.0013.

Vendors should consider using such techniques to isolate security critical code from general purpose code in separate Execution Environments. These techniques can be provided by a combination of hardware and software protection mechanisms; e.g. Virtual Machine monitors, hypervisors, UICCs with Java card support or separate chips that have a controller inside.

2.2.5 DEFINITION OF CONCURRENT PROCESSING THREAT

Malicious software could exploit scheduling concurrence to let a task access the resources (e.g. where an operating system dynamically schedules processes that use common resources, by exploiting a race condition) belonging to another task. This security problem is described in T.SWO.013.

Suggested Solutions:

- Avoid any concurrence (i.e. with a single thread model such as in the GlobalPlatform GPD/STIP Environment [5])
- Stay with a simple thread interaction model in any area that is dealing with security, and isolate that area cleanly from other more complex processing models
- If using multi-threaded model, use an approach that allows the demonstration of Robustness against the above Threat

3 TRUSTED ENVIRONMENT CORE REQUIREMENTS

3.1 TRUSTED ENVIRONMENT ASSET GROUPING

Assets for each enabler and use case are defined in the corresponding enabler and use case chapters, and are marked as Security Sensitive or Operational (not Security Sensitive).

The requirements in the following sections (section 3.2) will apply to all sensitive ME Assets defined in TR1 for all the enablers, so we group the TR1 ME Assets into the following Asset Groups:

- **TR1 Code Assets** group contains all Security Sensitive ME Code Assets defined in the TR1 document.
- **TR1 Data Assets** group contains all Security Sensitive ME Data Assets defined in the TR1 document.
- **TR1 Hardware Assets** group contains all Security Sensitive ME Hardware Assets defined in the TR1 document.
- **TR1 Key Assets** group contains all Security Sensitive ME Key Assets defined in the TR1 document.

Note that ME Assets marked as operational are not included in these Asset Groups. Furthermore, note that UICC Assets are not included in these Asset Groups because the requirements in this recommendation only apply to the ME.

Before describing the TR1 requirements, it is worth noting that requirements on key Assets in the TR1 document have implications on code handling these keys. For example, if a key requires Confidentiality, then not only does the key require protection during storage and manipulation, but the mechanisms ensuring the key's Confidentiality need to themselves be Integrity protected to ensure that the protection mechanisms cannot be compromised. This needs to be considered throughout the TR1 document.

Within the document there are a number of Asset tables (for example, Table 4 in section 4.3) which list the Assets referred to in the normative requirements in the section in which the Asset table resides. In all of these Assets tables there is a column "Security Properties Required" which lists, for each Asset in the table, the security properties of that Asset which must be protected. This column is significant and normative, in that there are normative requirements in section 3.2 related to *all* Assets listed in this document which require the protection of these security properties against various Threats. In addition, there are many requirements in the document which state more specific normative requirements for the protection of the security properties of various Assets.

3.2 TR1 Core Requirements

3.2.1 Requirements to Protect Against Software Threats

REQ. ID	**REQUIREMENT**	**PROFILE 1**	**PROFILE 2**
ATE-COR-0010	The software development process used for TR1 Code Assets SHALL make use of Software Quality Measures as defined in section 2.2.1 “Definition of Software Quality Measures”	✓	✓
ATE-COR-0020	All TR1 Code Assets SHALL use API related Coding Techniques as defined in section 2.2.2 ”Definition of API Related Coding Techniques”	✓	✓
ATE-COR-0030	All TR1 Code Assets SHALL use Buffer Overflow Protection Mechanisms as defined in section 2.2.3 “Definition of Buffer Overflow Protection Mechanisms”	✓	✓
ATE-COR-0040	The security properties of all TR1 Code Assets SHOULD be protected against abuse of concurrent processes by using methods described in section 2.2.5 “Definition of Concurrent Processing Threat”		✓
ATE-COR-0050	The security properties of all TR1 Code, Data and Key Assets SHALL be defended against the Threats contained in Threat Group 1 “Hardware Modules Used for Accessing Memories”		✓
ATE-COR-0060	The security properties of all TR1 Assets requiring Confidentiality SHALL be defended against the Threats contained in Threat Group 2 “CLCD Used for Displaying Memories and Interfering With Displayed Data”		✓

REQ. ID	REQUIREMENT	PROFILE 1	PROFILE 2
ATE-COR-0070	Isolation techniques as defined in section 2.2.4 Definition of Execution Isolation Techniques to Address Software Attacks SHOULD be used to protect all TR1 Code Assets against software Attacks as described in 'Security Threats on Embedded Consumer Devices' [2]		✓

3.2.2 REQUIREMENTS TO PROTECT AGAINST HARDWARE THREATS

REQ. ID	REQUIREMENT	PROFILE 1	PROFILE 2
ATE-COR-0080	The security properties of all TR1 Code, Data and Key Assets SHALL be defended against the Threats contained in Threat Group 3 "Bypass Security by Removal of Battery Power or Removal of External Memory Card"	✓	✓
ATE-COR-0090	All TR1 Code, Data and Key Assets SHALL be defended against the Threats contained in Threat Group 4 "Attack by Replacement of Flash When Power is Off (Pre-Boot)"	✓	✓
ATE-COR-0100	TR1 Assets requiring Confidentiality SHALL be defended against the Threats contained in Threat Group 5 "Extract Secret via Bus Monitoring (Hardware Probes)"		✓
ATE-COR-0110	The security properties of TR1 Key Assets SHALL be defended against the Threats contained in Threat Group 6 "Mod Chip Attacks on Data in External RAM"		✓
ATE-COR-0120	The security properties of TR1 Key Assets SHALL be defended against the Threats contained in Threat Group 7 "Attack by Replacement of Flash When Power is On (Post-Boot)"		✓

3.2.3 DEBUG REQUIREMENTS

Unauthorised access to debug facilities on a mobile Device remains a significant security Threat within an ME. Requirements preventing unauthorised access to debug ports were stated in the 'OMTP Trusted Environment; OMTP TR0' [1].

This recommendation adopts these requirements and uses the same definition of debug ports as given in the 'OMTP Trusted Environment: OMTP TR0' [1]. The Assets that implement TR0 compliant debug ports will be referred to as the debug mechanism in this recommendation.

The requirements below describe which Threats debug mechanisms shall be defended against in the different Profiles.

REQ. ID	**REQUIREMENT**	**PROFILE 1**	**PROFILE 2**
ATE-COR-0130	If the debug mechanism is implemented in software then its code SHALL comply with the requirements for TR1 Code Assets in ATE-COR-0010, ATE-COR-0020, and ATE-COR-0030	✓	✓
ATE-COR-0140	The debug mechanism SHALL be defended against the Threats contained in Threat Group 1 "Hardware Modules Used for Accessing Memories"		✓
ATE-COR-0150	The debug mechanism SHALL be defended against the Threats contained in Threat Group 2 "CLCD Used for Displaying Memories and Interfering With Displayed Data"		✓
ATE-COR-0160	The debug mechanism SHALL be defended against the Threats contained in Threat Group 3 "Bypass Security by Removal of Battery Power or Removal of External Memory Card"	✓	✓
ATE-COR-0170	The debug mechanism SHALL be defended against the Threats contained in Threat Group 4 "Attack by Replacement of Flash When Power is Off (Pre-Boot)"	✓	✓

REQ. ID	REQUIREMENT	PROFILE 1	PROFILE 2
ATE-COR-0180	The debug mechanism SHALL be defended against the Threats contained in Threat Group 5 “Extract Secret via Bus Monitoring (Hardware Probes)”		✓
ATE-COR-0190	The debug mechanism SHALL be defended against the Threats contained in Threat Group 6 “Mod Chip Attacks on Data in External RAM”		✓
ATE-COR-0200	The debug mechanism SHALL be defended against the Threats contained in Threat Group 7 “Attack by Replacement of Flash When Power is On (Post-Boot)”		✓

A debug mechanism will be referred to as a Profile 1 debug mechanism if it fulfils all of the requirements for Profile 1 above in ATE-COR-0130, ATE-COR-0160 and ATE-COR-0170. Correspondingly, a debug mechanism that fulfils all the requirements for Profile 2 will be referred to as a Profile 2 debug mechanism.

Table 2 summarises how the requirements above apply to the debug mechanism. Each cell contains the number of the Profile for which the debug mechanism has to be protected against a Threat group.

Table 2: Debug Mechanism Threat Grouping

Threat Group	1	2	3	4	5	6	7
Threats Assets	Hardware Modules Used for Accessing Memories	CLCD Used for Displaying Memories and Interfering With Displayed Data	Bypass Security by Removal of Battery Power or Removal of External Memory Card	Attack by Replacement of Flash When Power is Off (Pre-Boot)	Extract Secret via Bus Monitoring (Hardware Probes)	Mod Chip Attacks on Data in External RAM	Attack by Replacement of Flash When Power is On (Post-Boot)
Debug Mechanism	2	2	1, 2	1, 2	2	2	2

The requirements for debug in the ‘OMTP Trusted Environment: OMTP TR0’ [1] shall be considered only with respect to the relevant Threats for the associated Profile. As an example, consider a Profile 1 debug mechanism that includes an authentication mechanism, which prevents access to a debug port without authentication. Then the debug mechanism shall prevent access under Attack from the Threats defined in Profile 1 above. However, there are no requirements that the debug mechanism shall prevent access under other Threats.

The following requirements make the link between the debug requirements in TR0 and the requirements to Profile 1 and Profile 2 debug mechanisms given above.

REQ. ID	REQUIREMENT	PROFILE 1	PROFILE 2
ATE-COR-0210	Debug mechanisms SHALL comply with requirements DP 1, DP 2, DP 3, DP 4, and DP 5 of 'OMTP Trusted Environment: OMTP TR0' [1]	✓	✓
ATE-COR-0220	If a debug mechanism can access an Asset, the security properties of which have to be assured by requirements listed in Profile 1, then the debug mechanism SHALL be at least of Profile 1	✓	
ATE-COR-0230	If a debug mechanism can access an Asset, the security properties of which have to be assured by requirements listed in Profile 2, then the debug mechanism SHALL be at least of Profile 2		✓

3.2.4 CRYPTOGRAPHIC REQUIREMENTS

The following requirements define what level of cryptographic support is required for TR1 Assets. These requirements are based on recommendations for cryptographic algorithms and their use from NIST [6, 7, 8], the Austrian A-SIT [9], the German Federal Network Agency [10], and the French DCSSI [11].

REQ. ID	REQUIREMENT	PROFILE 1	PROFILE 2
ATE-COR-0240	For data that is cryptographically protected for Confidentiality with a symmetric cryptosystem, the UE SHALL support a cryptosystem that provides the mathematical security strength at least equivalent to the AES algorithm using a 128-bit symmetric key	✓	✓
ATE-COR-0250	For data that is cryptographically protected for Confidentiality with an asymmetric cryptosystem, the UE SHALL support a cryptosystem that provides the mathematical security strength at least equivalent to the RSA algorithm using a 2048-bit modulus	✓	✓

REQ. ID	REQUIREMENT	PROFILE 1	PROFILE 2
ATE-COR-0260	For data that is cryptographically protected for Authenticity, the UE SHALL support a cryptosystem that provides the mathematical security strength at least equivalent to one of the following: • An RSA signature algorithm using a 2048-bit modulus and a SHA-256 message digest OR • an ECC signature algorithm using a 256-bit prime order subgroup and a SHA-256 message digest OR • an HMAC using SHA-256 and a 128-bit symmetric key Note: SHA-1 may be used instead of SHA-256 where necessary for interoperability with existing technology - provided the manner of usage protects against attacks based on pre-computed collisions	✓	✓
ATE-COR-0270	For data that is cryptographically protected for Integrity, the Terminal SHALL support a cryptosystem such that the number of operations needed to find a second pre-image with the same digest as the protected data SHOULD be at least equivalent of Order of (2^128)	✓	✓

4 TRUSTED EXECUTION ENVIRONMENTS

This chapter describes the security requirements that pertain to Trusted Execution Environments, for the two different Profiles.

An introduction to Execution Environments and Trusted Execution Environments is provided in section 4.1.

The list of Actors and Actors' concerns is provided in section 4.2.

The list of Trusted Execution Environment Assets is defined in section 4.3.

The security requirements are defined in section 4.4.

4.1 INTRODUCTION TO TRUSTED EXECUTION ENVIRONMENTS

4.1.1 EXECUTION ENVIRONMENT DEFINITION

From a functional point of view, an Execution Environment (EE) is a set of hardware and software components providing facilities (e.g. Computing, Memory Management, input/output, etc.), necessary to support Applications.

4.1.2 EXECUTION ENVIRONMENT CORE COMPONENTS

An Execution Environment typically consists of the following elements:

- A processing unit (AC.EE.PU H) that:
    - initialises the EE during boot
    - defines the EE Instruction Set Architecture (ISA)
    - is used for executing code built using this set of instructions
- A set of connections (e.g. buses) between the processing unit and code/instructions, data, resources (I/O)
- Physical Memory for storing data and code
- A mechanism to initialise the boot process
- The code, data and keys of the EE itself (A.EE.C/D/K)

4.1.3 EXECUTION ENVIRONMENT FACILITIES

An EE offers facilities to its Applications. Such facilities are available through an interface often referred to as either an Application Programming Interface (APIs) or an Instruction Set Architecture (ISA), depending on the type and implementation of the interface.

An EE may offer some or all of the following facilities (the code, data and keys used to implement the following facilities is all contained within Assets A.EE.C/D/K respectively):

- Execution Management: Some EEs may provide facilities to dynamically manage the Execution of code in the EE (e.g. thread Execution, interrupt handling, scheduling, synchronisation mechanisms, Application Loading/unloading) and whether the code is executed natively or interpreted (e.g. byte code). Execution management may provide several Execution options (e.g. normal, debug, monitor).

- Memory Management: Some EEs may provide facilities to allocate/deallocate, map/unmap, protect/unprotect, and lock/unlock memory areas, either from anonymous main memory or from specific physical areas to access specific I/O and/or communication resources.
- Storage & File Management: Some EEs may provide facilities to enable Applications to store and retrieve data in non-volatile storage. Such storage facilities could also be used to store Applications themselves, to avoid them being resident in main memory.
- Intra-EE Communications: Some EEs may provide facilities to enable Applications running within the EE to communicate between themselves. An operating system would typically provide IPC (Inter-Process Communication) mechanisms to its Applications (processes).
- Application Management: Some EEs may provide facilities to configure, install, Upgrade, and manage Applications.
- Network & Communication: Some EEs may provide facilities to enable Applications to establish connections with the external world through whatever protocol stack is relevant to the Applications and the EE.
- Input/Output: Some EEs may provide facilities to access I/O devices either in some raw form or through more elaborate facilities. Examples of such I/O devices could be audio, graphics, keyboard, keypad and UART. These can be divided into user interface mechanisms and peripherals.
- Credential Management: Some EEs may provide facilities to identify users and processes, and assign them credentials and permissions so that Access Control policies may be enforced.

Facilities may vary from one EE to another. Similar facilities may be offered in different ways (different abstractions and APIs) by different EEs. EEs however, may implement all or a subset of the described individual facilities.

4.1.4 EXECUTION ENVIRONMENT CONFIGURATION MANAGEMENT

More than one EE can be present on the same platform. In this case, a resource can be:

- dedicated: the resource is dedicated to a particular EE
- shared: the resource is shared between multiple EEs

Some resources share Assets with one or multiple EEs: for example they may share the access to a given bus. These resources are an extension of the EEs, and must be considered together as a whole.

Resources can themselves be composed of EEs: e.g. micro-programmable DMA or multi-core hardware architectures.

When multiple EEs are present on the same platform there must be ways to configure the set of resources that are owned by an EE as well as the set of resources shared with other EEs. Such configurations may be pre-defined and can be static or dynamic.

Resources granted to an EE should be protected from Attacks from other EEs.

An EE may offer the following facilities (the code, data and keys used to implement the following facilities is all contained within Assets A.EE.C/D/K respectively):

- Configuration Management: Following defined security policies, this facility permits the ability to dynamically:
    - Create/destroy/monitor/debug an EE
    - Allocate a set of dedicated or shared resources available to that EE either at start time or during run-time
    - De-allocate dedicated or shared resources previously allocated to an EE. Note that upon destruction of an EE all resources previously allocated to that EE are de-allocated (recycled)
    - Switch resources to simplify dynamic management of resources so that they can be easily de-allocated from an EE and allocated to another one, through a single “switch” operation
    - Account for EE resource usage
    - Manage (define, delete, modify) the Security Policy applicable to the set of resources owned by an EE
- Inter-EE Communications: EEs may offer to their Applications a facility to establish communication channels. Such channels can be used to exchange data between Applications running in different EEs. Communication channels may also be destroyed. Inter-EE communications channels could include the use of shared resources such as memories.

This document assumes that certain authorised EEs have access to the Hardware Unique Key (A.EE.HUK.K) as defined in the ‘OMTP Trusted Environment: OMTP TR0’ [1].

4.1.5 *EXAMPLES OF EXECUTION ENVIRONMENTS*

Applications running in an EE may be either delivering a service to a user, or may themselves constitute another EE. On the other hand, an EE may be as simple as a tiny kernel providing basic elementary services to Applications or even, in the extreme, the EE may be limited to the bare hardware platform.

Here are some various examples of EEs:

- a Central Processing Unit (CPU)
- an Operating System (OS), running on top of a CPU
- an OS, running on top of virtualisation software, running on top of a CPU
- a Virtual Machine, running on top of an OS, running on top of a CPU
- an OS running on top of a Universal Integrated Circuit Card (UICC) CPU
- a Virtual Machine, running on top of an OS, running on top of a UICC

4.1.6 TRUSTED EXECUTION ENVIRONMENT DEFINITION

This document considers two sets of EE security requirements which are called Profiles 1 and 2, and any EE that meets these requirements is referred to as a Trusted Execution Environment (TEE). More specifically, an EE that meets the Profile 1 (or 2) requirements is referred to as a Profile 1 (or 2) TEE.

Each TEE Profile's requirements define certain required functionalities of the TEE and also the scope of the Attacks to which TEE Assets should be resistant. The two Profiles offer incremental levels of security (that is, Profile 2 offers stronger security than Profile 1). Both Profiles protect security property of the code Execution from basic identified Threats. Certain Applications may rely on the UICC to run the most sensitive code for trusted services.

4.1.7 OPEN TRUSTED EXECUTION ENVIRONMENT

If a TEE allows Applications to be installed (pre-sale or OTA) by the operator (and potentially other parties), we refer to this type of TEE as an "Open TEE". Analogously, we refer to Applications that can be installed in an Open TEE as Open TEE Applications.

4.2 TRUSTED EXECUTION ENVIRONMENT ACTORS AND THEIR CONCERNS

Table 3 describes all Actors involved in the Trusted Execution Environment, their main concerns, and the trust link between them.

Table 3: Trusted Execution Environment Actors and Their Concerns

Actor	Main Concerns (in this context)
Operators (O)	Operators wish to be able to download and run Applications and services from various TEEs. Operators will trust the UICC manufacturer and/or Terminal manufacturer and will want some control over the access other service providers are granted to TEEs. They would require cooperation between the UICC and the ME to deploy complex services
Other Service Providers (SP)	Service providers will want to deploy secure services using TEEs and will trust the UICC manufacturer and/or Terminal manufacturer, although not necessarily the operator
Content Providers (CP)	Content providers will want any sensitive data residing in a TEE to be protected from malicious Attacks in that TEE and other EEs, and will trust the UICC manufacturer, Terminal manufacturer and/or the operator
End User (U)	The end user will want seamless transition between EEs, and will expect the operator, UICC manufacturer and Terminal manufacturer to protect user Assets
Terminal Manufacturer (M)	The Terminal manufacturer implements TEEs to be available on the Terminal, and provides mechanisms to manage these TEEs. Manufacturers will install Applications to run in these TEEs, and will want to be able to install/Upgrade software running in theses TEEs

Actor	Main Concerns (in this context)
UICC Manufacturer (UM)	The UICC manufacturer implements TEEs to be available on the UICC, and provides mechanisms to manage these TEEs. Manufacturers will install Applications to run in these TEEs and will want to be able to install/Upgrade software running in theses TEEs
Attacker (A)	An Attacker may want to gain access to a TEE, control the management or settings of a TEE, or undermine the separation between EEs in order to access Assets on the Terminal and/or the UICC
Removable Device Manufacturer (RDM)	The Removable Device Manufacturer implements removable peripheral devices that could be attached or detached to specific TEEs by the user

4.3 TRUSTED EXECUTION ENVIRONMENT ASSETS

4.3.1 TRUSTED EXECUTION ENVIRONMENT ASSETS TABLE

Table 4 lists Assets in the implementation of a TEE that require protection. The Assets table describes various properties of TEE Assets that were used to derive the TEE requirements in section 4.4.

Table 4: Trusted Execution Environment Assets

Asset (A) or Asset Container (AC)	Description	Type	Owner†	Protection Level/Level of Sensitivity	Security Properties Required
A.EE.xxx or AC.EE.xxx		*Key/Data/ Code/Hardware*	*See Actors List*	*Sensitive/Operational*	*Integrity (I)* *Confidentiality (C)* *Authenticity (Au)* *Non-Replay (NR)*
A.EE.HUK.K	Hardware Unique Key	Keys	M or UM	S	I, C
A.EE.C	Execution Environment Code	Code	M, RDM or UM	S	I, Au

Asset (A) or Asset Container (AC)	Description	Type	Owner†	Protection Level/Level of Sensitivity	Security Properties Required
A.EE.D	Execution Environment Data (multiple Application running contexts: registers, etc.)	Data	M, RDM or UM	S	I, Au, NR
A.EE.K	Execution Environment Keys	Keys	M, RDM or UM	S	I, C**, Au**
A.EE.APP.C	Applications Code	Code (Data for the EE)	SP or U	O/S**	I, C**,Au**
A.EE.APP.D	Applications Data	Data	M or RDM or UM or O or SP or CP or U	O/S**	I**,C**,Au**, NR**
A.EE.APP.K	Applications Keys	Keys	M or RDM or UM or O	O/S**	I,C**,Au**

Notes:

† Denotes that the given set of owners of a specific Asset is only indicative

** Denotes the fact that the required security properties of these Assets depends on the requirements of the actual Application and the Application owner

4.3.2 Trusted Execution Environment Asset Grouping

In order to facilitate requirement understanding, the following groups of Assets are defined:

- **TEE Code Assets** group contains all TEE code Assets: A.EE.C, A.EE.APP.C
- **TEE Data Assets** group contains all TEE data Assets: A.EE.D, A.EE.APP.D
- **TEE Key Assets** group contains all TEE key Assets: A.EE.HUK.K, A.EE.K, A.EE.APP.K

4.4 Trusted Execution Environment Requirements

We now define the requirements for Profile 1 and 2 TEEs.

The Trusted Execution Environment Core Requirements in section 4.4.1 stipulate the Threats that must be defended against for each Asset in the different Profiles. Section 4.4.2 describes requirements for the use of cryptography and the handling of cryptographic keys. Section 4.4.3 describes the security facilities that a TEE shall provide to Applications running within it, and sections 4.4.4 and 4.4.5 describe requirements for Applications and Application lifecycle management. Sections 4.4.6 and 4.4.7 deal with requirements for TEE isolation and self-protection, while section 4.4.8 describes requirements for the protection of inter-EE communications.

4.4.1 TRUSTED EXECUTION ENVIRONMENT CORE REQUIREMENTS

The Asset security properties referred to in the following requirements are defined in section 4.3.1.

In addition to TR1 Core Requirements defined in Chapter 3 “Trusted Environment Core requirements”, further core requirements are defined for TEE in this chapter.

REQ. ID	REQUIREMENT	PROFILE 1	PROFILE 2
ATE-TEE-0280	The Confidentiality of the Hardware Unique Key (A.EE.HUK.K) SHALL be defended against the Threats contained in Threat Group 5 “Extract Secret via Bus Monitoring (Hardware Probes)”	✓	✓
ATE-TEE-0290	The security properties of the Hardware Unique Key (A.EE.HUK.K) SHALL be defended against the Threats contained in Threat Group 6 “Mod Chip Attacks on Data in External RAM”	✓	✓
ATE-TEE-0300	The security properties of the Hardware Unique Key (A.EE.HUK.K) SHALL be defended against the Threats contained in Threat Group 7 “Attack by Replacement of Flash When Power is On (Post-Boot)”	✓	✓
ATE-TEE-0310	The security properties of the Hardware Unique Key (A.EE.HUK.K) SHALL be defended against the Threats contained in Threat Group 1 “Hardware Modules Used for Accessing Memories”	✓	✓

REQ. ID	REQUIREMENT	PROFILE 1	PROFILE 2
ATE-TEE-0320	The security properties of the Hardware Unique Key (A.EE.HUK.K) SHALL be defended against the Threats contained in Threat Group 2 “CLCD Used for Displaying Memories and Interfering With Displayed Data”	✓	✓

Table 5 summarises the TEE core requirement table that apply to TEE Assets.

Each cell contains the Profiles for which an Asset group have to be protected against a Threat group.

Table 5: Trusted Execution Environment Threat Grouping

Threat Group	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Threats Assets	Hardware Modules Used for Accessing Memories	CLCD Used for Displaying Memories and Interfering With Displayed Data	Bypass Security by Removal of Battery Power or Removal of External Memory Card	Attack by Replacement of Flash When Power is Off (Pre-Boot)	Extract Secret via Bus Monitoring (Hardware Probes)	Mod Chip Attacks on Data in External RAM	Attack by Replacement of Flash When Power is On (Post-Boot).
A.EE.C A.EE.APP.C	2	2 (C)*	1, 2	1, 2	2 (C)*	-	-
A.EE.D A.EE.APP.D	2	2 (C)*	1, 2	1, 2	2 (C)*	-	-
A.EE.HUK.K	1, 2	1, 2 (C)	1, 2	1, 2	1, 2 (C)	1, 2	1, 2
A.EE.K A.EE.APP.K	2	2 (C)	1, 2	1, 2	2 (C)	2	2

Notes:

(C) Denotes the fact that only Confidentiality of these Assets needs to be defended against this Threat group

(C)* Denotes the fact that these Assets may not require Confidentiality protection, but if they do, then this Threat group must be defended against.

4.4.2 REQUIREMENTS FOR TRUSTED EXECUTION ENVIRONMENT KEYS

REQ. ID	REQUIREMENT	PROFILE 1	PROFILE 2
ATE-TEE-0330	The provisioning of any Private Keys or Secret Keys used by the TEE SHALL protect the Confidentiality and Integrity of the value of these keys	✓	✓
ATE-TEE-0340	The provisioning and storage of any Root Public Keys used by the TEE SHALL protect the Integrity and Authenticity of these keys	✓	✓
ATE-TEE-0350	The Integrity and Authenticity of Public Keys used by the TEE SHALL be verified before use unless they are stored in Integrity-Protected Memory	✓	✓
ATE-TEE-0360	The Integrity of Private Keys and Secret Keys used by the TEE SHALL be verified before use unless they are stored in Integrity-Protected Memory		✓
ATE-TEE-0370	The storage and manipulation of Private Keys and Secret Keys used by the TEE SHALL protect the Confidentiality of these keys	✓	✓
ATE-TEE-0380	The Confidentiality of Private Keys and Secret Keys used by the TEE SHALL be maintained until they are erased	✓	✓
ATE-TEE-0390	Any TEE code that manipulates cryptographic keys SHALL be Integrity protected OR the Integrity of this code SHALL be verified before being granted access to keys		✓

4.4.3 REQUIREMENTS FOR FACILITIES

REQ. ID	REQUIREMENT	PROFILE 1	PROFILE 2
ATE-TEE-0400	The TEE SHALL provide Applications with access to a Secure Storage facility, at least compliant with Profile 1	✓	
ATE-TEE-0410	The TEE SHALL provide Applications with access to a Secure Storage facility, at least compliant with Profile 2		✓
ATE-TEE-0420	An Open TEE SHALL provide Applications with access to a UICC Secure Channel facility, at least compliant with Profile 1	✓	
ATE-TEE-0430	A TEE MAY provide Applications with access to a UICC Secure Channel facility, at least compliant with Profile 1	✓	
ATE-TEE-0440	An Open TEE SHALL provide Applications with access to a UICC Secure Channel facility, at least compliant with Profile 2		✓
ATE-TEE-0450	A TEE MAY provide Applications with access to a UICC Secure Channel facility, at least compliant with Profile 2		✓
ATE-TEE-0460	An Open TEE SHALL provide Applications with access to a SUIO facility compliant with Profile 1	✓	✓
ATE-TEE-0470	An Open TEE SHALL provide Applications with access to a GBA facility, at least compliant with Profile 1	✓	
ATE-TEE-0480	An Open TEE SHALL provide Applications with access to a GBA facility, at least compliant with Profile 2		✓

4.4.4 APPLICATION LIFECYCLE REQUIREMENTS

REQ. ID	REQUIREMENT	PROFILE 1	PROFILE 2
ATE-TEE-0490	If the TEE can install Applications, then the TEE SHALL have an Application Installation policy, and the TEE SHALL only allow the Installation of Applications that comply with this Application Installation policy	✓	✓
ATE-TEE-0500	If the TEE can install Applications, then the TEE SHOULD provide a mechanism to disable revoked Applications	✓	✓
ATE-TEE-0510	The TEE SHOULD protect the Confidentiality of confidential Application Code (confidential A.EE.APP.C) during storage	✓	✓
ATE-TEE-0520	If the TEE can install Applications, the TEE SHOULD provide facilities to maintain the Confidentiality of confidential Application Code and Data (A.EE.APP.C and A.EE.APP.D) during Installation, Upgrade and Uninstallation		✓
ATE-TEE-0530	A TEE MAY have the capability to install Applications	✓	✓
ATE-TEE-0540	An Open TEE SHALL have the capability to install Applications	✓	✓
ATE-TEE-0550	It SHALL be possible to install an Application over the air on an Open TEE	✓	✓

An example of an Application Installation policy in Requirement ATE-TEE-0490 would be the ‘OMTP Application Security Framework’ [3].

4.4.5 APPLICATION REQUIREMENTS

REQ. ID	REQUIREMENT	PROFILE 1	PROFILE 2
ATE-TEE-0560	According to their Security Properties, Open TEE Application Code, Data and Keys (A.EE.APP.C, A.EE.APP.D and A.EE.APP.K) belonging to one TEE Application SHALL NOT be accessible from another Application within the TEE unless the other Application is authorised to have access and access is via an authorised communications channel	✓	✓
ATE-TEE-0570	TEE Application Code (A.EE.APP.C) that requires Integrity and/or Authenticity SHALL be checked before Execution, and Execution SHALL NOT proceed if these checks fail	✓	✓
ATE-TEE-0580	TEE Application Data and Keys (A.EE.APP.D and A.EE.APP.K) that require Integrity and/or Authenticity SHALL be checked before use, and usage SHALL NOT proceed if these checks fail	✓	✓
ATE-TEE-0590	The TEE SHOULD prevent any unauthorised Application from accessing the Application code (A.EE.APP.C) of an Application requiring Confidentiality protection while the Application requiring Confidentiality is loaded		✓
ATE-TEE-0600	When being used or modified, confidential TEE Application data (A.EE.APP.D) SHOULD be Confidentiality protected against potential eavesdropping by all components except the TEE in which the confidential TEE Application data (A.EE.APP.D) is processed and other Applications executing within that TEE	✓	✓

4.4.6 TRUSTED EXECUTION ENVIRONMENT SELF-PROTECTION

REQ. ID	REQUIREMENT	PROFILE 1	PROFILE 2
ATE-TEE-0610	TEE Code, Data and Key Assets that require Integrity and Authenticity SHALL be checked before use unless they are stored in memory that preserves at least Integrity and Authenticity	✓	✓
ATE-TEE-0620	The TEE Code Assets of a TEE Profile 1 SHALL be protected by a Flexible Secure Boot of at least Profile 1	✓	
ATE-TEE-0630	The TEE Code Assets of a TEE Profile 2 SHALL be protected by a Profile 2 Flexible Secure Boot		✓
ATE-TEE-0640	The TEE Code Assets of a Profile 1 TEE MAY be protected by a RIC of at least Profile 1	✓	
ATE-TEE-0650	The TEE Code Assets of a TEE Profile 2 MAY be protected by a Profile 2 RIC		✓

4.4.7 TRUSTED EXECUTION ENVIRONMENT ISOLATION

REQ. ID	REQUIREMENT	PROFILE 1	PROFILE 2
ATE-TEE-0660	Any mechanism responsible for switching or sharing Assets between the TEE and any other EE SHALL be protected against at least all the Attacks considered in all the EEs whose Assets it manages	✓	✓
ATE-TEE-0670	Any mechanisms responsible for switching or sharing Assets between the TEE and any other EE SHALL ensure the enforcement of all requirements related to those Assets	✓	✓

REQ. ID	REQUIREMENT	PROFILE 1	PROFILE 2
ATE-TEE-0680	If resources are de-allocated from a TEE and allocated to another EE, all data remaining in those resources belonging to the TEE SHALL either be erased before the resource is reallocated, or the data SHALL continue to be protected to the same degree as in the TEE, unless this TEE data is being transferred to the EE and the resource re-allocation is an authorised communications channel	✓	✓
ATE-TEE-0690	According to their Security Properties, TEE Code, Data and Key Assets SHALL NOT be accessible to other EEs except via authorised inter-EE communications channels. Note: Authorised inter-EE communications channels may include the use of shared resources such as memories	✓	✓
ATE-TEE-0700	When required by the communicating Applications, the TEE SHALL be able to protect the Confidentiality and Integrity of data exchanged between two Applications within the TEE		✓

4.4.8 INTER-EXECUTION ENVIRONMENT COMMUNICATIONS

This sub-section contains requirements related to inter-EE communications. The two communicating EEs (or TEEs if applicable) are called “Peer Execution Environments”. If a communication channel is established between two Applications running on two different EEs they are called “Peer Applications”.

For the purposes of the following requirements, we consider an Application to be the implementation of a use case (or part of a use case) which executes within an Execution Environment. For example, if a DRM agent runs within a single EE, then the DRM agent is considered to be an Application. If the agent is distributed between two different EEs, then the two portions are considered to be separate Applications that interact to provide a DRM agent.

Although the following requirements relate to inter-EE communications, we only mandate requirements on the TEE side of communications since we make no assumptions about the trustworthiness of other EEs. When a TEE Application is communicating with a peer Application in another EE, the TEE Application's ability to identify and authenticate the peer Application depends on the trustworthiness of the EE in which the peer Application resides. This is true even if the peer Application possesses credentials to authenticate itself, since the EE is ultimately responsible for storing and transmitting these credentials along with other credentials (e.g. Application IDs) to the TEE. Therefore if a TEE Application needs to authenticate Peer Applications, this indirectly imposes security requirements on the other EE (e.g. the OS of the EE may need to be signed). When designing TEEs and TEE Applications that communicate with other EEs, the above trust implications need to be carefully considered.

REQ. ID	**REQUIREMENT**	**PROFILE 1**	**PROFILE 2**
ATE-TEE-0710	It SHALL be possible to establish communication channels between two EEs, where at least one of the EEs is a TEE, if and only if they are authorised to communicate with each other. This authorisation MAY be given implicitly by the way the manufacturer configures the ME at manufacture	✓	✓
ATE-TEE-0720	In a communications channel between two Peer Applications running on different EEs, where at least one of the EEs is a TEE, the exchanged data exposed at the EE Application SHOULD NOT be accessible to any Applications other than the peer EE Application. Note: this requirement only relates to the EE. The equivalent *TEE* requirement is covered by ATE-TEE-0560	✓	✓

REQ. ID	REQUIREMENT	PROFILE 1	PROFILE 2
ATE-TEE-0730	If a peer EE or TEE requests communications with a TEE (called TEE X) and TEE X is notified of Integrity failures of TEE X Code and/or Data that could affect communications with the peer EE or TEE, TEE X SHALL reject communications with the peer EE or TEE and notify the RIC Error Handler (if available) of the Integrity failure, and the TEE MAY notify the peer EE or TEE of the Integrity failure	✓	✓
ATE-TEE-0740	If the TEE is notified of an Integrity failure of any sensitive Application Code and/or Data Assets, it SHALL: • Notify that to the peer Application requesting a secure communication channel OR • Reject access to the peer Application requesting a secure communication channel	✓	✓
ATE-TEE-0750	In the case that there are at least two EEs and at least one EE is a TEE, it SHALL be possible to launch from an EE an Application within a peer TEE	✓	✓
ATE-TEE-0760	It SHALL NOT be possible for an EE to communicate with or manipulate a TEE through any other means than through the authorised TEE interface	✓	✓
ATE-TEE-0770	In the case that there are at least two EEs and at least one EE is a TEE, it SHOULD be possible to trigger the launch from a TEE an Application within the peer EE	✓	✓

REQ. ID	REQUIREMENT	PROFILE 1	PROFILE 2
ATE-TEE-0780	Any Open TEE SHALL be able to communicate directly or through other EEs with the UICC (e.g. as defined in the section “Operator Application APIs for UICC Access” in ‘OMTP UICC/(U)SIM Defragmentation and Requirements: OMTP UICC0’ [12])	✓	✓
ATE-TEE-0790	Any TEE MAY be able to communicate directly or through other EEs with the UICC (e.g. as defined in the section “Operator Application APIs for UICC Access” in ‘OMTP UICC/(U)SIM Defragmentation and Requirements: OMTP UICC0’ [12])	✓	✓

5 SECURE STORAGE

This chapter describes the security requirements that pertain to the Secure Storage facility, for the two different Profiles.

An introduction to the Secure Storage Facility is provided in section 5.1.

The list of Actors and Actors' concerns is provided in section 5.2.

The list of Secure Storage Assets is defined in section 5.3.

The security requirements are defined in section 5.4.

5.1 AN INTRODUCTION TO THE SECURE STORAGE FACILITY

5.1.1 SECURE STORAGE FACILITY

The Secure Storage is a facility (hereafter called SST) that receives Sensitive Objects (A.SST.DO.D) and securely stores them as long as is required, in such a way that it maintains the Integrity and/or Confidentiality security properties of these Assets. Sensitive Objects can represent any information (data, code, keys), but are treated as opaque data by the SST.

The SST maintains general data (A.SST.AAM.D), used by the SST Application Assets Manager (A.SST.AAM.C). That data could store, for instance, the security properties of Sensitive Objects and Access Control policies for Sensitive Objects.

The SST may store the Sensitive Objects in appropriate Security-Preserving Memories: either Integrity-Protected Memory (AC.SST.IPM.H), Integrity and Confidentiality-Protected Memory (AC.SST.ICPM.H) or Integrity-Checked Memory (AC.SST.ICM.H). Definitions of these Security-Preserving Memories can be found in chapter 15. Alternatively, the SST may cryptographically protect certain of the Sensitive Object's security properties, by transforming it into a Cryptographically-Protected Data Object (A.SST.CPDO.D).

If certain security properties of the Sensitive Objects are not adequately protected by the applied cryptographic mechanism, (e.g. if A.SST.CPDO.D requires Integrity protection), then the Cryptographically-Protected Data Object shall be stored in the appropriate Security-Preserving Memory to protect it.

During the cryptographic protection of Sensitive Objects, the SST might manipulate cryptographic keys, called Secure Storage Keys (A.SST.KM.K), which can be either keys derived from the Hardware Unique Key (defined and required in the HU1 requirement of the 'OMTP Trusted Environment: OMTP TR0' [1]), or Application-specific keys. The SST Key Manager (A.SST.KM.C) is responsible for handling these keys and they are held in Integrity and Confidentiality-Protected Memory (AC.SST.ICPM.H) while being manipulated. It is the SST Key Manager that performs the cryptographic operations which convert Sensitive Objects into/from Cryptographically Protected Objects.

The Secure Storage Keys may be stored in Integrity and Confidentiality-Protected Memory (AC.SST.ICPM.H).

Alternatively, they may be cryptographically protected before storage either in General Purpose Memory or in an appropriate Security-Preserving Memory (either AC.SST.IPM.H or AC.SST.ICPM.H or AC.SST.ICM.H).

In addition, the SST will protect Sensitive Objects and Secure Storage Keys when being transferred between memories.

Note: A UE could provide a hierarchy of Secure Storage facilities. The facility described in this document is the inner one, with the most stringent security requirements, on top of which other ones can be layered.

We consider two sets of SST security requirements which we call Profiles 1 and 2. Each SST Profile's requirements define certain required functionalities of the SST and also the scope of the Attacks to which SST Assets should be resistant. The two Profiles offer incremental levels of security (that is, Profile 2 offers stronger security than Profile 1).

Both Profiles protect security properties of the Sensitive Objects from identified Threats. They may rely on the UICC to store and manage the most Sensitive Objects (data, code, keys).

The secure storage implementations may also make use of other types of memories, such as embedded NAND and NOR flash memories, on board EEPROMS and removable storage cards.

5.2 SECURE STORAGE ACTORS AND THEIR CONCERNS

Table 6 describes all Actors involved in the Secure Storage, their main concerns, and the trust link between them.

Table 6: Secure Storage Actors and Their Concerns

Actor	Main Concerns (in this context)
Operators (O) and other Service Providers (SP)	Protection of their revenue from unauthorised access to services. Protection of content and content usage rights obtained from Content Providers. Protection of SP sensitive keys, code and data (e.g. for intellectual property protection) Protection of UE Assets, such as IMEI, UICC life cycle and ME Personalisation Information (see 'OMTP Trusted Environment: OMTP TR0' [1]) and of sensitive Application Assets. Avoid Denial-of-Service

Actor	Main Concerns (in this context)
Content Providers (CP)	Protection of their content and content usage rights
End User (U)	Protection of their personal data (e.g. identity, banking information, sensitive contacts). Avoid Denial-of-Service
Terminal Manufacturer (M)	Ensuring Terminals cannot be hacked. Ensuring revenue and content is not lost from their Terminals. Protection of their sensitive keys, code and data (e.g. for intellectual property protection)
UICC Manufacturer (UM)	Provisioning of their code and data. Protection of the assets of the MNO
Attacker (A)	Obtaining unauthorised access to Key Storage, then to keys protecting the sensitive Application Assets. Obtaining unauthorised access to Storage Spaces, then to sensitive Application Assets (keys, code and data). Successful Denial-of-Service. Successful Replay Attack (e.g. to use indefinite rights on DRM-protected content)
Removable Device Manufacturer (RDM)	Ensuring removable devices cannot be hacked. Protection of their sensitive keys, code and data (e.g. for intellectual property protection)

5.3 SECURE STORAGE ASSETS

5.3.1 SECURE STORAGE ASSETS TABLE

Table 7 lists Assets in the implementation of an SST that require protection. The Assets table describes various properties of SST Assets that were used to derive the SST requirements in section 5.4.

Table 7: Secure Storage Assets

Asset	Description	Owner†	Protection Level/Level of Sensitivity	Security Properties Required
A.SST.xxx.y or AC.SST.xxx.y with y=K for Key, y=D for Data, y=C for Code, y=H for Hardware		*See Actors List*	*Sensitive (S)* *Operational (O)*	*Integrity (I)* *Confidentiality (C)* *Authenticity (Au)* *Non-Replay (NR)* *Terminal Binding (TB)*
A.SST.DO.D	Sensitive Objects: Objects provided to the SST by the calling Application. It can represent any information (data, code, keys), but is treated as opaque data by the Secure Storage facility. Requirements on this Asset will apply when this Asset under the control of the SST	M UM SP O U RDM	S	I, C**, NR*
A.SST.AAM.C	SST Application Assets Manager: Code of the SST that manages Sensitive Objects	M UM RDM	S	I, Au
A.SST.AAM.D	SST Application Assets Manager Data	M UM SP O U RDM	S	I,C**,Au
A.SST.CPDO.D	Cryptographically-Protected Data Objects: Data objects provided by the SST once they have been cryptographically protected	M UM SP O RDM	S***	I***, C***, Au***, NR***

Asset	Description	Owner†	Protection Level/Level of Sensitivity	Security Properties Required
A.SST.KM.K	SST Keys: Keys used to protect the security properties of Sensitive Objects or other keys, by transforming them into. Cryptographically-Protected Data Objects	M UM SP O RDM	S	I, C, Au, TB[1]
A.SST.KM.C	SST Key Manager: Code of the SST that manages keys	M UM RDM	S	I, Au
A.SST.KM.D	SST Key Manager Data	M UM RDM	S	I, C**[2], Au
AC.SST.IPM.H	SST Integrity-Protected Memory	M UM RDM	S	I
AC.SST.ICPM.H	SST Integrity and Confidentiality-Protected Memory	M UM RDM	S	I, C
AC.SST.ICM.H	SST Integrity-Checked Memory	M UM RDM	S	I

Notes:

† Denotes that the given set of owners of a specific Asset is only indicative.

* Denotes the fact that this Asset may or may not require replay protection. The need for replay protection is dictated by the requirements of the object provided to the SST.

** Denotes the fact that these Assets may or may not require Confidentiality. For Sensitive Data Objects this depends on the requirements of the Application calling the SST. Otherwise this may depend on the SST implementation and the purpose of the Asset. If these Assets require Confidentiality protection, this will be denoted by a (C) extension (e.g. A.SST.DO.D(C)).

[1] Some Sensitive Objects might require their data to be uniquely associated, or bound, to the ME, for example using a key derived from the HW Unique Key (A.EE.HUK.K).

[2] Confidentiality might be necessary for part of the data, e.g. the part used for key generation/derivation.

*** Denotes the fact that security sensitivity and security properties depend on the cryptographic protection applied and the security properties of the original Sensitive Data Object (A.SST.DO.D).

5.3.2 SECURE STORAGE ASSET GROUPING

In order to facilitate requirement understanding, the following groups of Assets are defined:

- **SST Code Assets** group contains all SST code Assets: A.SST.AAM.C and A.SST.KM.C
- **SST Data Assets** group contains all SST data Assets: A.SST.DO.D, A.SST.AAM.D, A.SST.CPDO.D and A.SST.KM.D
- **SST Confidential Data Assets** group contains all SST data Assets that require Confidentiality: A.SST.DO.D(C), A.SST.AAM.D(C) and A.SST.KM.D(C)
- **SST Key Assets** group contains all SST key Assets: A.SST.KM.K
- **SST Hardware Assets** group contains all SST hardware Assets: AC.SST.IPM.H, AC.SST.ICPM.H and AC.SST.ICM.H

5.4 REQUIREMENTS

5.4.1 SECURE STORAGE CORE REQUIREMENTS

The Asset security properties referred to in the following requirements are defined in section 5.3.1.

In addition to TR1 Core Requirements defined in chapter 3 “Trusted Environment Core requirements”, further core requirements are defined for SST in this section.

REQ. ID	REQUIREMENT	PROFILE 1	PROFILE 2
ATE-SST-0800	Confidentiality of SST Key Assets SHALL be defended against the Threats contained in Threat Group 5 “Extract Secret via Bus Monitoring (Hardware Probes)”	✓	✓
ATE-SST-0810	Confidentiality of SST Confidential Data Assets SHALL be defended against the Threats contained in Threat Group 5 “Extract Secret via Bus Monitoring (Hardware Probes)”		✓

REQ. ID	REQUIREMENT	PROFILE 1	PROFILE 2
ATE-SST-0820	The security properties of SST Key Assets SHALL be defended against the Threats contained in Threat Group 6 "Mod Chip Attacks on Data in External RAM”	✓	✓
ATE-SST-0830	The security properties of SST Confidential Data SHALL be defended against the Threats contained in Threat Group 6 "Mod Chip Attacks on Data in External RAM”		✓
ATE-SST-0840	The security properties of SST Key Assets SHALL be defended against the Threats contained in Threat Group 7 “Attack by Replacement of Flash When Power is On (Post-Boot)”	✓	✓
ATE-SST-0850	The security properties of SST Confidential Data SHALL be defended against the Threats contained in Threat Group 7 “Attack by Replacement of Flash When Power is On (Post-Boot)”.		✓
ATE-SST-0860	The security properties of SST Key Assets SHALL be defended against the Threats contained in Threat Group 1 “Hardware Modules Used for Accessing Memories”	✓	✓
ATE-SST-0870	The Confidentiality of SST Key Assets SHALL be defended against the Threats contained in Threat Group 2 “CLCD Used for Displaying Memories and Interfering With Displayed Data”	✓	✓

Table 8 summarises the Secure Storage Core Requirements. Each cell contains the Profiles for which an Asset group have to be protected against a Threat group.

Table 8: Secure Storage Threat Grouping

Threat Group	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Threats Assets	Hardware Modules Used for Accessing Memories	CLCD Used for Displaying Memories and Interfering With Displayed Data	Bypass Security by Removal of Battery Power or Removal of External Memory Card	Attack by Replacement of Flash When Power is Off (Pre-Boot)	Extract Secret via Bus Monitoring (Hardware Probes)	Mod Chip Attacks on Data in External RAM	Attack by Replacement of Flash When Power is On (Post-Boot)
A.SST.AAM.C A.SST.KM.C	2	2 (C)* or N/A	1, 2	1 2	2 (C)* or N/A	-	-
A.SST..DO.D A.SST.AAM.D A.SST.CPDO.D A.SST.KM.D	2	2 (C)* or N/A	1, 2	1 2	2 (C)* or N/A	-	-

A.SST..DO.D(C)
A.SST.AAM.D(C)
A.SST.KM.D(C)
	2	2 (C)	1, 2	1 2	2 (C)	2	2

A.SST. KM.K	1, 2	1, 2 (C)	1, 2	1 2	1, 2 (C)	1, 2	1, 2
AC.SST.IPM.H AC.SST.ICPM.H AC.SST.ICM.H	-	-	-	-	-	-	-

Notes:

(C) Denotes the fact that only Confidentiality of these Assets needs to be defended against this Threat group.

(C)* Denotes the fact that these Assets may not require Confidentiality protection, but if they do, then this Threat group must be defended against.

All of the following SST requirements shall be considered only with respect to the relevant Threats for the associated Profile. For example, a requirement to maintain Confidentiality of keys in a Profile 1 SST will be considered with respect to the Threats considered for SST Keys in Profile 1 as defined above.

5.4.2 SECURE STORAGE GENERAL REQUIREMENTS

REQ. ID	REQUIREMENT	PROFILE 1	PROFILE 2
ATE-SST-0880	The SST SHALL maintain the security properties of stored Sensitive Objects whilst these Assets are stored in the SST	✓	✓
ATE-SST-0890	According to the required security properties of the Sensitive Objects, the SST SHALL: (a) store Sensitive Objects in appropriate Security-Preserving Memories OR (b) cryptographically protect the required security properties of the Sensitive Objects by transforming them into Cryptographically-Protected Data Objects AND MAY store all Cryptographically Protected Data with both Integrity protection and Confidentiality OR (c) do both (a) and (b) above OR (d) store Sensitive Objects in the UICC, in memory maintaining the security properties of the Sensitive Objects	✓	✓
ATE-SST-0900	According to the required security properties of the Sensitive Objects, the SST SHALL process the Sensitive Objects in either SST Integrity-Protected Memory or SST Integrity and Confidentiality-Protected Memory. Implementations MAY process all Sensitive Objects in SST Integrity and Confidentiality-Protected Memory	✓	✓
ATE-SST-0910	The SST SHALL protect the security properties of the Sensitive Objects while being transferred between SST Security-Preserving Memories	✓	✓

REQ. ID	REQUIREMENT	PROFILE 1	PROFILE 2
ATE-SST-0920	The SST SHALL protect the security properties of the Cryptographically-Protected Data Objects while being transferred between SST Security-Preserving Memories	✓	✓
ATE-SST-0930	If the SST stores Sensitive Objects or Cryptographically-Protected Data Objects in the UICC, the SST SHALL protect the security properties of these Objects while being transferred between ME Security-Preserving Memories and the UICC	✓	✓

5.4.3 SECURE STORAGE ACCESS CONTROL REQUIREMENTS

REQ. ID	REQUIREMENT	PROFILE 1	PROFILE 2
ATE-SST-0940	Any Sensitive Object provided to the SST SHALL only be accessible by the SST Application Assets Manager, and by any Applications authorised by the SST Application Assets Manager's Access Control policy attached to the Sensitive Object	✓	✓
ATE-SST-0950	The SST services provided by the SST Application Assets Manager SHALL be accessible by the Applications calling the SST	✓	✓
ATE-SST-0960	The SST Application Assets Manager Data SHALL only be accessible by the SST Application Assets Manager	✓	✓
ATE-SST-0970	Cryptographically-Protected Data Objects SHALL be accessible by the SST Application Assets Manager and by any Applications authorised by the SST Application Assets Manager's Access Control policy attached to these objects	✓	✓

REQ. ID	REQUIREMENT	PROFILE 1	PROFILE 2
ATE-SST-0980	The SST Key Manager SHALL only be accessible by the SST Application Assets Manager	✓	✓
ATE-SST-0990	The SST Key Manager Data SHALL only be accessible by the SST Key Manager	✓	✓
ATE-SST-1000	The SST Keys SHALL only be accessible by the SST Key Manager	✓	✓
ATE-SST-1010	If available, SST Integrity-Protected Memory SHALL be accessible by the SST Application Assets Manager and by the SST Key Manager	✓	✓
ATE-SST-1020	If available, SST Integrity and Confidentiality-Protected Memory SHALL be accessible by the SST Application Assets Manager and by the SST Key Manager	✓	✓
ATE-SST-1030	If available, SST Integrity-Checked Memory SHALL be accessible by the SST Application Assets Manager and by the SST Key Manager	✓	✓

5.4.4 SECURE STORAGE APPLICATION ASSETS MANAGER REQUIREMENTS

REQ. ID	REQUIREMENT	PROFILE 1	PROFILE 2
ATE-SST-1040	The SST Application Assets Manager SHALL be able to bind the *supplied* identity of the EE (or TEE) Application to particular Sensitive Objects, and limit access (to those Sensitive Objects) only to Applications that are so bound to those Sensitive Objects. That binding information SHALL include (but it is not limited to) the Application's identity and whether or not it is part of the TEE	✓	✓

REQ. ID	REQUIREMENT	PROFILE 1	PROFILE 2
ATE-SST-1050	It SHALL not be possible for Trusted Identities to be Spoofed by Untrusted Identities	✓	✓
ATE-SST-1060	The SST Application Assets Manager SHALL apply and maintain the Access Control policy attached to the Sensitive Objects and to Cryptographically-Protected Data Objects, while those Assets are under the control of the Secure Storage facility	✓	✓
ATE-SST-1070	According to the security properties of the Sensitive Objects, the SST Application Assets Manager SHALL protect Sensitive Objects from Replay Attacks	✓	✓

5.4.5 SECURE STORAGE KEY MANAGER REQUIREMENTS

REQ. ID	REQUIREMENT	PROFILE 1	PROFILE 2
ATE-SST-1080	The provisioning of any SST Keys used by the SST Key Manager SHALL guarantee the Confidentiality and the Integrity of these keys	✓	✓
ATE-SST-1090	The Integrity and Authenticity of SST Keys SHALL be verified before use OR The Integrity and Authenticity of SST Keys SHALL be verified before being stored in Integrity-Protected Memory and subsequently used from that memory	✓	✓
ATE-SST-1100	The SST Key Manager SHALL protect the security properties of the SST Keys while being manipulated, stored and transferred between memories	✓	✓

REQ. ID	REQUIREMENT	PROFILE 1	PROFILE 2
ATE-SST-1110	The handling of the SST Keys SHALL be done by the SST Key Manager	✓	✓
ATE-SST-1120	The cryptographic material used for protection of the Sensitive Objects SHALL be stored in such a way that it does not compromise the security properties of the Sensitive Object	✓	✓
ATE-SST-1130	It SHALL NOT be possible for one Application to deduce the SST keys associated with another Application	✓	✓

6 FLEXIBLE SECURE BOOT

This chapter describes the security requirements that pertain to Flexible Secure Boot for two different Profiles.

An introduction to Flexible Secure Boot is provided in section 6.1.

The list of Actors and Actors' concerns is provided in section 6.2.

The list of Flexible Secure Boot Assets is defined in section 6.3.

The security requirements are defined in section 6.4.

6.1 FLEXIBLE SECURE BOOT DEFINITION

Secure Boot directly or indirectly verifies the Integrity of the ME code base (A.FSB.CB.D). The ME code base can be partitioned amongst several memory types including Integrity-Protected Memory devices. These devices may physically protect the Integrity of portions of their contents. However, despite having this capability, the Integrity-Protected Memory device must have its capabilities verified (e.g. Access Control) prior to use.

Secure Boot detects changes made by a Threat Agent that may affect the code base in an unauthorised way. The Threat Agent may for example modify the code base by re-flashing the Device via debug ports, or other legitimate access points to the device as described in T.HWE.002 and T.HWE.003 of 'Security Threats on Embedded Consumer Devices' [2] or by replacing the Flash components as described in T.CLO.005 in the same document..

The Integrity of the code base can be ensured using one of two methods (or a combination of the two). The first method is to verify the Integrity of the code base at system boot time. This method is called Integrity-Verification Secure Boot. The second method is to store the code base in Integrity-Protected Memory. This method is called Integrity-Protection Secure Boot.

'OMTP Trusted Environment: OMTP TR0' [1] has already defined requirements for Secure Boot. This sub-section explains the relationship between the requirements for Secure Boot in TR0 and the requirements for Flexible Secure Boot in TR1. TR0 and TR1 are complementary - TR1 does not explicitly assume that all of TR0 applies, though in practice, implementations compliant to TR1 should be compliant to many of the TR0 requirements. Operators may use both TR0 and TR1 requirements. In terms of technical detail, TR0 deals in detail with mechanisms for verification of the initial Secure Boot code and the process of extending trust from the initial Secure Boot code to the whole platform. However, TR0 does not contain any requirements about update of the code base. TR1 on the other hand, concentrates on requirements related to update of the code base. The core requirements in TR1 give indication of security strength required for Secure Boot but this is not the case in TR0. In addition, the cryptographic strength of protection required in TR1 is different to that required in TR0.

6.1.1 INTEGRITY VERIFICATION DURING SECURE BOOT

The Secure Boot Integrity verification process is initiated by code whose Integrity is guaranteed (A.FSB.ISB.C) and may call other code components (A.FSB.ASB.C) to complete the process. Any additional code components that are used in the Secure Boot process must be verified before they are executed.

Code Integrity is directly verified by cryptographic means. This can be a Hash value (A.FSB.HASH.D) (if the Hash is stored in Integrity-Protected Memory), a signed Hash value (A.FSB.HASH.D, A.FSB.VER.K) or other appropriate methods. Note that symmetric or asymmetric cryptographic mechanisms could be used to authenticate Hash values.

One element in the Secure Boot process is the definition of which parts of the program code and data must be verified. In this document, this is called the code base list (A.FSB.CBL.D).

6.1.2 USE OF INTEGRITY PROTECTED MEMORIES DURING SECURE BOOT

A Secure Boot process may also make use of Integrity-Protected Memory (AC.FSB.IPM.H) to obtain trusted code during the boot process. However this code cannot be used until the Integrity-Protected Memory has been verified as being the Integrity-Protected Memory that is expected. This means it should be verified that the Integrity-Protected Memory has not been replaced by another, unauthorised Integrity-Protected Memory. The actual method of verification is decided by design and the capabilities of the Integrity-Protected Memory, but all Assets involved in the verification must be trusted and their Integrity and other required Security Properties must be intact. All these Security Properties shall either be intrinsic or guaranteed by the Secure Boot process.

If an Integrity-Protected Memory is verified prior to use, there is no need for the Integrity of the code base stored within to be directly verified at system boot.

6.1.3 FLEXIBLE SECURE BOOT

Flexible Secure Boot adds the possibility of modifying the code base via an update code base image (A.FSB.UPD.D) which is obtained via an over-the-air download or via some other connection to the ME. An Application (A.FSB.UPD.C) is run which updates the code base with the update code base image. Depending on how the code base list is defined, this may modify the code base list.

The modification of the code base list is a security-sensitive operation.

The update image must be verified for Authenticity and Integrity before the code base is updated.

6.1.4 CODE BASE

There are three possible configurations of the code base.

The first configuration is one where there are two distinct code images that can be executed on the Terminal. The first code image is the main code base that is normally executed on the Terminal. The second code image is the Emergency Code Base. The Emergency Code Base will be executed if there is a reason that prevents the main code base from being executed. For example, the main code base may not be loaded if the UICC is not present in the phone. Another example is if the Secure Boot process fails to verify the Integrity of the main code base. The emergency code normally provides limited functionality such as the ability to call emergency numbers, e.g. 911.

The second configuration is one where there is one code image that is divided into parts, critical and non-critical code. For example, the OS would normally be considered to be part of the critical code, while a game would be considered to be non-critical code.

The third configuration is one where there is one, monolithic code base.

These three configurations are addressed by three distinct requirements in the Flexible Secure Boot Specific Requirements chapter.

6.2 FLEXIBLE SECURE BOOT ACTORS AND THEIR CONCERNS

Table 9 describes all the Actors involved in the Flexible Secure Boot mechanism, their main concerns, and the trust link between them.

Table 9: Flexible Secure Boot Actors and Their Concerns

Actor	Main Concerns (in this context)
Operators (O)	Operators wish to be able to run Applications on an ME that boots from a known code base. Operators trust the Terminal manufacturer to implement a Secure Boot procedure that ensures that only authentic code can boot in the system
Other Service Providers (SP)	Service providers wish to be able to run Applications on a ME that boots from a known code base. Service providers trust the Terminal manufacturer to implement a Secure Boot procedure that ensures that only authentic code can boot in the system

Actor	Main Concerns (in this context)
Terminal Manufacturer (M)	The Terminal manufacturer implements a Secure Boot procedure that ensures the Integrity of the code base at system boot. The Terminal manufacturer trusts the operator and service providers to ensure that their code authorisation process is secure
Application Provider (AP)	The Application Provider wishes to ensure that its code runs as written on the ME, and that it executes in the anticipated environment
End User (U)	The end user wants the ME to operate correctly because only authentic code can boot on the ME
Attacker (A)	The Attacker attempts to insert unauthorised code into the phone by hacking the Secure Boot procedure or code authorisation mechanisms

6.3 FLEXIBLE SECURE BOOT ASSETS

6.3.1 FLEXIBLE SECURE BOOT ASSETS TABLE

Table 10 lists Assets in the implementation of an FSB that require protection. The Assets table describes various properties of FSB Assets that were used to derive the FSB requirements in section 6.4.

Table 10: Flexible Secure Boot Assets

Asset (A) or Asset Container (AC)	Description	Owner†	Protection Level /Level of Sensitivity	Security Properties Required
A.FSB.xxx.y or AC.FSB.xxx.y with y=K for Key, y=D for Data, y=C for Code, y=H for Hardware		*See Actors List*	*Sensitive (S)* *Operational (O)*	*Integrity (I)* *Confidentiality (C)* *Authenticity (Au)*
A.FSB.ISB.C[3]	Initial Secure Boot code that initiates the Secure Boot procedure	M	S	I, Au

[3] It is assumed that the Integrity of the initial Secure Boot code is intrinsically guaranteed, e.g. it may be implemented as ROM code.

Asset (A) or Asset Container (AC)	Description	Owner†	Protection Level /Level of Sensitivity	Security Properties Required
A.FSB.ASB.C[4]	Additional Secure Boot code that is called by the initial Secure Boot to execute part of the Secure Boot procedure	M	S	I, Au
A.FSB.BIND.D	Binding data used to verify the Authenticity of the Integrity-Protected Memory. The specific nature of this data is determined by the technical solution used. A.FSB.BIND.D may include cryptographic keys.	M	S	I, Au, C*
A.FSB.VER.K	Verification keys that are used to verify the signature and Authenticity of the code base, both during the boot process, and when code base updates are received (if required)	M O SP	S	I, Au, C**
A.FSB.HASH.D	Hash digest values that verify the Integrity of the code base	M O AP	S	I, Au
AC.FSB.IPM.H[5]	Integrity-Protected Memory where code base is stored	M	S	I
A.FSB.CB.D	Code base data – the ME program code and data that is verified by the Secure Boot procedure. It is treated by the Secure Boot as data	M O AP	S	I, Au
A.FSB.UPD.C	Update code base Application – the Application that updates the code base with the code base image	M	S	I, Au
A.FSB.UPD.D	Update code base image – the code base image that is downloaded to replace some or all of the code base	M O AP	S	I, Au
A.FSB.CBL.D	Code base list – data structure that defines what program code and data must be verified by the Secure Boot procedure	M O AP	S	I, Au

Notes:

† Denotes that the given set of owners of a specific Asset is only indicative.

[4] It is assumed that the additional Secure Boot code is loaded from some program memory by the initial Secure Boot code. Therefore, its Integrity is not intrinsically guaranteed. This is the main difference between the two Secure Boot code Assets.

[5] Relevant only for the Integrity-Protection Secure Boot method.

* Denotes the fact that this Asset may or may not require Confidentiality. It is determined by the Security Property requirements of the specific Asset instance.

** Denotes the fact that the Confidentiality of these keys will depend on whether these keys are symmetric, Private or Public Keys.

6.3.2 FLEXIBLE SECURE BOOT ASSET GROUPING

In order to facilitate requirement understanding, the following groups of Assets are defined:

- **FSB Code Assets** group contains all FSB code Assets: A.FSB.ISB.C, A.FSB.ASB.C and A.FSB.UPD.C
- **FSB Data Assets** group contains all FSB data Assets: A.FSB.BIND.D, A.FSB.HASH.D, A.FSB.CB.D, A.FSB.UPD.D and A.FSB.CBL.D
- **FSB Hardware Assets** group contains FSB hardware Asset: AC.FSB.IPM.H
- **FSB Key Assets** group contains all FSB key Assets: A.FSB.VER.K

6.4 FLEXIBLE SECURE BOOT REQUIREMENTS

The Asset security properties referred to in the following requirement are defined in section 6.3.1.

6.4.1 FLEXIBLE SECURE BOOT CORE REQUIREMENTS

In addition to TR1 Core Requirements defined in Chapter 3 “Trusted Environment Core requirements”, further core requirements are defined for FSB in this section.

REQ. ID	REQUIREMENT	PROFILE 1	PROFILE 2
ATE-FSB-1140	The security properties of A.FSB.ISB.C SHALL be defended against the Threats contained in TR1 Threat Groups 1, 3, 4, 6 and 7	✓	✓

Table 11 summarises the Flexible Secure Boot Core requirement table that apply to FSB Assets. Each cell contains the Profiles for which an Asset group have to be protected against a Threat group.

Table 11: Flexible Secure Boot Threat Grouping

Threat Group	1	2	3	4	5	6	7
Threats Assets	Hardware Modules Used for Accessing Memories	CLCD Used for Displaying Memories and Interfering With Displayed Data	Bypass Security by Removal of Battery Power or Removal of External Memory Card	Attack by Replacement of Flash When Power is Off (Pre-Boot)	Extract Secret via Bus Monitoring (Hardware Probes)	Mod Chip Attacks on Data in External RAM	Attack by Replacement of Flash When Power is On (Post-Boot)
A.FSB.ASB.C A.FSB.UPD.C	2	2 (C)* or N/A	1, 2	1, 2	2 (C)* or N/A	-	-
A.FSB.BIND.D A.FSB.HASH.D A.FSB.CB.D A.FSB.UPD.D A.FSB.CBL.D	2	2 (C)* or N/A	1, 2	1, 2	2 (C)* or N/A	-	-
A.FSB.VER.K	2	2 (C)*	1, 2	1, 2	2 (C)*	2	2
AC.FSB.IPM.H	-	-	-	-	-	-	-
A.FSB.ISB.C	1, 2	-	1, 2	1, 2	-	1, 2	1, 2

Note:

(C)* Denotes the fact that these Assets may not require Confidentiality protection, but if they do, then this Threat group must be defended against.

6.4.2 FLEXIBLE SECURE BOOT SPECIFIC REQUIREMENTS

REQ. ID	REQUIREMENT	PROFILE 1	PROFILE 2
ATE-FSB-1150	The security properties of the Initial Secure Boot code, A.FSB.ISB.C, SHALL be intrinsically guaranteed	✓	✓
ATE-FSB-1160	The security properties of the additional Secure Boot code, A.FSB.ASB.C, SHALL be verified before this code is executed	✓	✓
ATE-FSB-1170	The security properties of the update code base Application, A.FSB.UPD.C, SHALL be verified before the update code base Application is executed	✓	✓

REQ. ID	**REQUIREMENT**	**PROFILE 1**	**PROFILE 2**
ATE-FSB-1190	In the case of Integrity-Protection Secure Boot, with reference to storage of A.FSB.BIND.D in the ME, A.FSB.BIND.D SHALL either Be considered an SST Sensitive Object (A.SST.DO.D) OR Be stored in some other way which protects its Security Properties.	✓	✓
ATE-FSB-1210	The Integrity and Authenticity of the update code base Application (A.FSB.UPD.C) and of the update code base image (A.FSB.UPD.D) and of the code base list (A.FSB.CBL.D) SHALL be maintained in one of the following ways: The update code base Application (A.FSB.UPD.C) SHALL be executed before the Platform Software is executed OR The update code base Application (A.FSB.UPD.C) SHALL be executed in a TEE that maintains the Integrity and Authenticity of the update code base Application (A.FSB.UPD.C), of the update code base image (A.FSB.UPD.D) and of the code base list (A.FSB.CBL.D)	✓	✓
ATE-FSB-1220	The security properties of the update code base image (A.FSB.UPD.D) SHALL be verified before the update code base is installed	✓	✓

REQ. ID	REQUIREMENT	PROFILE 1	PROFILE 2
ATE-FSB-1230	In the case of Integrity-Protection Secure Boot, the controller of the Integrity-Protected Memory (AC.FSB.IPM.H) SHALL verify the security properties of the update code base image (A.FSB.UPD.D) before updating the code base in the Integrity-Protected Memory	✓	✓
ATE-FSB-1240	In the case of Integrity-Protection Secure Boot, it SHALL NOT be possible to subvert the security of the FSB process by replacing one Integrity-Protected Memory device in an ME with another Integrity-Protected Memory device	✓	✓
ATE-FSB-1250	A Device MAY rollback to an update code base image that contains an earlier version number. The rollback to an update code base that contains an earlier version number SHALL only be possible after verifying that this rollback is authorised by the owner of the currently installed code package	✓	✓
ATE-FSB-1260	The Device SHALL support a minimum accepted software version number which cannot be reduced. The current software version number MAY be higher than this minimum version number, but cannot be rolled back below this minimum version number. New MEs SHALL have this minimum version number set to the highest current stable software version number	✓	✓
ATE-FSB-1270	If the Installation of the update code base image (A.FSB.UPD.D) fails, the update code base Application (A.FSB.UPD.C) SHALL revert to the currently installed code base	✓	✓

REQ. ID	REQUIREMENT	PROFILE 1	PROFILE 2
ATE-FSB-1280	The security properties of the code base list (A.FSB.CBL.D) SHALL be intrinsically guaranteed OR The security properties of the code base list SHALL be verified by the Secure Boot code	✓	✓
ATE-FSB-1290	If the Secure Boot process fails to verify the Integrity of the code base and if there is an additional Emergency Code Base, then the Secure Boot process MAY boot this emergency code after verifying the Integrity of the Emergency Code Base. Otherwise, the entire boot process SHALL be aborted	✓	✓
ATE-FSB-1300	If the code base is partitioned into critical and non-critical code, and the critical code has failed the Integrity verification, the entire boot process SHALL be aborted. If the critical code has passed the Integrity verification and part or all of the non-critical code has failed the Integrity verification, then the Secure Boot process SHALL either: · Boot the critical code and other verified non-critical code OR · Boot only the critical code OR · Abort	✓	✓
ATE-FSB-1310	If the Secure Boot process fails to verify the Integrity of the code base and this is the only available code, then the entire boot process SHALL be aborted	✓	✓

7 GENERIC BOOTSTRAPPING ARCHITECTURE

This chapter describes the security requirements related to Generic Bootstrapping Architecture (GBA) for two different Profiles.

An introduction to GBA is provided in section 7.1.

The list of Generic Bootstrapping Architecture Actors and Their Concerns is provided in section 7.2.

The list of GBA Assets is defined in section 7.3.

The security requirements are defined in section 7.4.

7.1 GENERIC BOOTSTRAPPING ARCHITECTURE DEFINITION

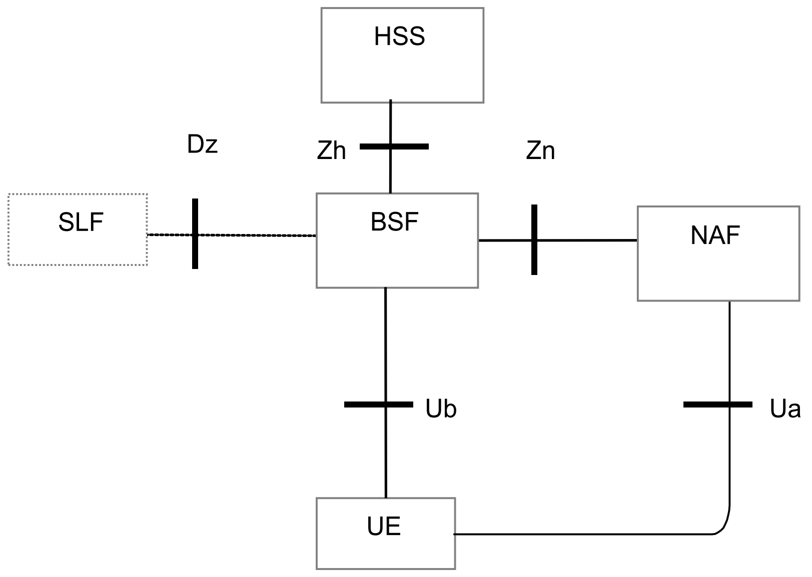
GBA is a method for using the existing security relationship between the mobile network operator and the USIM of the user, which is primarily for establishment of security on the 3GPP or GSM bearer layer, for Application layer purposes. It is specified in 3GPP TS 33.220 – ’Generic Authentication Architecture (GAA); Generic Bootstrapping Architecture (Release 7)’ [13].

The reader is assumed to understand the basics of 3GPP bearer layer security architecture. Those not familiar should refer to 3GPP TS 33.102 – ‘3G Security; Security Architecture (Release 7)’ [14].

With reference to Figure 1 below:

- Authentication Vectors for the user’s USIM are stored in the Home Subscriber Server (HSS, the IMS equivalent of the HLR)
- The Bootstrapping Server Function (BSF) is an operator-owned server that acts as the interface between the HSS and other entities for purposes of GBA.
- The Network Application Function (NAF) is a server owned by an operator or 3rd party that wishes to conduct secure Application layer communications with the user.
- The Subscription Locator Function (SLF) is required if there is more than one HSS for the operator and tells the BSF which HSS to use for a particular user.

Figure 1: GBA Architecture



7.1.1 GENERIC BOOTSTRAPPING ARCHITECTURE EXCHANGES

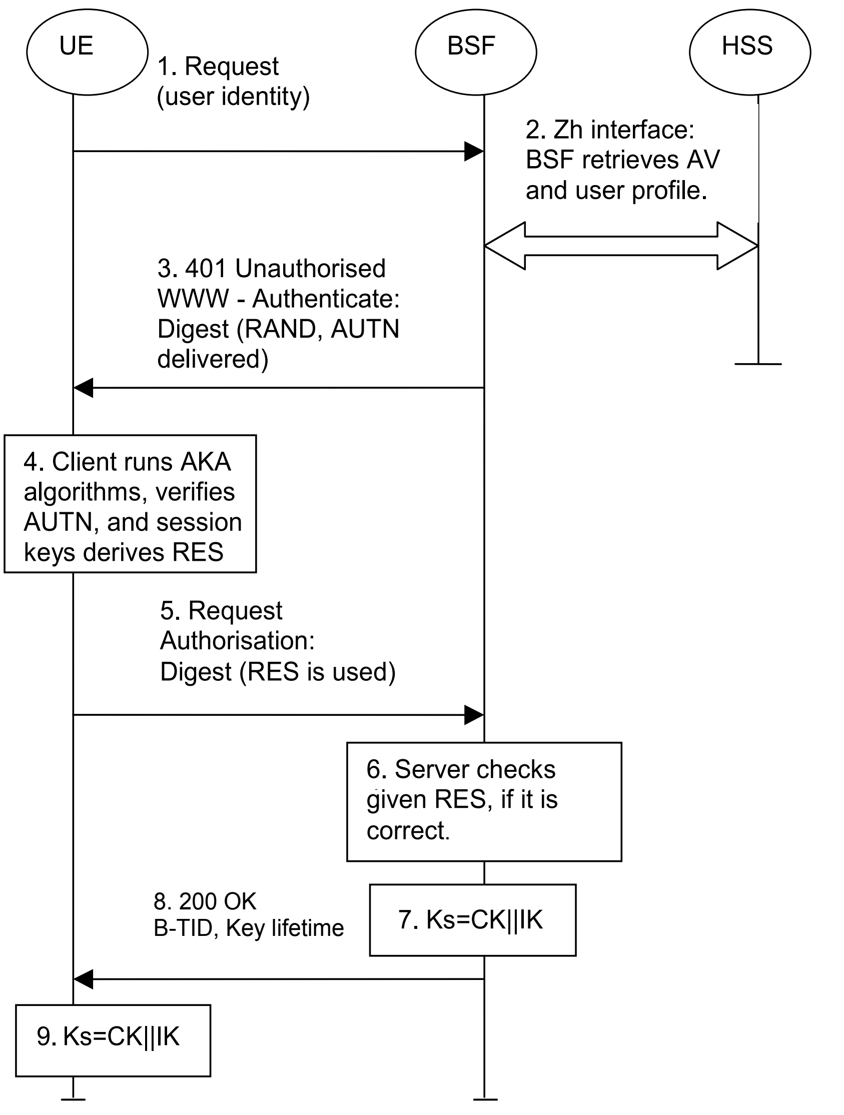
At a high level, GBA works according to the following sequence of steps:

- The UE sends an HTTP request to the BSF, initiating a bootstrapping run.
- The BSF requests Authentication Vectors (AVs) for the user from the HSS.
- The BSF sends the RAND and AUTN portions of the AV to the UE. AUTN is different in GBA_U and GBA_ME. In GBA_U, the MAC value is exclusive OR with truncated sha1 value of IK.
- The UE sends RAND and AUTN to the UICC. AUTN is verified authenticating the BSF/HSS to the UICC. RAND is processed and the response (RES) is returned to the BSF.
- BSF verifies the digest AKA response by doing the similar Hash with XRES. It doesn't compare the RES with XRES directly. If the verification is successful the UICC has been authenticated. The cipher and Integrity keys of the AV (CK and IK), are concatenated to form Ks, and a NAF-specific key K_S_(ext/int)NAF[6] to be used in Application layer

[6] This notation encompasses both GBA_ME and GBA_U cases. In the GBA_ME variant, K_SNAF is derived from Ks=CK||IK in the ME. In the GBA_U variant, two keys are derived from Ks=CK||IK inside the UICC, namely K_S_extNAF and K_S_intNAF, but only K_S_extNAF is sent to the ME by the UICC.

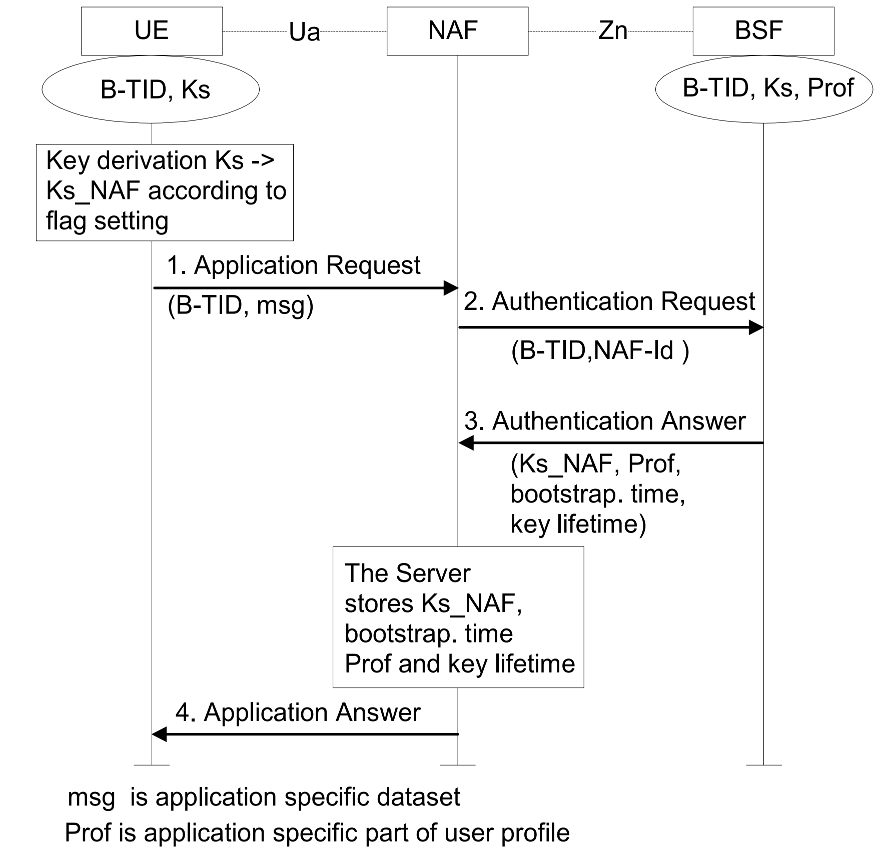
services is calculated from Ks and NAF ID. The NAF ID is the FQDN of the NAF concatenated with the protocol ID used over the Ua interface. In the case of GBA_U Ks shall not leave the UICC.

Figure 2: The Bootstrapping Procedure



- The last step allows the NAF (upon request from the UE with BTID) to get the K_S_(ext/int)NAF key from the BSF for use with the UE. In this step, NAF should provide NAF_ID to the BSF for key derivation.

Figure 3: Bootstrapping Usage Procedure


UE
Ua
NAF
Zn
BSF
B-TID, Ks
B-TID, Ks, Prof
Key derivation Ks -> Ks_NAF according to flag setting
1. Application Request
(B-TID, msg)
2. Authentication Request
(B-TID,NAF-Id)
3. Authentication Answer
(Ks_NAF, Prof, bootstrap. time, key lifetime)
The Server stores Ks_NAF, bootstrap. time Prof and key lifetime
4. Application Answer
msg is application specific dataset
Prof is application specific part of user profile

In order to allow the use of SIM cards or SIM Applications on the UICC for GBA, 3GPP also defines in 3GPP TS 33.220 – 'Generic Authentication Architecture (GAA); Generic Bootstrapping Architecture (Release 7)' [13] a specification for 2G GBA, which works as follows.

- The UE establishes a server-authenticated TLS tunnel with the BSF.
- The UE sends an HTTP request including identity information, to the BSF to initiate a bootstrapping run.
- The BSF retrieves 2G Authentication Vectors from the HSS.
- The BSF sends the RAND and a Nonce Ks-input to the UE.
- The UE runs the GSM authentication algorithms and responds to the BSF with RES, which is calculated from 2G credentials.
- BSF verifies the digest AKA response by doing the similar Hash. The Hash value is compared with the received one and if correct, sends an OK to the UE.
- Both UE and BSF compute Ks from Kc, SRES and Ks-input and a NAF-specific key K_SNAF is derived and sent to the NAF for use in Application layer services.

7.1.2 USES OF GENERIC BOOTSTRAPPING ARCHITECTURE

GBA has been defined as a generic mechanism for providing Application-layer keys for mobile services. To date, GBA is specified for use in MBMS security (see 3GPP TS 33.246 – '3G Security; Security of Multimedia Broadcast/Multicast Service (MBMS) (Release 7)' [15]), for Key Establishment between a UICC and a Terminal (see 3GPP TS 33.110 – 'Key Establishment Between a Universal Integrated Circuit Card (UICC) and a Terminal (Release 7)' [16]) and for the OMA BCAST Smart Card profile 'Service and Content Protection for Mobile Broadcast Services' [17], which uses the MBMS security architecture.

It is also specified in 3GPP TS 33.222 – 'Generic Authentication Architecture (GAA); Access to Network Application Functions Using Hypertext Transfer Protocol over Transport Layer Security (HTTPS) (Release 7)' [18] that GBA could be used to provide a shared symmetric key for the Pre-shared key variant of TLS (PSK-TLS). It may also potentially be used also for IMS services such as Presence, and to provide keying material for the Secure Channel between the UICC and Mobile Equipment.

7.1.3 ATTACKS: MOTIVATIONS AND METHODS

As the primary aim of GBA is to establish keys for Application security, a natural motivation for attacking GBA is to gain access to those keys and undermine the security of the Application in question (e.g. determining and revealing keys to be used in PSK-TLS sessions). Other Attacks may be envisioned too, however. For example, the GBA Server might collect and reveal the set of identities of NAFs the subscriber has accessed, in order to launch a linkability Attack against a subscriber, or may reveal RAND so that a malicious NAF may gain subscriber information from the BSF, or may be provided with Ks_NAF and then purport to the operator that it authenticated and offered service to a subscriber (the motivation for Attacks will depend on the particular Application of GBA and the trust and billing relationships among the Actors.). Denial-of-Service Attacks against a BSF, NAF or subscriber are also possibilities.

7.1.4 3GPP RECOMMENDATIONS FOR TRUSTED OPEN PLATFORMS

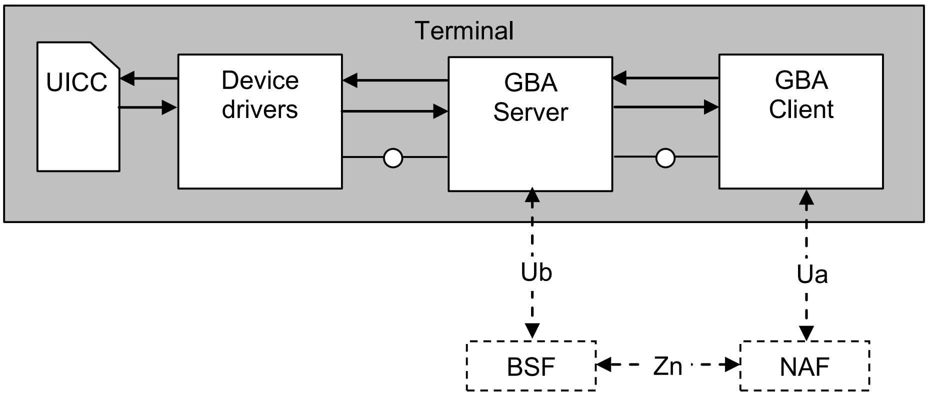
3GPP TR 33.905 – 'Recommendations for Trusted Open Platform (Release 7)' [19] proposes some recommendations for Trusted Open Platforms. An example Terminal architecture to implement GBA functionality on a UE is proposed in that document which includes:

- GBA Server: the functional entity in the UE communicating as a gateway with the SIM/USIM/ISIM Application in the UICC (to access GBA-related UICC APIs) and with the BSF during bootstrapping procedure. It can be viewed as a public API towards the GBA Client in the Terminal (thus presenting NAF-specific credentials to the GBA Client).

- GBA Client: gains access to the NAF-specific credentials from the GBA Server and communicates with the NAF over the Ua interface.

We distinguish between the functional components of GBA Server and GBA Client, however we do not mandate the particular architecture proposed in 3GPP TR 33.905 – 'Recommendations for Trusted Open Platform (Release 7)' [19]. For example, we do not preclude the possibility that the GBA Server and client may be implemented in the same software component.
Figure 4 depicts the different components in the Terminal that are needed for GBA use, known as the 'GBA architecture'. When a NAF server requests a GBA Client to authenticate itself with GBA credentials, the GBA Client communicates with the GBA Server for GBA credentials specific to that NAF.

Figure 4: GBA Related Modules in the Terminal



7.2 GENERIC BOOTSTRAPPING ARCHITECTURE ACTORS AND THEIR CONCERNS

Table 12 describes all Actors involved in the GBA mechanism, their main concerns and the trust link between them.

Table 12: Generic Bootstrapping Architecture Actors and Their Concerns

Actor	Main Concerns (in this context)
Operators (O)	Security of own services (e.g. MBMS or broadcast using OMA BCAST Smart Card profile) secured using GBA. Security of 3rd party services secured using GBA (operator wants to ensure that GBA is seen as a secure way for 3rd parties to offer services to users). Operator will trust the Terminal manufacturer for the GBA implementation in the ME and the UICC manufacturer for the GBA implementation in the UICC
Service Providers using GBA-secure services (SP)	Security of services secured using GBA (and keys derived from GBA). Service provider will trust the Terminal and UICC manufacturers and the operator for GBA implementation in the ME and UICC
UICC Manufacturer (UM)	Security of delivering GBA bootstrap data to the Terminal
Terminal Manufacturer (M)	Security of the implementation on the Terminal of services secured using GBA (and keys derived from GBA)
Application Provider (AP, provider of downloaded Application that uses GBA keys for security)	Security of the Application implementation on the Terminal performing services that are secured using GBA (and keys derived from GBA). Application Provider will trust the Terminal manufacturer

Actor	Main Concerns (in this context)
Attacker (A)	The Attacker tries to obtain the GBA-derived keys that are used to secure Applications on the Terminal. They will also attempt to exploit GBA vulnerabilities to launch Denial-of-Service and linkability Attacks
End User (U)	Protection of their personal data (e.g. identity, banking information). Avoid Denial-of-Service

7.3 GENERIC BOOTSTRAPPING ARCHITECTURE ASSETS

Table 13 lists Assets in the implementation of GBA that require protection. The Assets table describes various properties of GBA Assets that were used to derive the GBA requirements in section 7.4. The Assets described in table 13 are not all of the Assets in the UE, but only the GBA-related Assets.

7.3.1 GENERIC BOOTSTRAPPING ARCHITECTURE ASSETS TABLE

Table 13: Generic Bootstrapping Architecture Assets

Asset (A) or Asset Container (AC)	Description	Type	Owner†	Protection level /Level of Sensitivity	Security Properties Required
A.GBAn.xxx.y or AC.GBA.xxx.y with y=K for Key, y=D for Data, y=C for Code, y=H for Hardware			*See Actors List*	*Sensitive (S) Operational (O)*	*Integrity (I) Confidentiality (C) Authenticity (Au)*
A.GBA1: UICC GBA-related data					
A.GBA1.AUTN.D	Authentication value for user authentication to the network Authentication value which authenticates HSS/BSF to the UICC	Data	O	O	I, Au
A.GBA1.RAND.D	Random value for user authentication to the network	Data	O	S	I, Au

Asset (A) or Asset Container (AC)	Description	Type	Owner[†]	Protection level /Level of Sensitivity	Security Properties Required
A.GBA1.BTID.D	Identifier used to bind the subscriber identity to the keying material	Data	O	S	I, Au
A.GBA2: GBA-related Applications on UICC					
A.GBA2.SIM.C	Code of SIM Application	Code	O[*]	S	I
A.GBA2.USIM.C	Code of USIM Application	Code	O[*]	S	I
A.GBA2.ISIM.C	Code of ISIM Application	Code	O[*]	S	I
A.GBA3: Main UICC GBA-related keys					
A.GBA3.CK.K	Keying material	Key	O	S	I, Au, C
A.GBA3.IK.K	Keying material	Key	O	S	I, Au, C
A.GBA3.Ks.K	Keying material	Key	O	S	I, Au, C
A.GBA3.KsIntNAF.K	Resulting key	Key	O	S	I, Au, C
A.GBA3.KsExtNAF.K	Resulting key	Key	O	S	I, Au, C
A.GBA4: Main ME GBA-related keys, 2G GBA & GBA_ME					
A.GBA4.Kc.K	Keying material for 2G-GBA	Key	O	S	I, Au, C
A.GBA4.Ks.K	Keying material	Key	O	S	I, Au, C
A.GBA4.KsNAF.K	Resulting key	Key	O	S	I, Au, C
A.GBA5: GBA Server Assets (data, code)					
A.GBA5.NAFId.D	FQDN concatenated with protocol id	Data	O	S	I, Au
A.GBA5.KsInput.D	Random value for key derivation	Data	O	S	I
A.GBA5.CertBSF.D	BSF certificate for TLS connection with UE in 2G-GBA	Data	O	S	I
A.GBA5.BTID.D	BTID is used to bind the subscriber identity to the keying material	Data	O	S	I, Au

Asset (A) or Asset Container (AC)	Description	Type	Owner†	Protection level /Level of Sensitivity	Security Properties Required
A.GBA5.Server.API.C[7]	Code of the GBA Server interface	Code	M	S	I
A.GBA5.Server.Crit.C[8]	Code of the GBA Server	Code	M	S	I, Au
A.GBA5.IMSI/IMPI.D	Identifier used to define the subscribers	Data	O	S	I, Au
A.GBA5.Keylifetime.D	Key lifetime	Data	O	S	I, Au

Note:

† Denotes that the given set of owners of a specific Asset is only indicative.

7.3.2 Generic Bootstrapping Architecture Asset Grouping

In order to facilitate requirement understanding, the following groups of Assets are defined:

- **GBA Code Assets** group contains all ME GBA code Assets: A.GBA5.Server.API.C, A.GBA5.Server.Crit.C and A.GBA5.Server.Op.C
- **GBA Data Assets** group contains all ME GBA data Assets: A.GBA5.NAFId.D, A.GBA5.KsInput.D, A.GBA5.CertBSF.D and A.GBA5.BTID.D
- **GBA Key Assets** group contains all ME GBA key Assets: A.GBA4.Kc.K, A.GBA4.Ks.K and A.GBA4.KsNAF.K

7.4 Generic Bootstrapping Architecture Requirements

The Asset security properties referred to in the following requirement are defined in section 7.3.

[7] The access to the GBA Server API must be restricted to authorised Applications. The decision to grant the access to an Application is done by the Terminal manufacturer or the operator. The Terminal manufacturer can pre-configure a Security Policy on the Terminal

[8] This Asset is necessary to guarantee the GBA Server Authenticity and Integrity to the GBA Client.

7.4.1 GENERIC BOOTSTRAPPING ARCHITECTURE CORE REQUIREMENTS

All GBA Core Requirements are given in Chapter 3 “Trusted Environment Core requirements.

Table 14 summarises the GBA Core Requirements that apply to GBA Assets.

Table 14: Generic Bootstrapping Architecture Threat Grouping

Threat Group	1	2	3	4	5	6	7
Threats Assets	Hardware Modules Used for Accessing Memories	CLCD Used for Displaying Memories and Interfering With Displayed Data	Bypass Security by Removal of Battery Power or Removal of External Memory Card	Attack by Replacement of Flash When Power is Off (Pre-Boot)	Extract Secret via Bus Monitoring (Hardware Probes)	Mod Chip Attacks on Data in External RAM	Attack by Replacement of Flash When Power is On (Post-Boot)
A.GBA5.Server.API.C A.GBA5.Server.Crit.C A.GBA5.Server.Op.C A.GBA5.NAFId.D A.GBA5.KsInput.D A.GBA5.CertBSF.D A.GBA5.BTID.D	2 (C)	2(C)*	1, 2	1, 2	2(C)*	-	-
A.GBA4.Kc.K A.GBA4.Ks.K A.GBA4.KsNAF.K	2	2 (C)	1, 2	1, 2	2 (C)	2	2

Notes:

(C) Denotes the fact that only Confidentiality of these Assets needs to be defended against this Threat group.

(C)* Denotes the fact that these Assets may not require Confidentiality protection, but if they do then this Threat Group must be defended against.

7.4.2 GENERIC BOOTSTRAPPING ARCHITECTURE SPECIFIC REQUIREMENTS

These requirements are to be implemented in accordance with 3GPP TR 33.905 – ‘Recommendations for Trusted Open Platform (Release 7)’ [19] (i.e. requirements ATE-GBA-1360 to ATE-GBA-1420) are made with respect to recommendations 1 to 5 and other requirements only in section 4.1.1 of 3GPP TR 33.905 – ‘Recommendations for Trusted Open Platform (Release 7)’ [19].

REQ. ID	REQUIREMENT	PROFILE 1	PROFILE 2
ATE-GBA-1320	If the GBA mechanism is used by an Application residing in a particular TEE then the components of the GBA Server responsible for GBA security SHALL be executed either from that TEE or from a TEE of an equivalent or higher Profile	✓	✓
ATE-GBA-1330	The Authenticity and Integrity of the GBA Server and of the cryptographic software related to GBA in the ME SHALL be verified by the FSB	✓	✓
ATE-GBA-1340	If persistent, A.GBA4 keys related to 2G GBA and GBA_ME mechanisms SHALL be stored using SST	✓	✓
ATE-GBA-1350	When key lifetime (A.GBA5.Keylifetime.D) has expired or UICC is changed, the GBA Key Assets SHOULD be removed from the cached store	✓	✓
ATE-GBA-1360	The platform SHALL allow only authorised Applications to access the UICC for GBA purposes, as requested in 3GPP TR 33.905 – 'Recommendations for Trusted Open Platform (Release 7)' [19]	✓	✓
ATE-GBA-1370	The platform SHALL restrict access to the GBA Master Secret of the GBA Server (that means A.GBA4.Ks.K stored in the GBA Server in case of 2G-GBA or GBA_ME), as requested in 3GPP TR 33.905 – 'Recommendations for Trusted Open Platform (Release 7)' [19]	✓	✓
ATE-GBA-1380	The platform SHALL control general access to the GBA Server so that an unauthorised Application is not able to retrieve any NAF specific GBA credentials from the GBA Server (e.g. A.GBA4.KsNAF.K), as requested in 3GPP TR 33.905 – 'Recommendations for Trusted Open Platform (Release 7)' [19]	✓	✓

REQ. ID	REQUIREMENT	PROFILE 1	PROFILE 2
ATE-GBA-1390	The ACP (Access Control Policy) in the UE controls whether an Application is authorised to have access to NAF-specific GBA credentials. This ACP SHALL be provisioned/updated in the ME by the Terminal manufacturer, or in the UICC or in the ME by the operator, as requested in 3GPP TR 33.905 – 'Recommendations for Trusted Open Platform (Release 7)' [19]	✓	✓
ATE-GBA-1400	The manufacturer or the operator only SHALL provision and modify the ACP to grant access to the GBA Server and sensitive GBA Assets, as requested in 3GPP TR 33.905 – 'Recommendations for Trusted Open Platform (Release 7)' [19]	✓	✓
ATE-GBA-1410	The ACP MAY allow the End User to grant access to the GBA server and sensitive GBA Assets to Applications resident on the ME, provided that such access is not prevented by Terminal Manufacturer or Operator policy	✓	✓
ATE-GBA-1420	The GBA Key Asset exchanges between the UICC and the ME SHALL be sent through a Secure Channel (as defined in chapter 10)	✓	✓

8 RUN-TIME INTEGRITY CHECKING

This chapter describes the security requirements related to Run-time Integrity Checking (RIC) for two different Profiles.

An introduction to RIC is provided in section 8.1.

A use cases section is provided in section 8.2.

The list of Run-time Integrity Checking Actors and Their Concerns is provided in section 8.3.

The list of RIC Assets is defined in section 8.4.

RIC Threats are defined in section 8.5.

The RIC security requirements are defined in section 8.6.

8.1 RUN-TIME INTEGRITY CHECKING DEFINITION

Following a Secure Boot, before the initialisation of any operating system or system operating code, and then during normal operation, the Integrity of principal hardware and software components must be determined and maintained. Run-time monitoring and detection of unauthorised changes of state can provide protection against Attacks on many Asset classes.

Run-time Integrity Checking (RIC) is a function that may be initiated as an integral part of a Secure Boot, or from a known initial state within a Trusted Execution Environment, and which performs run-time monitoring of hardware- and software-level resources in order to provide detection of unauthorised changes in platform Integrity. The Integrity checker function must be a component whose Integrity has been ensured. In the case of multiple Execution Environments, the RIC may need to be capable of performing Integrity checks across all of the appropriate Environments.

Certain ME Assets are particularly security critical and would include:

- Kernel code
- OS and platform configuration data such as interrupt vectors
- CPU and other ME hardware

Also, certain ME software and data components should not be modified after a Secure Boot. These would include:

- Elements of the kernel code
- Trusted Root Public Keys
- Network and subscriber identify information

Additionally, certain maintenance functions should not be invoked during normal operation. These include:

- Debug ports or test functionality, e.g. JTAG or scan functions

Although the protection of both static and dynamic code and data is desirable, static components may be the easiest Assets for the malicious Attacker to exploit and yield the largest impact. Hence protection of static Assets, particularly the ones which are security critical, is of specific importance.

Upon detection of a loss of Integrity, a mechanism that is itself protected against the attacking exploit should be provided to signal and respond to this failure. The resulting action taken should minimise the impact of the Integrity failure. The RIC is the mechanism intended to provide this robust policing of the run-time Environment.

The corresponding action taken upon failure detection should be defined for each Asset within the appropriate requirements function. Additionally, a default action (e.g. Terminal reset) for critical failures (e.g. kernel code Integrity loss) may be defined in future versions of this document.

8.2 RUN-TIME INTEGRITY CHECKING USE CASES

Run-time Integrity Checking (RIC) is a security process that initiates after the end of the Flexible Secure Boot (FSB) process. In general, RIC performs a housekeeping function, during the normal run-state of the ME, monitoring the contents of program and specific data memory and determining whether the monitored memory has been modified in some illegal way.

The RIC function is initiated by the completion of FSB. FSB may initialise certain data, code, hardware or other Assets required by the RIC in order for it to perform its tasks.

Once active, the RIC has a limited number of methods to deal with changes in monitored memory, up to and including a forced reset of the ME. If the system has become permanently corrupted, the repeated forced resets will have the consequence of making the Terminal non-functional.

8.2.1 KEY OPERATOR USE CASES

The key operator use cases are listed below in priority order.

RIC Detects Malicious Attack on System Kernel or Other Critical Code

- An Attacker uses an operating system exploit to modify a system software component in order to bypass the Application Security Framework (ASF) trust tier enforcement and allow a malicious Application to execute without checking in order to gain access to sensitive user information (software modification, class of software Attacks).
- The RIC, through periodic monitoring or triggered after a particular event, such as application execution or service request, detects an

Integrity failure of the system kernel or other critical code components. It then indicates a system Integrity failure to the RIC error handler and forces the default condition of reboot. If provided for, the RIC may store a record of the failure and other auxiliary information in non-volatile memory for later forensic analysis.

- A system reboot is performed in which the system software is authenticated and reloaded from non-volatile memory allowing normal usage. The logged error condition (if enabled) allows the system software to report the condition and allow a higher authority to monitor for future occurrences.

RIC Detects Compromise of RIC Function

- The RIC determines that RIC functionality has been compromised in some manner.
- The RIC then forces the default condition of reboot. If provided for, the RIC may store a record of the failure and other auxiliary information in non-volatile memory for later forensic analysis.
- A system reboot is performed in which the system software is authenticated and reloaded from non-volatile memory allowing normal usage. The logged error condition (if enabled) allows the system software to report the condition and allow a higher authority to monitor for future occurrences.

Auxiliary System Software Components Legally Loaded by Kernel

- The system has minimal RAM and auxiliary code blocks are loaded or removed from RAM depending on ME current need.
- RIC detects the Loading of a legal code block and calculates a Hash value for that block, then compares that Hash value to the Hash previously generated by a trusted authority and stored in a protected manner.
- If the Hashes do not compare, the RIC may alert the system error handler or otherwise prevent the corrupted code block from executing.

Changes to SIM or IMEI Protected Content

- The RIC detects a change to SIM or IMEI protected content.
- The RIC may check the SIM/IMEI access log for authorised access.
- If the SIM/IMEI access log does not indicate authorised access, then the RIC alerts the system error handler to the changes.

8.3 Actors And Their Concerns

This section defines the interests of Actors on the functionality of the platform. These interests may impact the behaviour of the RIC.

Table 15: Run-time Integrity Checking Actors and Their Concerns

Actor	Main Concerns (in this context)
Operators (O)	Operators will want to run Applications and store certificates and Device data within an Environment that verifies the Integrity of associated hardware and software components
Other Service Providers (SP)	Service providers will want to download Applications within an Environment that verifies the Integrity of associated hardware and software components
Content Providers (CP)	Content providers will want any sensitive data to be protected from malicious Attacks that may result from taking advantage of a loss of Integrity of hardware and software components
End User (U)	The end-user will want the user Assets to be protected from malicious Attacks that may result from taking advantage of a loss of Integrity of hardware and software components
Manufacturer (M)	The manufacturer implements the RIC function and its real-time behaviour and any hardware that can be used for test/debug. The manufacturer will want to run Applications and store Device data within an Environment that verifies the Integrity of associated hardware and software components

Actor	Main Concerns (in this context)
Attacker (A)	An Attacker may want to compromise the Terminal Integrity in order to either gain direct access to Assets or to enable undetected component modifications that will lead to further Attacks. Run-time Integrity checks help to guarantee that an Attacker who is able to compromise the Terminal Integrity will not be able to capitalise on that breach

8.4 RUN-TIME INTEGRITY CHECKING ASSETS

The following Assets in the implementation of a RIC require protection. Table 16 describes various properties of RIC Assets that were used to derive the RIC requirements in section 8.4.

Hardware components that monitor and provide an indication of system Integrity and in particular failure detection status, may be included in the A.RIC.STS.H function in order to provide a security system level indication of correct operation.

It is currently envisaged that the RIC could be implemented either as a hardware function (A.RIC.RIC.H) as software (A.RIC.RIC.C), or as some combination of both. These possibilities are reflected in table 16. In both cases, the RIC verification data would be defined as A.RIC.RIC.D. The RIC may use public, symmetric or Private Keys (A.RIC.KEYS.K) to verify Integrity of code and data or to protect RIC Assets.

The Assets of the primary RIC are established and made secure through a preceding Secure Boot mechanism. The Assets of any secondary RICs are controlled and authenticated by the primary RIC. There may be more than one level of RIC beyond the primary, but this document does not go beyond a secondary RIC, nor does it attempt to define how many secondary RICs may exist.

The motivation for doing this is to permit a hierarchical implementation of run-time Integrity checking. The “primary” Assets relate to the run-time Integrity checking function that is initialised at boot time, and continues to perform its function throughout the subsequent power-on session. The Integrity and Availability of the Assets of the primary run-time Integrity function are inherently protected. Some of the code and data checked by the primary run-time Integrity checking function may be the Assets of a secondary run-time Integrity function. The distinguishing characteristic of a secondary run-time Integrity checking mechanism is that the Integrity and Availability of its Assets need not be inherently protected, since they are protected by the primary run-time Integrity function.

8.4.1 RUN-TIME INTEGRITY CHECKING ASSETS TABLE

Table 16: Run-time Integrity Checking Assets

Asset (A) or Asset Container (AC)	Description	Type	Owner†	Protection Level /Level of Sensitivity	Security Properties Required
A.RIC.xxx.y *with y=K for Key, y=D for Data, y=C for Code, y=H for Hardware*		*Key (K) Data (D) Code (C) Hard ware(H)*	*See Actors List*	*Sensitive (S) Operational (O)*	*Integrity (I) Confidentiality (C). Authenticity (Au)*
A.RIC.RIC.H1	Primary Integrity checking function	H or C	M	S	I
A.RIC.RIC.D1	Primary Integrity checking function verification data (May include reference Hashes, initialisation values, parameters, etc.)	D	M	S	I, Au
A.RIC.RIC.H2	Secondary Integrity checking function	H or C	M	S	I
A.RIC.RIC.D2	Secondary Integrity checking function verification data (May include reference Hashes, initialisation values, parameters, etc.)	D	M	S	I, Au
A.RIC.KEYS.K	RIC key material	K	M	S	I, C**
A.RIC.INI.C1	Primary initialisation function	C	M	S	I

Asset (A) or Asset Container (AC)	Description	Type	Owner†	Protection Level /Level of Sensitivity	Security Properties Required
A.RIC.INI.C2	Secondary initialisation function	C	M	S	I
A.RIC.ERR.C1	Primary Error Handler	C	M	S	I
A.RIC.ERR.C2	Secondary Error Handler	C	M	S	I
A.RIC.APP.D1	Primary Application Code/Data segment to be checked	C D	O M SP	S	I
A.RIC.APP.D2	Secondary Application Code/Data segment to be checked	C D	O M SP	S	I
A.RIC.STS.H1	Primary Monitored Hardware status	H	M	S	I
A.RIC.STS.H2	Secondary Monitored Hardware status	H	M	S	I

Notes:

† Denotes that the given set of owners of a specific Asset is only indicative.

** Denotes the fact that the Confidentiality of these keys will depend on whether these keys are symmetric, Private or Public.

8.4.2 RUN-TIME INTEGRITY CHECKING ASSET GROUPING

In order to facilitate requirement understanding, the following groups of Assets are defined:

- **Run-Time Integrity Code Assets group** contains all code Assets: A.RIC.RIC.H1 (if Code), A.RIC.RIC.H2 (if Code), A.RIC.INI.C1, A.RIC.INI.C2, A.RIC.ERR.C1, A.RIC.ERR.C2, A.RIC.APP.D1 (Application code) and A.RIC.APP.D2 (Application code)
- **Run-Time Integrity Data Assets** group contains all data Assets: A.RIC.RIC.D1, A.RIC.RIC.D2, A.RIC.APP.D1 (Application data) and A.RIC.APP.D2 (Application data)
- **Run-Time Integrity Hardware Assets** group contains all hardware Assets: A.RIC.RIC.H1 (if Hardware), A.RIC.RIC.H2 (if Hardware), A.RIC.STS.H1 and A.RIC.STS.H2

- **Run-Time Integrity Key Assets** group contains the key Asset: A.RIC.KEY.K

8.5 RUN-TIME INTEGRITY CHECKING THREATS

One of the most significant Threats to a Handset is the potential for system intrusion though malicious code that might masquerade as valid system or Application code. The ability to establish a temporary root privilege and then escalate that privilege to a permanent one opens the platform to various exploits ranging from subscriber compromise to network takedown. Once this privilege is secured, the platform can be hijacked by “bot” and “zombie” Attacks to take control of the platform; “Pharming/Phishing” Attacks that capture confidential user data and Denial-of-Service Attacks against the network. Run-time Integrity checking (RIC) provides a flexible and well-vetted method to ensure that executables accepted into the Handset system are what they claim to be, and have not been tampered with or replaced by malicious code.

RIC can be implemented in a number of ways; a common one is for the executable developer to generate a Hash of the executable and provide access to that Hash value by an RIC process within the ME.

8.6 RUN-TIME INTEGRITY CHECKING REQUIREMENTS

Profiles 1 and 2 define two sets of RIC security requirements. The requirements for each of these Profiles define certain required functionalities of the RIC and also the scope of the Attacks to which RIC Assets should be resistant. The two Profiles offer incremental levels of security (that is, Profile 2 offers stronger security than Profile 1). RIC could need to protect a normal EE or TEE; in this case the ME may (or may not) have the ability to download ringtones, simple games, and Applications that might be generated by either trusted or untrusted 3rd parties. In addition, the Terminal critical codebase will likely execute from RAM and may be upgradeable via FOTA or some other mechanism. It is the combination of these two factors (system software modifiability and the ability to import foreign data or Applications) that justify the use of RIC even in a typical EE. As the functionality of the ME increases, the potential for Threats increases as well. The RIC could also protect a TEE to increase the resistant and protection against identified Threats.

8.6.1 RUN-TIME INTEGRITY CHECKING CORE REQUIREMENTS

The Asset security properties referred to in the following requirements are defined in Section 8.3.

In addition to TR1 Core Requirements defined in Chapter 3 “Trusted Environment Core requirements”, further core requirements are defined for RIC in this section.

REQ. ID	REQUIREMENT	PROFILE 1	PROFILE 2
ATE-RIC-1430	If the RIC keys require Confidentiality, the Confidentiality of RIC Key Assets SHALL be defended against the Threats contained in Threat Group 5 “Extract Secret via Bus Monitoring (Hardware Probes)”		✓
ATE-RIC-1440	The security properties of RIC Code, Data and Key Assets SHALL be defended against the Threats contained in the Threat Group 6 "Mod Chip Attacks on Data in External RAM”		✓
ATE-RIC-1450	The security properties of RIC Code, Data and Key Assets SHALL be defended against the Threats contained in the Threat Group 7 “Attack by Replacement of Flash When Power is On (Post-Boot)”		✓

Table 17 summarises the RIC core requirements that apply to RIC Assets.

Table 17: Run-time Integrity Checking Threat Grouping

Threat Group	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Threats Assets	Hardware Modules Used for Accessing Memories	CLCD Used for Displaying Memories and Interfering With Displayed Data	Bypass Security by Removal of Battery Power or Removal of External Memory Card	Attack by Replacement of Flash When Power is Off (Pre-Boot)	Extract Secret via Bus Monitoring (Hardware Probes)	Mod Chip Attacks on Data in External RAM	Attack by Replacement of Flash When Power is On (Post-Boot)
A.RIC.INI.C1 A.RIC.INI.C2 A.RIC.ERR.C1 A.RIC.ERR.C2 A.RIC.RIC.D1 (code) A.RIC.RIC.D2 (code) A.RIC.RIC.H1 (code) A.RIC.RIC.H2 (code)	2	2 (C)*	1, 2	1, 2	2 (C)*	2	2
A.RIC.APP.D1 A.RIC.APP.D2 A.RIC.RIC.D1 (data) A.RIC.RIC.D2 (data)	2	2 (C)*	1, 2	1, 2	2 (C)*	2	2
A.RIC.KEY.K	2	2 (C)*	1, 2	1, 2	2 (C)*	2	2
A.RIC.STS.H1 A.RIC.STS.H2 A.RIC.RIC.H1 (Hardware) A.RIC.RIC.H2 (Hardware)	-	-	-	-	-	-	-

Note:

(C)* Denotes the fact that these Assets may not require Confidentiality protection, but if they do, then this Threat group must be defended against.

8.6.2 RUN-TIME INTEGRITY CHECKING SPECIFIC REQUIREMENTS

REQ. ID	REQUIREMENT	PROFILE 1	PROFILE 2
ATE-RIC-1460	The security properties of the primary RIC including any data and code Assets SHALL be intrinsically guaranteed OR The security properties of the primary RIC at its initiation including any data and code Assets SHALL be guaranteed by the FSB	✓	✓
ATE-RIC-1470	The Integrity and Authenticity of verification data (A.RIC.RIC.D) and any verification keys (A.FSB.VER.K) used by the RIC SHALL be determined before use unless these data and keys are stored in Integrity-Protected Memory that is write-accessible only to the RIC and FSB functions	✓	✓
ATE-RIC-1480	Modifiable primary RIC Assets SHALL be modifiable only through the FSB or other mechanism of equivalent security level	✓	✓
ATE-RIC-1490	The primary RIC SHALL operate: - Either independently of any OS or EE or run-time modifiable software elements to be checked OR - In collaboration with software parts which have been checked for Integrity prior to the execution of the primary RIC	✓	✓
ATE-RIC-1500	The primary RIC SHALL have read access to memory spaces containing data and code to be Integrity checked	✓	✓

REQ. ID	REQUIREMENT	PROFILE 1	PROFILE 2
ATE-RIC-1510	The primary RIC SHALL use Integrity-guaranteed reference measurements to validate the contents of applicable memory space	✓	✓
ATE-RIC-1520	The primary RIC SHALL recognise modifications to the monitored memory space and report those changes to an RIC error handler	✓	✓
ATE-RIC-1530	The primary RIC error handler SHALL have a mechanism to force a system reboot if specified Integrity parameters have been violated or are out of specification	✓	✓
ATE-RIC-1540	The secondary RIC SHALL have a mechanism to invoke a system error handler if specified Integrity parameters have been violated or are out of specification	✓	✓
ATE-RIC-1550	The secondary RIC error handler MAY have the ability to force reloads or resets of Application-level code if specified security parameters have been violated or are out of specification	✓	✓
ATE-RIC-1560	For primary RICs implemented in software, secured RIC Assets SHALL NOT be modifiable by EEs that are less trusted than the EE that contains the primary RIC	✓	✓
ATE-RIC-1570	It SHALL NOT be possible to disable the primary RIC	✓	✓
ATE-RIC-1580	The Availability, Integrity and, if applicable, the Authenticity of the secondary RIC Code, Data and Key Assets SHALL be checked by the primary RIC	✓	✓

REQ. ID	REQUIREMENT	PROFILE 1	PROFILE 2
ATE-RIC-1590	If the RIC reference measurements are obtained by measuring the target code at load time, then the Integrity of this target code SHALL be checked before its Execution unless it is stored in Integrity-Protected Memory	✓	✓
ATE-RIC-1600	The initialisation of reference measurements used by the secondary RIC to check code segments loaded during ME run-time SHALL be protected by the Primary RIC or other mechanisms of an equivalent or greater trust level	✓	✓
ATE-RIC-1610	The RIC MAY have a mechanism to store signed logs of event data in non-volatile memory, write-accessible only by the FSB or RIC, for later retrieval by the Terminal manufacturer	✓	✓
ATE-RIC-1620	The Assets to be checked by the RIC, the frequency of these checks, and any action to be taken SHOULD be reconfigurable (i.e. A.RIC.RIC.D1 and A.RIC.RIC.D2 SHOULD be reconfigurable) by the manufacturer	✓	✓

8.7 INFORMATIVE NOTES FOR RIC IMPLEMENTATIONS

While specifically not requirements, the following notes are intended to be informative and to add to the understanding of the RIC functionality. The following represents a non-inclusive list of what is out of scope for the RIC function, as well as discussing some significant considerations that should be made when architecting an ME.

- No RIC is responsible for detecting unauthorised fetches or reads of monitored memory space. While there are potential security risks from Applications gaining access to data that is not their own, it is out-of-scope for the RIC to monitor these events.
- Containing the primary RIC or Assets within non-volatile storage opens the platform to risks from power interruption. Upon power loss, it may

be appropriate to reinitialise via the FSB any primary RIC or Assets that are contained or operate within non-volatile storage.

- The RIC has no responsibility to detect any of the cloning Attacks listed in 'Security Threats on Embedded Consumer Devices' [2].
- It is important to note that security functions like the RIC should not have a significant impact upon processor performance, memory speed, etc. While not specified, attention should be paid to ensuring that the primary RIC takes no more than an appropriate percentage of system resources during normal background monitoring (i.e. where no errors are detected). An alternative method may be to quantify usage of system resources through a formula that includes processor speed, size of memory to be monitored, etc.
- The RIC is an essential part of a secure platform; however, the physical design of the RIC can compromise any security solution if appropriate steps are not taken:
    - All primary RIC security Assets and mechanisms should be contained within permanently attached physical devices internal to the ME, significantly reducing the Threat from physical "Man-In-The-Middle" Attacks.
    - While the operating system and/or system code does not have visibility of the mechanism and components of the RIC, there may be value in allowing the operating system to "know" whether or not the RIC is active, through the use of a "status flag" mechanism, accessible as read-only by BIST.
    - Sometimes it may be useful, especially in forensic analysis, to gain access to the mechanism and functionality of the RIC through a secure debug mechanism (e.g. JTAG).
- A RIC cannot be guaranteed to protect things upon which it is dependent. If implemented in a TEE, the RIC cannot be expected to defend that TEE against attacks which could compromise the RIC itself. A robust RIC is therefore one that has minimal dependencies on other code.

9 SECURE ACCESS TO USER INPUT/OUTPUT FACILITY

This chapter describes the security requirements related to the Secure User Input/Output (SUIO) facility for Profile 1. There are no Profile 2 requirements for the Secure User Input/Output facility specified in this version of the document. Such Profile 2 requirements for the SUIO facility may be defined in future versions.

A use cases section is provided in section 9.1.

An introduction to the Secure User Input/Output Facility is provided in section 9.2.

The list of Actors and Actors' concerns is provided in section 9.3.

The list of Secure User Input/Output Assets and the Secure User Input/Output Asset Grouping is defined in section 9.4.

The security requirements are defined in section 9.5

9.1 SECURE USER INPUT/OUTPUT KEY OPERATOR USE CASES

The key operator use cases are listed below in priority order.

9.1.1 USE CASE GROUPING 1 – SECURE I/O FOR SIM CARD APPLICATIONS

9.1.1.1 SIM Application Access via Web Browser

Laurent wants to use a service stored in his SIM card which can be accessed on the web server stored in his SIM, via a browser stored in his Handset. His SIM Application is able to use I/O capacities offered by the Handset, such as keyboard, display and audio drivers.

Some abuse cases

- Man-In-The-Middle Attack: e.g. the interception of sensitive data between the client and the server
- Replay-Attack
- Cookie spying

9.1.1.2 Proximity Payment

Caroline wants to use the payment Application stored in her UICC via the wallet Application stored in her Handset. This Application allows Caroline to activate a proximity payment via contactless connectivity to a contactless EFTPOS (Electronic Fund Transfer Point Of Sale).

There are three steps required for correct payment:

- Caroline selects her payment Application using the wallet Application in her Handset.
- She validates the selected items and the amount of the transaction.

- She enters a password (for example, for pre-paid services) or a credit-card number.

Some abuse cases

- A Man-In-The-Middle Attack is performed by a Handset Application against the link between a web browser and SIM in order to display false information to the user.
- A Man-In-The-Middle Attack is organised by a Handset Application which stores secrets entered by the user when she wants to enter sensitive information.
- The server is faked.
- Unfeasibility of WYSIWYS (What You See Is What You Sign).

9.1.2 USE CASE GROUPING 2 – DIGITAL SIGNATURE

David wants to sign data using an electronic method. This can be used in various situations, such as when sending an email or for approving a digital contract.

Some abuse cases

- Replay Attack, Man-In-The-Middle Attack
- Modification of the signature
- Unfeasibility of WYSIWYS (What You See Is What You Sign)

9.1.3 GENERAL ABUSE CASES

9.1.3.1 Key Logger

A Key Logger is a program that runs in background that captures the user's keystrokes. They can be very hard to detect, if not listed in the process list. Key Loggers are able to detect and store sensitive information such as login details, passwords and credit card numbers.

9.1.3.2 Tampering Inputs

Tampering with inputs allows an Attacker to modify the input events. This is critical because graphics systems allow developers to create and send UI events. Moreover, graphics systems usually manage the display of the user's input by themselves and send the value of this data to an Application through software call-back functions. Applications only receive the value of the data entered, which they consider to be analogous to the data that is displayed on the screen.

If a malicious Application could hook call-backs in order to receive the events sent by the graphics system after a user keystroke, then it would be also possible to change the data and send fake keystrokes to the Application. The user would not detect any corruption because the data displayed on screen would be the same as that which he entered.

9.1.3.3 Faked Data

Tampering with data to be displayed means modifying the data an Application sends to the graphics system. For example, in a software management use case, a malicious Application could intercept the certificate data the Application sent to be displayed and replace it with fake data.

Tampering with displayed data can be performed in different ways, depending on the system architecture and tolerance.

9.1.3.4 Driver Hooking

An Application or driver can hook the API calls, by overloading part or all functions of the graphic system with custom functions. The hook can then modify the data it receives and then send modified data to the real function. The issues created by this Attack are similar to the ones for faked data.

9.1.3.5 Social Engineering Attack/Phishing

By simulating the normal running of an Application (which can be a faked one or a new one which simulates a trusted Application), a malicious Application can obtain sensitive information such as passwords or PIN codes and send this sensitive information to other Applications or external servers for future reuse.

Such social engineering/Phishing Attacks can be prevented using dedicated UIO hardware, although this is not required by this document. An Application with the support of a secure UIO can defend against such Attacks (e.g. by displaying data that the user can associate with the genuine Application).

9.2 SECURE USER INPUT/OUTPUT INTRODUCTION

User Input/Output (UIO) is a facility enabling interactions with the User. It consists of:

- Output e.g.:
  - Display e.g. text, graphics, video
  - Audio e.g. speech, music
  - Control: e.g. LEDs, labels
- Input e.g.:
  - Keypad e.g. text, prompts
  - Touch Screen e.g. text, prompts, signatures
  - Microphone e.g. speech, prompts
  - Biometric e.g. fingerprints

Secure access to User Input/Output is a facility (called hereafter SUIO) used by value-added secure services (called hereafter Applications). This includes Broadcast Service Protection agents, DRM, GBA, Device Management, corporate Applications and data security, m-commerce, and by security-demanding Applications. The SUIO facility ensures that the I/O data of such security-demanding Applications is defended against Application level Attacks by other Applications or malware.

The SUIO is used to ensure that relevant Application Assets' security properties are valid in the following cases:

- when they are input from the user
- when they are output to the user

Examples of using the SUIO are to:

- protect against software Attacks on decoded DRM-protected content when it is output to the user
- securely collect sensitive user information provided by means of user prompts
- aid the implementation of anti-Phishing mechanisms

9.3 SECURE USER INPUT/OUTPUT ACTORS AND THEIR CONCERNS

Table 18 describes all Actors involved in the SUIO, their main concerns, and the trust link between them.

Table 18: Secure User Input/Output Actors and Their Concerns

Actor	**Main Concerns (in this context)** *(role of the Actor, responsibilities, trust relationship with the other Actors)*
Operators (O)	Protection of their revenue from unauthorised access to services. Protection of content and content usage rights obtained from Content Providers. Protection of ME Assets (e.g. User authentication mechanisms) and of Application-sensitive Assets. Maintaining user trust in their service. Operators will trust the UICC manufacturer and/or Terminal manufacturer and will want some control over the access other service providers are granted to UIs

Actor	**Main Concerns (in this context)** *(role of the Actor, responsibilities, trust relationship with the other Actors)*
Other Service Providers (SP)	Service providers will trust the UICC manufacturer and/or Terminal manufacturer although they will not necessarily trust the Operator
Content Providers (CP)	Protection of their content and content usage rights. Content providers will trust the UICC manufacturer, Terminal manufacturer and/or the Operator
End User (U)	Protection of their personal data (identity, banking information, sensitive contacts, etc). End users will expect the operator, UICC manufacturer, and Terminal manufacturer to protect their Assets
Terminal Manufacturer (M)	Ensuring Terminals cannot be hacked. Ensuring revenue and content is not lost from their Terminals. Maintain user trust in their Terminals
UICC Manufacturer (UM)	Maintaining user trust in their UICC
Attacker (A)	Obtaining unauthorised access to Application Assets as they are being output to the user. Obtaining unauthorised access to user inputs (e.g. Phishing), then to user data or Application Assets (keys, code and data). Successful Replay Attack (e.g. to gain indefinite rights on User-protected actions or on DRM-protected content)

9.4 SECURE USER INPUT/OUTPUT ASSETS

The SUIO receives Sensitive Objects (A.SUIO.DO.D) from Applications and securely outputs them to a hardware user interface or it receives them from a hardware user interface and securely inputs them to Applications. In both cases, it does this in such a way that it maintains the security properties of these Assets. Such objects can represent any information, but are treated as opaque data by the SUIO.

The SUIO maintains general data (A.SUIO.D) used by the SUIO Code (A.SUIO.C) for example, to maintain isolation between sensitive Assets belonging to different Applications, or can represent a Security Policy that defines an Access Control policy and security properties for Assets.

The SUIO inputs and outputs the Sensitive Objects from and to appropriate input/output interface hardware, using corresponding hardware peripheral device drivers (A.SUIO.IO.C/D), according to appropriate Application access rights.

9.4.1 SECURE USER INPUT/OUTPUT ASSET TABLE

The following Assets in the implementation of Secure UIO require protection. The Assets table describes various properties of SUIO Assets that were used to derive the SUIO requirements.

Table 19: Secure User Input/Output Assets

Asset (A) or Asset Container (AC)	Description	Owner†	Protection Level /Level of Sensitivity	Security Properties Required
A.SUIO.xxx.y or AC.SUIO.xxx.y with y=K for Key, y=D for Data, y=C for Code, y=H for Hardware		*See Actors List*	*Critical (C), Sensitive (S), Operational(O)*	*Integrity (I), Confidentiality (C), Authenticity (Au), Non-Replay (NR)*
A.SUIO.DO.D	Sensitive object provided to the UI by the Application or by the UI to the Application calling it. It can represent any information, but is treated as opaque data by the SUIO. Requirements on this Asset will apply when this Asset is under the control of the SUIO	M UM SP O U	S	I, C**, Au, NR**
A.SUIO. C	Code of the Secure UI software components	M	S	I, Au

Asset (A) or Asset Container (AC)	Description	Owner†	Protection Level /Level of Sensitivity	Security Properties Required
A.SUIO.D	Data required by A.SUIO.C (e.g. access rights)	M	S	I[9], C*, Au
A.SUIO.IO.C	Code of the Secure UI software components for managing secure input/output devices (SUIO secure input/output drivers)	M	S	I, Au[10]
A.SUIO.IO.D	Data of the Secure UI SW components for managing secure input/output devices (SUIO secure input/output drivers)	M	S	I, C**, NR**

Notes:

† Denotes that the given set of owners of a specific Asset is only indicative.

* Denotes the fact that these Assets may or may not require Confidentiality, depending on the SUIO implementation and the purpose of the Asset.

** Dependant on the requirements of the specific Asset instance.

9.4.2 ASSETS GROUPING

In order to facilitate requirement understanding, the following groups of Assets are defined:

- **SUIO Code Assets** group contains all SUIO code Assets: A.SUIO. C and A.SUIO.IO.C.
- **SUIO Data Assets** group contains all SUIO data Assets: A.SUIO.DO.D, A.SUIO.D and A.SUIO.IO.D.

9.5 SECURE USER INPUT/OUTPUT REQUIREMENTS

9.5.1 SECURE USER INPUT/OUTPUT CORE REQUIREMENTS

The Asset security properties referred to in the following requirements are defined in Section 9.4.1.

[9] Depends on the level of sensitivity of the data of the Application calling the SUIO.

[10] This middleware code is assumed to authenticated as part of the Flexible Secure Boot process.

Since we only define a single Profile for SUIO (Profile 1), only the Profile 1 requirements defined in Chapter 3 “Trusted Environment Core requirements” apply to SUIO Assets. In addition, further core requirements are defined for Secure User Input/Output below.

Note: In addition, since SUIO Data Objects (A.SUIO.DO.D) are usually temporary objects which would be lost at power down, defence against Threat Groups 3 and 4 may not be applicable for these Assets.

REQ. ID	**REQUIREMENT**	**PROFILE** **1**
ATE-SUIO-1630	The security properties of the SUIO User Data Object (A.SUIO.DO.D) and input and output data (A.SUIO.IO.D) SHALL be defended against the Threats contained in Threat Group 2 “CLCD Used for Displaying Memories and Interfering With Displayed Data”	✓[11]

Table 20 summarises the SUIO core requirements that apply to SUIO Assets.

Table 20: Secure User Input/Output Threat Grouping

Threat Group	1	2	3	4	5	6	7
Threats Assets	Hardware Modules Used for Accessing Memories	CLCD Used for Displaying Memories and Interfering With Displayed Data	Bypass Security by Removal of Battery Power or Removal of External Memory Card	Attack by Replacement of Flash When Power is Off (Pre-Boot)	Extract Secret via Bus Monitoring (Hardware Probes)	Mod Chip Attacks on Data in External RAM	Attack by Replacement of Flash When Power is On (Post-Boot)
A.SUIO. C A.SUIO.IO.C	-	1	1	1	-	-	-
A.SUIO.D A.SUIO.DO.D A.SUIO.IO.D	-	1	1	1	-	-	-

9.5.2 *SECURE USER INPUT/OUTPUT GENERAL REQUIREMENTS*

This set of requirements is intended to guarantee that “What You See is What You Sign”.

[11] In Profile 1, this requirement is intended to protect “User Prompts” in an ASF.

REQ. ID	**REQUIREMENT**	**PROFILE 1**
ATE-SUIO-1640	It SHALL NOT be possible for one Application to violate the security properties of A.SUIO.DO.D of another Application	✓
ATE-SUIO-1650	A.SUIO.IO.C and A.SUIO.IO.D SHALL NOT be able to violate the security properties of any instance of A.SUIO.DO.D	✓
ATE-SUIO-1660	SUIO Code and Data Assets (A.SUIO.C, A.SUIO.D, A.SUIO.IO.C, A.SUIO.IO.D) SHALL be Integrity checked during boot by the FSB	✓
ATE-SUIO-1670	Only authorised SUIO code and device drivers SHALL be installed and only authorised updates to SUIO code and device drivers SHALL be installed	✓
ATE-SUIO-1680	When being used by the SUIO, the access to SUIO user input and output peripherals SHALL be restricted to the SUIO	✓
ATE-SUIO-1690	When a user is interacting with a SUIO associated with a TEE then there MAY be an indication clearly made to that user that they are using a SUIO controlled by that TEE. That TEE specific indication SHALL NOT be controllable by any other EE	✓

Note: SUIO use of output display peripherals requires that, during the period of SUIO use, only the SUIO can access that display peripheral and system features which that display peripheral depends on (i.e. frame buffers for display controllers). It does not imply that data previously placed on the display must be cleared prior to SUIO use. As an example, a screen may have a window open to a secure PIN entry. While the SUIO is dealing with the PIN entry only the SUIO may access the display and screen buffer. SUIO use of input peripherals requires that, during the period of SUIO use, only the SUIO can access the input data.

10 SECURE INTERACTION OF UICC WITH MOBILE

This chapter describes the requirements related to secure interaction between a UICC and an ME (SUM).

A list of use cases is provided in section 10.1.

An introduction is provided in sections 10.2.

The list of Actors and Actors' concerns is provided in section 10.3.

The list of Assets is defined in section 10.4.

The security requirements are defined in section 10.5.

10.1 USE CASES

In general a communication channel enables:

- Partitioning of an Application between the UICC Execution Environment and the ME Execution Environments (TEEs as well as EEs).
- Exchange of data between UICC and ME

This schema can enable different use cases. As examples:

- Secure user I/O (e.g. when an Application on the UICC requests a PIN the PIN insertion and communication to the UICC shall be secured and vice versa when the UICC would like to display a security alert the communicated text and its display shall be secured)
- UICC as Device Management control point
- SIM based DRM
- Mobile Ticketing
- Broadcast

In some of these use cases the communication between UICC and ME shall be protected against certain Threats.

10.1.1 KEY OPERATOR USE CASES

The key operator use cases are listed below in priority order. The following use cases are taken from ETSI TS 102.412: Smart Card Platform Requirements [20].

10.1.1.1 User Interface

A large amount of information currently flows in GSM/3G network enabled services that make use of Application server software and toolkit Applications. In most of these services, at least a part of the information flow has no protection from eavesdropping or tampering: if we focus on the communication between the UICC and the Terminal, the information flowing from the card to the Terminal, and vice versa, is in plain text. The implementation of a Secure Channel will allow a secure data exchange between the end user and the service provider.

10.1.1.2 SCWS User Interface SIM Application Access Via Web Browser

Alan wants to use a service stored in his SIM card which can be accessed through the Smart Card Web Server (SCWS) stored in his SIM, via a browser stored in his Handset. His Application is able to use I/O capacities offered by the Handset, like keyboard, display and audio driver.

Some abuse cases

- Denial-of-Service: the link between the browser and SIM web server is off; this can be done by corrupted port; another Denial-of-Service can be performed via slow down of interaction between the SIM web server and the browser Application.
- Man-In-The-Middle Attack, interception of sensitive data between the client and the server
- Replay Attack
- Cookie spying

10.1.1.3 Proximity Payment

Julie wants to use the payment Application stored in her UICC via her wallet Application stored in her Handset. This Application allows Julie to activate a proximity payment via contactless connectivity to a contactless EFTPOS.

There are three steps to pay:

- Julie selects her payment Application using the wallet Application in her Handset
- She validates the selected items and the amount of the transaction
- She enters a password (for example, for pre-paid services) or a credit-card number

Some abuse cases

- The link between web browser and SIM is Spoofed by a Handset Application and displays false information on the display
- A Man-In-The-Middle Attack is organised by an Handset Application which stored secrets entered by the user when she wants to enter sensitive information
- The server is faked
- Unfeasibility of WYSIWYS (What You See Is What You Sign).

10.1.1.4 UICC as a Control Point for Device Management

Device Management, or DM, specified in OMA, intends to provide the protocols and mechanisms allowing remote settings management of Devices.

It means that the Smart Card (SC) shall store DM objects (Management Objects, or MO) accessible by the Device through the SC to Device interface and also manageable by a remote server (through the Device). This interface is currently not ciphered and DM information will be exchanged without protection. The Availability of a Secure Channel will enable the securing and protection of the communications occurring between the Device DM user agent and the Smart Card.

10.1.1.5 DRM and Distributed Applications

Digital Rights Management (DRM) is meant to secure media content owned by a service provider; the end-user has a limited set of rights to use the content. Open Mobile Alliance (OMA) DRM specifies a model where the rights are bound to a Device, not to a user. This implies that when the user needs to change the player (i.e. the Handset), the rights have to be downloaded onto the new Device and the certificates are to be recalculated with the new Terminal ID. A different scheme is proposed, in order to link the rights to a user rather than to the Handset: the Rights Object (RO) might be stored in the user's UICC together with part of the DRM user agent. This implies that when the user needs to change the player (i.e. the Handset), the rights do not have to be downloaded onto the new Terminal. This solution has the following advantages:

1. The user can play content in any mobile Terminal containing a genuine media player (OMA compatible) and accepting the UICC.
2. The user would not require a network connection. This is useful for situations where the user does not have network coverage (e.g. underground station or on a plane).
3. The operator stores its RO in a tamper-resistant device, which is under its control (administrated via an OTA platform).

This scenario is only possible with a Secure Channel between a Trusted Execution Environment in the Handset and the UICC based DRM User Agent (UA) providing the Content Encryption Key (CEK).

10.2 INTRODUCTION

Mutual authentication between an ME and a UICC and the security of communications between an ME and a UICC is a valuable enabler for a number of services.

The UICC contains the main trusted-by-operator Execution Environment and a trusted-by-operator Secure Storage, it is a tamper-resistant device, i.e. it is resistant to invasive Attacks, fault Attacks and side-channel analysis.

A Secure Channel protects the Confidentiality and Integrity of communication between the UICC and an ME. Ad-hoc solutions may be used to secure communications, but a standardised protocol has been specified in ETSI SCP.

The ETSI requirements for a Secure Channel (SC) are described in ETSI TS 102.412: Smart Card Platform Requirements [20] and the SC specification is defined in ETSI TS 102 484: Secure Channel between a UICC and an End Point Terminal; (Release 7) [21]

The following requirements are agnostic with respect to the protocols used, although they are based primarily on the technical specifications of ETSI SCP.

Secure UICC-ME Communication is a facility (called hereafter SUM) that manages and implements the Secure Channel between ME and UICC.

We consider two sets of SUM security requirements which we call Profiles 1 and 2. Each SUM Profile's requirements define certain required functionalities of the SUM and also the scope of the Attacks to which SUM Assets should be resistant. The two Profiles offer incremental levels of security (that is, Profile 2 offers stronger security than Profile 1).

The Applications on an EE or TEE request the SUM to open Secure Channels towards UICC Applications and receive requests to open Secure Channels from the UICC Applications. The Secure Channel will be handled by the Critical SUM Secure Channel Manager (A.SUM.SC.MNG.C) and the General SUM Secure Channel Manager (A.SUM.SC.DIS). The SUM will protect A.SUM.UAUTH.K, A.SUM.TAUTH.K, A.SUM.TSC.K and A.SC.SC.D against identified Threats.

10.3 SUM ACTORS AND THEIR CONCERNS

Table 21 describes all Actors involved in the SUM, their main concerns, and the trust link between them.

Table 21: SUM Actors and Their Concerns

Actor	Main Concerns (in this context)
Operators	Security of own services in the UICC as well as security of the enabled service on the ME through UICC enablers. Security of communication between ME and UICC and Confidentiality and Integrity of information on the ME coming from the UICC. Security of UICC Assets
Service Providers	For some use cases the Application and content
Content Providers	Protection of their content, keys and content usage rights that pass between the UICC and ME

Actor	Main Concerns (in this context)
End User	Personal information stored on UICC and service Availability
Manufacturer	Security of ME information communicated to the UICC and trustworthiness and security of information obtained from the UICC
UICC provider	Security of communication between ME and UICC and Confidentiality, Integrity and Authenticity of information on the ME coming from the UICC
Attacker	Obtaining unauthorised access to UICC keys, data and services. Obtaining unauthorised access to SC communicated data and SC master and session key. Successful substitution Attack. Successful Man-In-The-Middle Attack

10.4 INTERACTION BETWEEN UICC AND MOBILE EQUIPMENT ASSETS

10.4.1 INTERACTION BETWEEN UICC AND MOBILE EQUIPMENT ASSETS TABLE

The following Assets in the implementation of a TEE require protection. The Assets table 22 describes various properties of Secure UICC-ME (SUM) Assets that were used to derive the SUM requirements.

Table 22: SUM Assets

Asset (A) or Asset Container (AC)	Description	Type	Owner†	Protection Level /Level of Sensitivity	Security Properties Required
A.SUM.xxx.y *with y=K for Key,* *y=D for Data,* *y=C for Code,* *y=H for Hardware* or AC.SUM.xxx		*Key/Data* */Code /Hardware*	*See Actors List*	*Sensitive/ Operational*	*Integrity (I)* *Confidentiality (C)* *Authenticity (Au)*
A.SUM.UAUTH.K	SC UICC Key Keys used to authenticate UICC to ME	Key	O	S	I, Au, C

Asset (A) or Asset Container (AC)	Description	Type	Owner†	Protection Level /Level of Sensitivity	Security Properties Required
A.SUM.TAUTH.K	SC ME Keys Keys used to authenticate ME to UICC	Key	M	S	I, Au, C
A.SUM.SC.K	SC Session Keys Session keys used to protect channel communications	Key		S	I, Au, C
A.SUM.SC.D	SC Data Exchanged data in the SC	Data	O U SP CP M	S	I, Au, C
A.SUM.SC.MNG.C	Critical SUM Secure Channel Manager ME software component managing a SC (including key management)	Code	O M	S	I, Au
A.SUM.SC.MNG.D	Critical SUM Secure Channel Manager Data Data used by A.SUM.SC. MNG.C to authenticate UICC and define policy	Data	M	S	I, Au
A.SUM.SC.DIS.{C,D}	General SUM Secure Channel Manager ME software component managing discovery process and communications protocols	Code, Data	M	O	I, Au

Note:

† Denotes that the given set of owners of a specific Asset is only indicative.

10.4.2 SUM ASSET GROUPING

In order to facilitate requirement understanding, the following groups of Assets are defined:

- **ME SUM Code and Data Assets** groups contain: A.SUM.SC.D, A.SUM.SC.MNG.{C, D} and A.SUM.SC.DIS.{C,D}
- **ME SUM Key Assets** group contains: A.SUM.TAUTH.K and A.SUM.SC.K

10.5 SECURE INTERACTION OF UICC WITH MOBILE REQUIREMENTS

10.5.1 SECURE INTERACTION OF UICC WITH MOBILE CORE REQUIREMENTS

The Asset security properties referred to in the following requirements are defined in Section 10.4.1. All SUM Core Requirements are given in Chapter 3 “Trusted Environment Core requirements”.

Table 23 summarises the Core SUM requirements in the table above.

Each cell contains the Profiles for which an Asset group has to be protected against a Threat group.

Table 23: SUM Threat Grouping

Threat Group	1	2	3	4	5	6	7
Threats Assets	Hardware Modules Used for Accessing Memories	CLCD Used for Displaying Memories and Interfering With Displayed Data	Bypass Security by Removal of Battery Power or Removal of External Memory Card	Attack by Replacement of Flash When Power is Off (Pre-Boot)	Extract Secret via Bus Monitoring (Hardware Probes)	Mod Chip Attacks on Data in External RAM	Attack by Replacement of Flash When Power is On (Post-Boot)

A.SUM.SC.D
A.SUM.SC.MNG.{C, D}
A.SUM.SC.DIS.{C,D}
	2	2(C)* or N/A	1, 2	1, 2	2(C)* or N/A	-	-

A.SUM.TAUTH.K A.SUM.SC.K	2	2 (C)	1, 2	1, 2	2 (C)	2	2

Note:

(C) Denotes the fact that only Confidentiality of these Assets needs to be defended against this Threat group.

(C)* Denotes the fact that these Assets may not require Confidentiality protection, but if they do, then this Threat group must be defended against.

10.5.2 SECURE INTERACTION OF UICC WITH MOBILE SPECIFIC REQUIREMENTS

REQ. ID	REQUIREMENT	PROFILE 1	PROFILE 2
ATE-SUM-1700	A TEE in which it is possible to run Applications that needs secure communication with UICC or a TEE where an operator can load Applications SHALL: be able to setup and manage a Secure Channel with the UICC as defined in ETSI TS 102 484: Secure Channel between a UICC and an End Point Terminal; (Release 7) [21] OR be able to support the setup and management of such channels by providing services guaranteeing the security properties of such channels	✓	✓
ATE-SUM-1710	If a Secure Channel mechanism is used by an Application residing in a particular TEE then the parts of the ME implementation of the Secure Channel that guarantee the security properties of the channel (i.e. A.SUM.SC.MNG.C) SHALL be executed either from that TEE or from a TEE of an equivalent or higher Profile	✓	✓
ATE-SUM-1720	The ME TEE(s) SHALL satisfy the requirements listed in section 4.4.8: Inter-Execution Environment Communications, with the exception of ATE-TEE-0750 and ATE-TEE-0770 where the UICC is considered a peer EE and the UICC Application is considered a peer Application	✓	✓

REQ. ID	REQUIREMENT	PROFILE 1	PROFILE 2
ATE-SUM-1725	In the case of an Open TEE having a Secure Channel connection to the UICC, it SHALL be possible to launch an operator Application (defined in TS 102 223 section 8.88 [29]) inside an Open TEE from the UICC, based on TEE acceptance, using the Terminal in Server mode mechanism from TS 102 223 section 7.8 [29]	✓	✓

PART II: USAGE OF TR1 ENABLERS AND OPERATOR USE CASES

In Part II of this document we consider a sample of potential operator use cases. The use cases were selected from a range of operator services: Broadcast TV, Device Management and Mobile commerce. For each service, a likely architecture is selected and analysed, high level security goals are identified, and then TR1 requirements are derived, showing how TR1 enablers can be used to meet these security goals.

Chapters 11 to 13 are dedicated to introducing and defining requirements for the different TR1 use cases:

- Broadcast
- Trusted Device Management
- Mobile Commerce

The selected use cases by no means represent the only way of implementing the various operator services, but are examples of likely service implementations for some operators. An operator who adopts these use cases is therefore likely to adopt the derived requirements. The example use cases covered in Part II were also selected because they make good use of TR1 enablers. Therefore even if an operator does not adopt these use cases, the resulting requirements provide a good indication of how operators are likely to make use of TR1 enablers to satisfy other use cases.

11 BROADCAST SERVICE PROTECTION

This section describes the security requirements related to the Broadcast Service Protection use case.

An introduction to the Broadcast Service Protection use case is provided in section 11.1.

The list of Actors and Actors' concerns is provided in section 11.2.

The list of Broadcast Service Protection Assets is defined in section 11.3.

The security requirements are defined in section 11.4.

11.1 BROADCAST SERVICE PROTECTION USE CASE INTRODUCTION

Broadcast TV services can in principle be received by anyone. The broadcast channel must therefore be encrypted to prevent just anyone from receiving the content. Mechanisms are used to ensure that only legitimate subscribers can obtain the keys needed to decrypt the broadcast channel and so receive the content. The generic term for these mechanisms and the channel encryption is service protection.

The main Broadcast Service Protection protocols designed for the Mobile industry are:

- OMA BCAST DRM Profile
- OMA BCAST Smart Card Profile
- OSF Open Security Framework

The protocol that is taken into account for the subsequent sections is OMA BCAST Smart Card Profile, as defined in the OMA specification 'Service and Content Protection for Mobile Broadcast Services' [17].

11.1.1 ATTACKS – MOTIVATIONS AND SCENARIOS

11.1.1.1 Attacks on Service

These Attacks will be made to get free access, i.e. free decryption, of the broadcast channel, at the time of broadcast. That is, the Attack will be for a fraudulent user to be able to decrypt the broadcast channel even though they have not paid to be able to do this.

The way this Attack is achieved is for there to be at least one user who has paid their subscription (or other charge) and so is entitled to be able to decrypt the broadcast and the user receives the keys required to do this. This user then extracts keys from their receiving Terminal and then distributes them to unauthorised users who insert the keys into their Terminals in an unauthorised manner.

The Attack has three parts:

- Extraction of the key(s) from the paying user's Terminal

- Distribution of the keys to non-paying users
- Insertion by the non-paying users of these keys into their Terminals (or Terminal simulators).

All three parts must be possible for the Attack to succeed.

11.1.1.2 Attacks on Content

These Attacks will be made to get free access and free distribution of value-added content delivered through the broadcast channel, after the content has been broadcast. That is, the Attack goal will be for a fraudulent user to be able to read/use the content that was delivered to them through the broadcast channel. They will be able to do this at will, despite the fact that they do not have the rights to read/use this content anymore. The fraud can be extended if this Attacker delivers the unprotected content to other users.

The way this Attack is achieved is for there to be at least one user who has paid their subscription (or other charge) and so is entitled to be able to decrypt the broadcast. The user receives the keys required to do this. This user then extracts keys from their receiving Terminal, stores the value-add decrypted content at the time of broadcast, then replays it and/or redistributes it to unauthorised users.

The Attack has three parts:

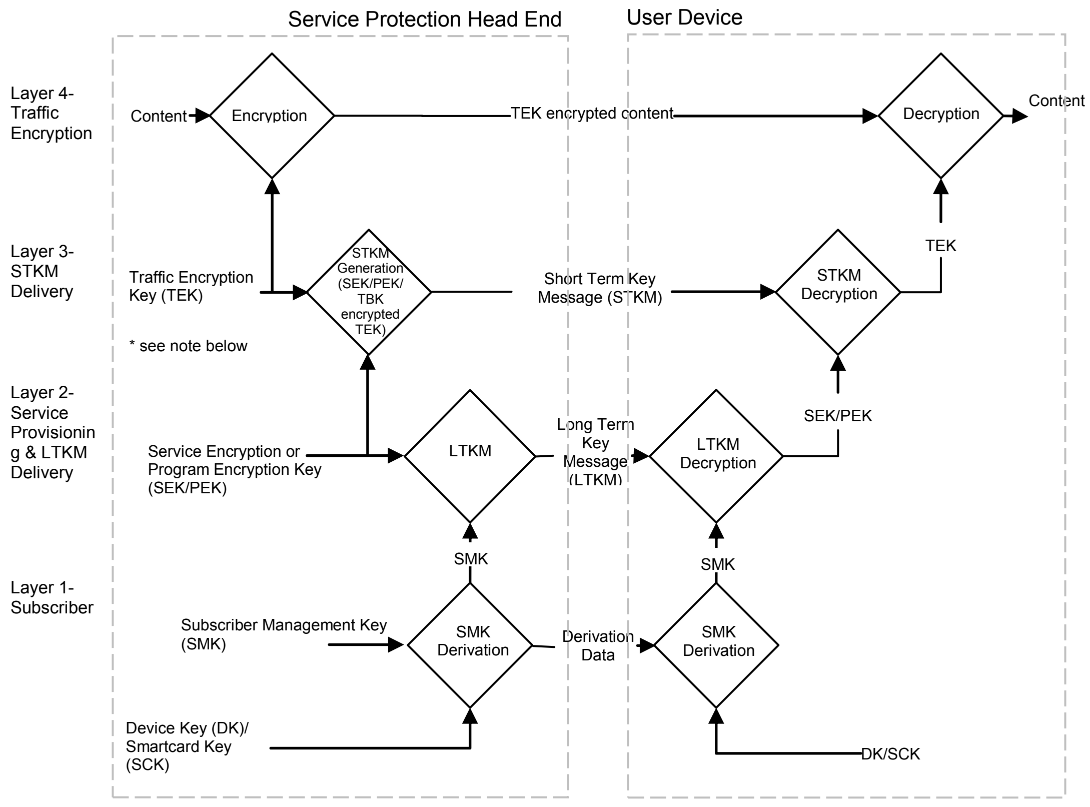
- Unauthorised storage of value-add content, circumventing the rights management system
- Distribution of the decrypted content to non-paying users
- Insertion by the non-paying users of this content into their Terminals (or Terminal simulators).

All three parts must be possible for the Attack to succeed.

11.1.2 HIERARCHY OF KEYS

Broadcast Service Protection protocols use a hierarchy of keys. The key hierarchy and layering of OMA BCAST Smart Card Profile is shown in Figure 5.

Figure 5: OMA BCAST Smart Card Profile Key Hierarchy


Service Protection Head End
User Device
Layer 4-Traffic Encryption
Content
Encryption
TEK encrypted content
Decryption
Content
Layer 3-STKM Delivery
Traffic Encryption Key (TEK)
STKM Generation (SEK/PEK/ TBK encrypted TEK)
Short Term Key Message (STKM)
STKM Decryption
TEK
* see note below
Layer 2-Service Provisioning & LTKM Delivery
Service Encryption or Program Encryption Key (SEK/PEK)
LTKM
Long Term Key Message (LTKM)
LTKM Decryption
SEK/PEK
SMK
SMK
Layer 1-Subscriber
Subscriber Management Key (SMK)
SMK Derivation
Derivation Data
SMK Derivation
Device Key (DK)/ Smartcard Key (SCK)
DK/SCK

Note: TBK use is optional

The OMA specification 'Service and Content Protection for Mobile Broadcast Services' [17] defines two variants of the Smart Card Profile. The two variants are referred to as the (U)SIM Smart Card Profile and the (R-)UIM/CSIM Smart Card Profile respectively. The two variants differ in the way that the Smart Card establishes the SMK for the first layer, but are otherwise the same for the other three layers. The focus from here will be on the (U)SIM Smart Card Profile.

- Layer 1: Registration - Subscriber Key Establishment

  This layer makes use of a Secret Key stored on a Smart Card based identity module. This key is referred to as "Smart Card Key" (SCK) in the Smart Card Profile. The SCK is a pre-provisioned Secret Key that is shared between the Smart Card and the Smart Card issuer. If the Smart Card issuer is not also the Broadcast Service Provider, then the SCK is unknown to the Broadcast Service Provider.

  The SCK is used to create two keys used at Layer 1. The Subscriber Management Key (SMK), which is used to protect the delivery of SEK/PEKs within LTKM from the BSM (BCAST Subscription

Management) to the Terminal and the Subscriber Request Key (SRK), which is used to authenticate the UE and the BSM.

As per ’Service and Content Protection for Mobile Broadcast Services’ [17], for the (U)SIM Smart Card Profile, the SMK SHALL be stored on a USIM when using GBA_U, and on a Terminal when using GBA_ME or 2G GBA. Requirements in this chapter apply to GBA_U only and do not apply to GBA_ME. Implementations of GBA_ME are out of scope for this chapter. Therefore, there will be no specific security requirements for the ME related to SMK establishment and security properties maintenance.

Note: Clarification of interaction between OMA BCAST (U)SIM Smart Card Profile, MBMS and GBA (as per ’Service and Content Protection for Mobile Broadcast Services’ [17]):

The (U)SIM Smart Card Profile uses the key management defined by Multimedia Broadcast/Multicast Service (MBMS) security framework as defined in 3GPP TS 33.246 – ‘3G Security; Security of Multimedia Broadcast/Multicast Service (MBMS) (Release 7)’ [15]. To that extent, the SMK is equivalent to the MBMS User Key (MUK), the SRK is equivalent to the MBMS Registration Key (MRK), and the BSM provides the functionality that in MBMS is provided by the MBMS Broadcast-Multicast Service Centre (BM-SC) security functions.

The MBMS uses the Generic Bootstrapping Architecture (GBA) from 3GPP TS 33.220 – ‘Generic Authentication Architecture (GAA); Generic Bootstrapping Architecture (Release 7)’ [13] to establish an MUK and MRK between the BM-SC, an instance of a GBA Network Application Function (NAF) and the USIM/Terminal. GBA requires the implementation of a Bootstrapping Server Function (BSF) to enable the bootstrapping procedure required to establish MUK and MRK. The (U)SIM Smart Card Profile BSM is assumed to support BSF functionality required to establish SMK and SRK, which are equivalent to the MBMS MUK and MRK respectively.

- Layer 2: Long Term Key Message Delivery

  Depending on the service configuration, within the LTKM, a Program Encryption Key (PEK) or a Service Encryption Key (SEK), used respectively for pay per view or subscription customers, is delivered protected by SMK. The messages carrying encrypted SEKs or PEKs are called the Long Term Key Messages (LTKMs). As per ’Service and Content Protection for Mobile Broadcast Services’ [17], for the (U)SIM Smart Card Profile, the SEK or PEK SHALL be stored on a USIM when using GBA_U or on a Terminal when using GBA_ME or 2G GBA. It is assumed in the rest of the document that GBA_U will be used. Therefore, there will be no specific security requirements for the ME related to SEK and PEK security properties maintenance.

- Layer 3: Short Term Key Message Delivery

  It delivers the Short Term Key Message (STKM) within which Traffic Encryption Keys (TEKs) are protected using SEK or PEK. The TEKs may optionally be encrypted with a Terminal Binding Key (TBK) before being encrypted by the SEK/PEK, to provide for Terminal Binding.

- Layer 4: Content

  The content is encrypted with Traffic Encryption Keys (TEKs) before being broadcast. TEKs change frequently (often every minute or less) to make it harder for an Attacker, who has managed to access the TEKs within his Device, to redistribute the TEKs to unauthorised receivers.

The four-layer cryptographic architecture presented above provides de-coupling of, on the one hand, Device and service/program keys (SEKs/PEKs), and on the other hand, content and content-related key (TEKs).

11.1.3 SERVICE ENCRYPTION KEYS, CHANNELS AND PAY PER VIEW

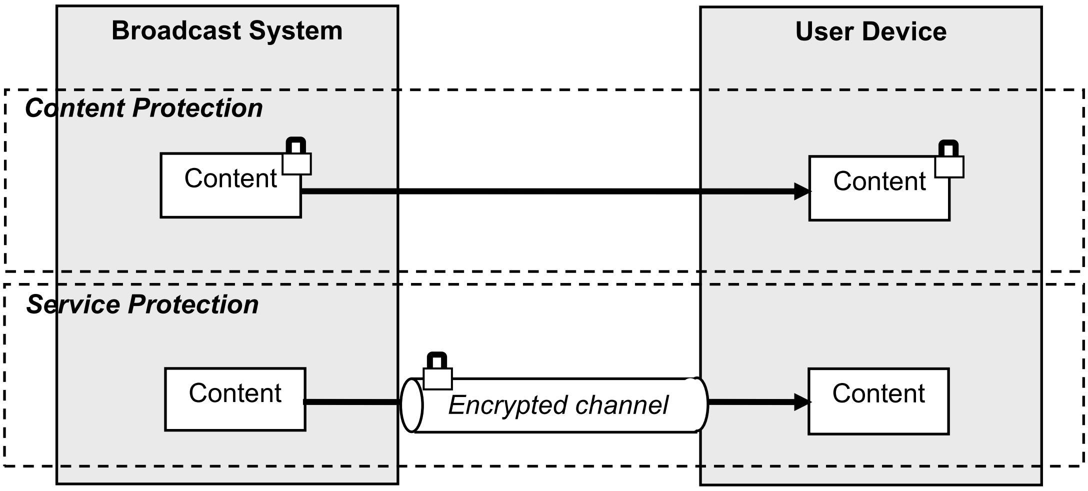
- A Broadcast Service Provider will usually assign a separate SEK for each channel they broadcast. If a user subscribes to more than one channel they therefore require a SEK for each channel they subscribe to.
- For services made available via monthly subscription (for example), SEK could have a lifetime of a month. Valid subscribers to a particular channel can be given a new SEK for that channel at the end of one month and this SEK will allow decryption of the next month's TEKs and therefore the next month's content on that channel. Users who are ending their subscription for some reason are just not given the new SEK.
- SEKs can have a lifetime shorter than a month. The lifetime could be a day, for example, and this would allow the Broadcast Service Provider to offer subscription on a daily, as oppose to a monthly basis. However, distribution of the new SEK individually to subscribers would be a correspondingly larger task.
- In the extreme, service keys can have a lifetime of a single program, for example a pay-per-view film or sporting event. Service keys are often called Program Encryption Keys (PEKs) in such a case.
- Individual programs in a channel can be available both for subscribers of the channel and on a pay-per-view basis. In this case, the keys for the program are encrypted with a PEK (so that the PEK is provided to the pay-per-view customers) and the PEK is encrypted with an SEK (so that the program is also available to the channel subscribers).

11.1.4 SERVICE PROTECTION AND CONTENT PROTECTION

Service protection differs from content protection (which is usually provided by Digital Rights Management (DRM) technology). With service protection, in some cases (e.g. service protection of content that is Free to Air (FTA) over terrestrial broadcasts) the Broadcast Service Provider does not practically care what the user does with the content after they have received it. In contrast, the whole aim of content protection is to ensure the long-lived protection of the content. The Broadcast Service Provider does not care in the FTA case because this content is available over other channels which allow the user to record the content, so there is no point in sealing within the user's phone the lower quality mobile-only version of the content that is broadcast.

However, if premium content (e.g. content only available over premium TV channels) is being broadcast then more protection may be required. For example, the Broadcast Service Provider may need to authenticate the Terminal itself (and not just the user's (U)SIM) in order to determine whether the Terminal is "compliant" to rules concerning recording and storage of the content on the phone and rules about whether the content can be moved off the Terminal (e.g. memory card or VGA output) in any way.

Figure 6: Service and Content Protection



11.2 BROADCAST SERVICE PROTECTION ACTORS AND THEIR CONCERNS

The goal of Mobile TV Broadcasting is to provide broadcasting contents in a mobile Environment and to ensure that the broadcasted content and the access rights of the content are protected. Table 24 describes all Actors involved in Broadcast Service Protection, their main concerns, and the trust link between them.

Table 24: Broadcast Service Protection Actors and Their Concerns

Actor	**Main Concerns (in this context)** *(role of the Actor, responsibilities, trust relationship with the other Actors)*
Operators (O)	Protection of their revenue against unauthorised access to Broadcast Service. Protection of Broadcast Service and Broadcast Service usage rights. Operators will have to trust the manufacturers, the UICC manufacturer and the Broadcast Service Providers on the security properties that are required for an end-to-end Broadcast Application
Broadcast Service Providers (BSP)	Protection of Broadcast Service and Broadcast Service usage rights. The Broadcast Service Provider will have to trust the manufacturer and the UICC manufacturer and possibly validate the mobile Broadcast Service Protection end to end architecture
Broadcasters (BC)	Protection against unauthorised access to Broadcast Service. Protection of Broadcast Service and Broadcast Service usage rights. Broadcasters will have to trust the operator, the Broadcast Service Provider, the manufacturer and the UICC manufacturer
Content Providers (CP)	Protection of their content and content usage rights, and possibly access to Broadcast Service if their revenues are linked to this. Content Providers will have to trust the operator, the Broadcast Service Provider, the manufacturer and the UICC manufacturer
End Users (U)	End users expect full access to Broadcast Service in accordance with their Broadcast Service usage rights.

Actor	**Main Concerns (in this context)** *(role of the Actor, responsibilities, trust relationship with the other Actors)*
Terminal Manufacturer (M)	Implements an Environment allowing Access protection, continuity of service for Broadcast Service and Content. Ensures the Terminal cannot be hacked. Maintain end user and Broadcast Service Provider trust in the Terminal
UICC Manufacturer (UM)	The UICC manufacturer implements an Environment supporting the infrastructure for Layer 1, 2 and 3, including support for GBA as per 3GPP TS 33.220 – 'Generic Authentication Architecture (GAA); Generic Bootstrapping Architecture (Release 7)' [13] and MBMS as per 3GPP TS 33.246 – '3G Security; Security of Multimedia Broadcast/Multicast Service (MBMS) (Release 7)' [15]. Maintain security properties of SCK, SMK, PEK and SEK in UICC
Attacker (A)	The Attacker wants: Unauthorised access to Broadcast Services and Contents. Unauthorised access to keys protecting the Broadcast content and service. Successful Denial-of-Service. Unauthorised access by re-use of Broadcast Service usage rights

11.3 BROADCAST SERVICE PROTECTION ASSETS

11.3.1 ASSETS TO BE PROTECTED FOR BROADCAST SERVICE PROTECTION

The following Assets have to be protected to secure Broadcast Service.

Assets held entirely in the UICC:

- Smart Card Key: A.BC.SCK.K
- Subscriber Management Key: A.BC.SMK.K
- Long Term Key Message, decrypted: A.BC.LTKM.D

- Service Encryption Key: A.BC.SEK.K
- Program Encryption Key: A.BC.PEK.K

There will not be security requirements on the above Assets since security provided by UICC is considered adequate.

Assets in the ME:

- Subscriber Request Key: A.BC.SRK.K
- Bootstrapping Transaction Identifier: A.BC.BCA.BTID.D. This Asset is used by the Broadcast Service Protection Application code to send it as username in Service Request procedures to the NAF.
- Short Term Key Message, decrypted: A.BC.STKM.D
- Traffic Encryption Keys: A.BC.TEK.K
- Terminal Binding Key: A.BC.TBK.K
- Broadcast Service Protection Application code: A.BC.BCA.C
- Broadcast Service Protection Application data: A.BC.BCA.D
- Broadcast content in clear text: A.BC.CON.D.
- Access criteria flags that are part of STKM: A.BC.FLA.D
- Electronic Service Guide (ESG): A.BC.ESG.D
- Device or Broadcast Application keys used to authenticate ME to BSP (NAF) for TBK retrieval or Secure Channel key agreement A.BC.MEK.K

11.3.2 BROADCAST SERVICE PROTECTION ASSET TABLE

Table 25 lists Assets in the implementation of Broadcast Service Protection that require protection. The Assets table describes various properties of Assets that were used to derive the BC requirements.

Table 25: Broadcast Service Protection Assets

Asset (A) or Asset Container (AC)	**Description**	**Type**	**Owner†**	**Protection Level /Level of Sensitivity**	**Security Properties Required**
A.BC.xxx.y or AC.BC.xxx.y with y=K for Key, y=D for Data, y=C for Code, y=H for Hardware		*Keys (K), Code (C), Data (D), Hardwar e (H)*	*See Actors List*	*Sensitive (S) Operational (O)*	*Integrity (I) Confidentiality (C) Authenticity (Au) Non-Replay (NR)*
Assets held entirely in the UICC					
A.BC.SCK.K	Smart Card Key	K	O	S	I, C, Au No requirements since stored in UICC
A.BC.SMK.K	Subscriber Management Key	K	O	S	I, C, Au No requirements since stored in UICC
A.BC.LTKM.D	Long Term Keystream Messages, decrypted	D	BSP	S	I, C, Au, NR No requirements since stored in UICC
A.BC.SEK.K	Service Encryption Key	K	BSP	S	I, C, Au No requirements since stored in UICC
A.BC.PEK.K	Program Encryption Key	K	BSP	S	I, C, Au No requirements since stored as plain-text in UICC
Assets in the ME					
A.BC.SRK.K	Subscriber Request Key	K	O	S	I, C, Au

Asset (A) or Asset Container (AC)	Description	Type	Owner†	Protection Level /Level of Sensitivity	Security Properties Required
A.BC.BCA.BTID.D	Bootstrapping Transaction Identifier	D	O	S	I, Au
A.BC.STKM.D	Short Term Keystream, Messages, decrypted	D	BSP	S	I, C, Au
A.BC.TEK.K	Traffic Encryption Keys	K	BSP	S	C
A.BC.TBK.K	Terminal Binding Key	K	BSP	S	I, C
A.BC.BCA.C	Broadcast Service Protection Application code, running on the ME	C	BSP/M	S	I, Au
A.BC.BCA.D	Data used by the Broadcast Service Protection Application	D	BSP/M	S	I
A.BC.CON.D	Broadcast content, decrypted	D	CP	S	I, C*
A.BC.FLA.D	Access criteria flags contained in STKMs	D	BSP	S	I
A.BC.ESG.D	Electronic Service Guide	D	BSP	O	I
A.BC.MEK.K	Device or Broadcast Application keys used to authenticate ME to BSP (NAF) for TBK retrieval or Secure Channel key agreement	K	BSP/M	S	I, C, Au

Notes:

† Denotes that the given set of owners of a specific Asset is only indicative.

* Confidentiality has to be protected if and only if Content Protection is required beyond Service Protection.

11.3.3 Broadcast Service Protection Asset Grouping

In order to facilitate requirement understanding, the following groups of Assets are defined:

- **BC Code Assets** group contains the Broadcast Service Protection code Asset: A.BC.BCA.C
- **BC Data Assets** group contains all Broadcast Service Protection data Assets that are handled in the ME: A.BC.BCA.D, A.BC.CON.D, A.BC.FLA.D, A.BC.STKM.D and A.BC.ESG.D
- **BC Key Assets** group contains all Broadcast Service Protection key Assets that are handled in the ME: A.BC.SRK.K, A.BC.TEK.K, A.BC.TBK.K and A.BC.MEK.K

11.4 REQUIREMENTS

The Assets security properties referred to in the following requirements are defined in section 11.3.2.

11.4.1 BROADCAST SERVICE PROTECTION CORE REQUIREMENTS

In addition to TR1 Core Requirements defined in Chapter 3 “Trusted Environment Core requirements”, further core requirements are defined for BC in this section.

REQ. ID	SECURITY GOAL	TR1 REQUIREMENT
ATE-BCA-1730	The security properties of BC Key Assets, of A.BC.STKM.D, of A.BC.FLA.D and of A.BC.CON.D (in the content protection use case) SHALL be defended against Threats contained in Threat Group 3 "Bypass Security by Removal of Battery Power or Removal of External Memory Card"	BC Key Assets SHALL be managed by a Profile 1 TEE and considered as Application Keys (A.EE.APP.K). A.BC.SRK.K and A.BC.TBK.K SHALL be stored by a Profile 1 SST and considered as SST Sensitive Objects (SST.DO.D(C)). A.BC.CON.D (only in the content protection use case) and A.BC.STKM.D SHALL be managed by a Profile 1 TEE and considered as Application Data (A.EE.APP.D) requiring Integrity and Confidentiality protection. They SHALL be stored by a Profile 1 SST and considered as SST Sensitive Objects (SST.DO.D(C)). A.BC.FLA.D SHALL be managed by a Profile 1 TEE and considered as Application Data (A.EE.APP.D) requiring Integrity protection. It SHALL be stored by a Profile 1 SST and considered as an SST Sensitive Object (SST.DO.D)
ATE-BCA-1740	The security properties of BC Code Assets SHALL be defended against Threats contained in Threat Group 3 "Bypass Security by Removal of Battery Power or Removal of External Memory Card"	BC Code Assets SHALL be managed by a Profile 1 TEE and considered as Application Code (A.TEE.APP.C)

REQ. ID	SECURITY GOAL	TR1 REQUIREMENT
ATE-BCA-1750	The security properties of BC Code Assets SHALL be defended against Threats contained in Threat Group 4 "Attack by Replacement of Flash When Power is Off (Pre-Boot)"	BC Code Assets SHALL be managed by a Profile 1 Flexible Secure Boot Facility and considered as Code Base Data (A.FSB.CB.D)
ATE-BCA-1760	When present at boot, the security properties of BC Data and Key Assets SHALL be defended against Threats contained in Threat Group 4 "Attack by Replacement of Flash When Power is Off (Pre-Boot)"	When present at boot, BC Data and Key Assets SHALL be managed by a Profile 1 Flexible Secure Boot Facility and considered as Code Base Data (A.FSB.CB.D)

Table 26 summarises the BC core requirements that apply to BC Assets.

Table 26: Broadcast Service Protection Threat Grouping

Threat Group	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Threats Assets	Hardware Modules Used for Accessing Memories	CLCD Used for Displaying Memories and Interfering With Displayed Data	Bypass Security by Removal of Battery Power or Removal of External Memory Card	Attack by Replacement of Flash When Power is Off (Pre-Boot)	Extract Secret via Bus Monitoring (Hardware Probes)	Attacks on Data in External RAM	Attack by Replacement of Flash When Power is On (Post-Boot)
A.BC.BCA.C	-	-	Y	Y	-	-	-
A.BC.CON.D A.BC.STKM.D A.BC.FLA.D	-	-	Y	Y	-	-	-
A.BC.BCA.D A.BC.ESG.D	-	-	-	Y	-	-	-
A.BC.SRK.K A.BC.TEK.K A.BC.TBK.K	-	-	Y	Y	-	-	-

11.4.2 SERVICE PROTECTION REQUIREMENTS

REQ. ID	SECURITY GOAL	TR1 REQUIREMENT
ATE-BCA-1770	The security properties of Keys exchanged between the Broadcast Service Protection Application (A.BC.BCA.C) and the UICC SHALL be protected	A.BC.SRK.K SHALL be managed by a Profile 1 TEE and considered as an Application Key (A.EE.APP.K) requiring Integrity, Confidentiality and Authenticity protection. A.BC.SRK SHALL be stored by a Profile 1 SST and considered as an SST Sensitive Object (SST.DO.D(C)) If A.BC.TBK.K is not used to protect A.BC.TEK.K, then A.BC.SRK.K and A.BC.TEK.K SHALL be exchanged through a Profile 1 SUM and SHALL be considered as Exchanged data in the SC (A.SUM.SC.D)
ATE-BCA-1780	The Terminal Binding Key, if it exists, SHALL be Integrity, Confidentiality and Authenticity protected during use and storage on the ME	A.BC.TBK.K SHALL be managed by a Profile 1 TEE and considered as an Application Key (A.EE.APP.K) requiring Confidentiality protection. A.BC.TBK.K SHALL be stored by a Profile 1 SST and considered as an SST Sensitive Object (SST.DO.D(C))

REQ. ID	SECURITY GOAL	TR1 REQUIREMENT
ATE-BCA-1790	The Traffic Encryption Keys SHALL be Integrity, Confidentiality and Authenticity protected when retrieved from UICC and during use on the ME	A.BC.TEK.K SHALL be managed by a Profile 1 TEE and considered as an Application Key (A.EE.APP.K) requiring Confidentiality protection. A.BC.TEK.K SHALL be either exchanged through a Profile 1 SUM and SHALL be considered as exchanged data in the SC (A.SUM.SC.D), or delivered encrypted by A.BC.TBK.K on the ME for subsequent deciphering in the Profile 1 TEE
ATE-BCA-1800	The Subscriber Request Key, the Bootstrapping Transaction Identifier, the Short Term Key Message, the Traffic Encryption Key, the Terminal Binding Key, the Data used by the Broadcast Service Protection Application and the Access Control flags contained in STKM SHALL only be accessible by the Broadcast Service Protection Application on the ME	A.BC.SRK.K, A.BC.STKM.D, A.BC.TEK.K, A.BC.TBK.K, A.BC.BCA.D and A.BC.FLA.D Assets SHALL be managed by a Profile 1 TEE that SHALL guarantee restrictive access by A.BC.BCA.C only
ATE-BCA-1810	The Access Control flags contained in STKMs SHALL be Integrity protected during use and storage on the ME	A.BC.FLA.D SHALL be managed by a Profile 1 TEE and considered as Application Data (A.EE.APP.D) requiring Integrity protection. A.BC.FLA.D SHALL be stored by a Profile 1 SST and considered as an SST Sensitive Object (SST.DO.D)

REQ. ID	SECURITY GOAL	TR1 REQUIREMENT
ATE-BCA-1820	Traffic Encryption Keys SHALL NOT be stored on the Terminal once used	Traffic Encryption Keys (A.BC.TEK.K) SHALL NOT be stored on the Terminal once used
ATE-BCA-1830	If recording of content is not permitted by STKM control flags, then the Terminal SHALL not record and store any encrypted or decrypted content on the Terminal	If recording of content is not permitted by STKM control flags, then the Terminal SHALL not record and store any encrypted or decrypted content on the Terminal
ATE-BCA-1840	If content protection is required then decrypted content SHALL NOT be stored on the Terminal once consumed	If content protection is required then A.BC.CON.D SHALL NOT be stored on the Terminal once consumed

11.4.3 CONTENT PROTECTION REQUIREMENTS

REQ. ID	SECURITY GOAL	TR1 REQUIREMENT
ATE-BCA-1850	If content protection is required after content has been broadcast, the Confidentiality and the Integrity of decrypted broadcast content SHALL be protected during use and consumption on the ME	A.BC.CON.D SHALL be considered as Application Data (A.EE.APP.D) requiring Integrity and Confidentiality protection, and SHALL be protected from any user installed Applications through a Profile 1 TEE

12 TRUSTED DEVICE MANAGEMENT

This chapter describes the security requirements related to the Trusted Device Management use case. Trusted Device Management is used for the management of the TEEs available on the ME.

An introduction to the Trusted Device Management use case is provided in section 12.1.

The list of Actors and Actors' concerns is provided in section 12.2.

The list of Trusted Device Management Assets is defined in section 12.3.

The security requirements are defined in section 12.4.

12.1 TRUSTED DEVICE MANAGEMENT USE CASES

12.1.1 KEY OPERATOR USE CASES

The two key operator use cases are listed below in priority order, and concern respectively the management of the firmware of a TEE (UC1) and the management of the Software of a TEE (UC2). This chapter focuses on the use of 'Open Mobile Alliance Device Management' [22] as the underlying management protocol. Hence, the two use cases UC1 (Trusted Firmware Management) and UC2 (Trusted Software Management) correspond respectively to the management objects defined in the Open Mobile Alliance Firmware Update Management Object [23] and the Open Mobile Alliance Software Component Management Object Candidate [24].

12.1.2 UC1: TRUSTED FIRMWARE MANAGEMENT (UPGRADE, UPDATE)

The Terminal manufacturer develops software to correct bugs, to improve the performance of the ME or to install new drivers. The update package is sent to the management server. The management server retrieves the current configuration (e.g. the version number of the firmware) of the ME, and sends the update package to the ME. The ME installs the software using dedicated mechanisms. Upon completion, it reports status to the management server. An end user confirmation may be requested prior to the update.

12.1.3 UC2: TRUSTED SOFTWARE MANAGEMENT (INSTALLATION, UPDATE, REMOVAL, ACTIVATION AND DEACTIVATION)

This use case encompasses the management of any other type of software Asset than firmware. Examples of software components are Applications, executables, libraries, user-interface elements, certificates, licenses etc.

The Terminal manufacturer or a Service provider develops software (e.g. related to a new service) or a patch to correct existing software. The corresponding package is sent to the management server. The management server retrieves the current software configuration (e.g. the software inventory) of the ME, and sends the package to the ME. The ME installs the software using dedicated mechanisms. Upon completion, it reports status to the management server. An end user confirmation may be requested prior to the update.

The operator or a service provider wants to uninstall a piece of software on the ME. The management server sends a management command to the ME. The ME uninstalls the piece of software. Upon completion, it reports status to the management server. An end user confirmation may be requested prior to the removal.

The operator or a service provider wants to activate or deactivate a piece of software on the ME. The management server sends a management command to the ME. The ME activates or deactivates the piece of software. Upon completion, it reports status to the management server. An end user confirmation may be requested prior to the activation or deactivation.

12.1.4 POSSIBLE FUTURE OPERATOR USE CASES

The following use cases are out-of-scope, they may however be addressed in a future release of this document.

12.1.5 CONFIGURATION PARAMETERS PROVISIONING

A new ME is purchased by an end user and provisioned with parameters. This may be done at the retail store using a Device Management system, or using a Smart Card, or using a Bootstrap mechanism for used Terminals.

12.1.6 CONFIGURATION PARAMETERS MANAGEMENT

An operator changes its platform infrastructure (e.g. the IP address of a gateway). The management server receives the information for the client ME to be updated, potentially retrieves the current configuration of the ME to customise the configuration and sends the information to the ME. The ME stores it and reports status to the management server. An end user confirmation may be requested prior to the update.

12.1.7 LOG REPORTING

The ME silently sends information to the management server about performance, configuration, etc. of the platform. An end user authorisation may be requested for this functionality to be activated.

12.1.8 FAULT DETECTION AND REPORTING

The end user calls the operator's customer care facility to report an error message or a malfunctioning service. The customer care facility agent can query the ME to inspect its current configuration and take the management action to solve the problem.

12.1.9 BACKUP AND RESTORE

Information about the current configuration of an ME and the end user personal data are stored on a backup server. In case this information is accidentally lost, it can be restored on the ME of the end user at his request.

12.2 AN INTRODUCTION TO TRUSTED DEVICE MANAGEMENT

12.2.1 OVERVIEW

Device Management is a generic term encompassing a lot of functionalities such as:

- Configuration and Settings Management
  - Provisioning of parameters for a new Device and maintenance of those parameters.
- Software Management
  - Installation, update, Upgrade, delete etc.
- Firmware Management
  - Bug fixes, new drivers, etc.
- Backup and Restore
- Device Diagnosis and Fault Management

It always implies a Management Authority (which may be the operator, the manufacturer or a Service provider) and a Device on which Management Authorities have the right to perform specific management actions or to manipulate a specific data element or parameter.

The main Device Management protocols designed for the mobile industry are:

- OMA Device Management [22]
- OMA Client Provisioning [25]
- OSGi (Open Services Gateway Initiative) [26]
- GlobalPlatform Device Application Security Management [27]

This document focuses on the use of OMA DM as the management protocol.

12.2.2 OMA DM

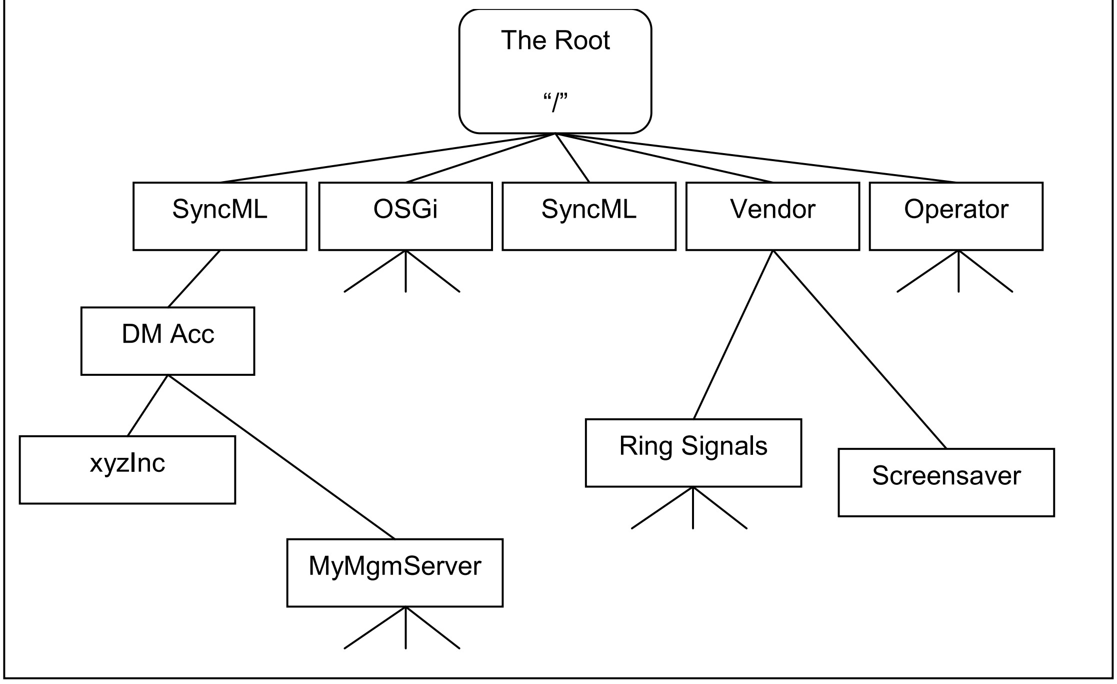
OMA DM is a Management Protocol defined by the Open Mobile Alliance. Its data model is based on a Device Management Tree whose nodes (Management Objects) can be managed with SyncML commands such as Add, Replace and Get. A Management Object can be as small as an integer or large and complex like a background picture or screen saver. Each node possesses a number of properties (Format, Name, Size, etc.) used to provide meta-information about the node in question. A particularly important property from the point of view of security is the ACL (Access Control List) which lists the Management Servers and the SyncML commands they are allowed to carry on the node. This data is thus very sensitive. Indeed any Attacker succeeding in changing the value of the ACL could subsequently send unauthorised management commands to the Device. The ACL corresponds to the Assets Access Control List of the FUMO Management Tree (A.TDM.ACLF.D) and Access Control List of the SCOMO Management Tree (A.TDM.ACLS.D).

Mutual authentication of the Management Server and the Management Client is mandatory in OMA DM and can be provided either by the transport layer or at the protocol layer through CRAM (Challenge Response Authentication Mechanism) MD5. The latter is performed in the following manner:

- Each Management Server possesses a serverID and a corresponding password, and each Device possesses a username and a corresponding password
- A Management Server stores the value of MD5(username:password) for each Device it manages, and a Device stores the value of MD5(serverID:password) for each server managing it
- When a Device wants to authenticate a Management Server, it challenges it with a Nonce and the Management Server must return MD5(MD5(serverID:password):nonce) in order to be authenticated (or vice-versa)

In this example, the credentials that must be securely stored by a Device are its password and the values of MD5(serverID:password) of each Management Server managing it. This corresponds to the Asset Server Authentication Keys (A.TDM.SA.K).

Figure 7: Example Device Management Tree



OMA has defined the Open Mobile Alliance Firmware Update Management Object [23] and a Management Object for Software Management, SCOMO defined in Open Mobile Alliance Software Component Management Object Candidate [24].

The download of the packages containing the firmware update code (A.TDM.DPF.D) or the Software Installation or Update code (A.TDM.DPS.D) is handled by the Download Agent (A.TDM.DL.C).

Once the packages are downloaded, the update or Installation operation is carried out by the firmware or the software management agent (A.TDM.FMA.C, A.TDM.SMA.C). If Authenticity of the package is needed, then it is checked with the keys A.TDM.FPA.K or A.TDM.FPA.K. OMA DM supports encryption. In case encryption is used, the keys A.TDM.FPC.K (Firmware Package Confidentiality) or A.TDM.SPC.K (Software Package Confidentiality) are used to decrypt the content of the package.

12.3 TRUSTED DEVICE MANAGEMENT ACTORS AND THEIR CONCERNS

Table 27 describes all Actors involved in Trusted Device Management, their main concerns, and the trust link between them.

Table 27: Trusted Device Management Actors and Their Concerns

Actor	**Main Concerns (in this context)** *(role of the Actor, responsibilities, trust relationship with the other Actors)*
Content Providers (CP)	Content Providers want any sensitive data residing in the Device to be protected from malicious Applications. Content Providers will trust the operators, the Service providers and the manufacturers
End User (U)	The end user wants his personal data to be protected
Terminal Manufacturer (M)	The Terminal manufacturer wants to ensure that the mechanisms it provides to manage the Devices are secure
UICC Manufacturer (UM)	Protection of their sensitive keys, code and data (e.g. for intellectual property protection)
Attacker (A)	The Attacker may want to obtain unauthorised access to key material, to management data, to inject unauthorised management data, to perform Denial-of-Service Attacks, Replay Attacks of Management commands, etc.
Operator (O)	Operators want to be able to manage securely and remotely their Devices, e.g. settings and configuration management, software management, firmware management and to prevent unauthorised Management commands on their Assets. Operators also want to enable other Actors (e.g. Service providers) to manage their own Applications, without having any access to Applications that are not under their authority. Operators will trust the manufacturer

12.4 TRUSTED DEVICE MANAGEMENT ASSETS

12.4.1 TRUSTED DEVICE MANAGEMENT ASSET TABLE

The following Assets in the implementation of the Trusted Device Management use case require protection. The Assets are split into three groups: the group of Assets shared by UC1 and UC2, the group of Assets specific to UC1 and the group of Assets specific to UC2.

Table 28: Trusted Device Management Assets

Asset (A) or Asset Container (AC)	Description	Owner†	Protection Level /Level of Sensitivity	Security Properties Required
A.TDM.xxx.y or AC.TDM.xxx.y with y=K for Key, y=D for Data, y=C for Code, y=H for Hardware		*See Actors List*	*Sensitive (S), Operational (O)*	*Integrity (I), Confidentiality (C), Authenticity (Au), Non-Replay (NR)*
Assets Shared by the two use cases UC1 and UC2				
A.TDM.SA.K	Server Authentication Keys – the credentials used to authenticate the management server during the download process at the transport layer or the SyncML command	M O	S	I, Au, C[12]
A.TDM.DL.C	Download Agent	M	S	I, Au
A.TDM.DISP.D	Data I/O displayed on Screen – User prompts	M	S	I, Au, NR
A.TDM.KB.D	Data I/O on Keyboard – User Approval	M	S	I, Au, NR
Assets Specific to UC1 (Firmware Management)				
A.TDM.FMA.C	Firmware Management Agent – Code in charge of upgrading the firmware of the ME	M	S	I, Au

[12] The Confidentiality requirement only applies to symmetric keys used for server authentication - asymmetric keys do not need Confidentiality protection

Asset (A) or Asset Container (AC)	Description	Owner†	Protection Level /Level of Sensitivity	Security Properties Required
A.TDM.FPA.K	Firmware Package Authentication Keys – cryptographic keys used to ensure the Authenticity of the downloaded package A.TDM.DPF.D	M O	S	I, C**, Au
A.TDM.FPC.K	Firmware Package Confidentiality Protecting Keys – cryptographic keys used to ensure the Confidentiality of the downloaded package A.TDM.DPF.D	M O	S	I, C**, Au
A.TDM.ACLF.D	Access Control List of the FUMO Management Tree – Data Structure that defines which management commands each management server is allowed to perform on the nodes of the OMA DM FUMO sub-tree	O	S	I, Au
A.TDM.DPF.D	Download Package containing the Firmware Update Code	M O	S	I, C, Au, NR
Assets Specific to UC2 (Software Management)				
A.TDM.SMA.C	Software Management Agent – Code in charge of installing, upgrading and uninstalling software of the ME	M	S	I, Au
A.TDM.SPA.K	Software Package Authentication Keys – cryptographic keys used to ensure the Authenticity of the downloaded package A.TDM.DPS.D	M O	S	I, C**, Au

Asset (A) or Asset Container (AC)	Description	Owner†	Protection Level /Level of Sensitivity	Security Properties Required
A.TDM.SPC.K	Software Package Confidentiality Protecting Keys – cryptographic keys used to ensure the Confidentiality of the downloaded package A.TDM.DPS.D	M O	S	I, C**, Au
A.TDM.ACLS.D	Access Control List of the SCOMO Management Tree – Data Structure that defines which management commands each management server is allowed to perform on the nodes of the OMA DM SCOMO sub-tree	O	S	I, Au
A.TDM.DPS.D	Download Package containing the Software Installation Code or Software Update Code	M O	S	I, C, Au, NR
A.TDM.ROOT.K	Root Public Keys used by the TDM	M O	S	I, Au

Notes:

† Denotes that the given set of owners of a specific Asset is only indicative.

** Denotes the fact that the Confidentiality of these keys will depend on whether these keys are symmetric, Private or Public.

12.4.2 ASSETS GROUPING

In order to facilitate requirement understanding, the following grouping of Assets is defined:

- **TDM Code Assets** group contains all TDM code Assets: A.TDM.DL.C, A.TDM.FMA.C and A.TDM.SMA.C
- **TDM Data Assets** group contains all TDM data Assets: A.TDM.DISP.D, A.TDM.KB.D, A.TDM.ACLF.D, A.TDM.ACLS.D, A.TDM.DPF.D and A.TDM.DPS.D

- **TDM Key Assets** group contains all TDM key Assets: A.TDM.SA.K, A.TDM.FPA.K, A.TDM.FPC.K, A.TDM.SPA.K, A.TDM.ROOT.K and A.TDM.SPC.K

12.5 REQUIREMENTS

The Asset security properties referred to in the following requirements are defined in section 12.5.1.

Two Profiles, Profile 1 and Profile 2, are defined for Trusted Device Management. They are intended to be implemented in respectively a Profile 1 ME and a Profile 2 ME as defined in section 1.5, meaning that the firmware and software of a Profile 1 Handset must be managed according to requirements of Trusted Device Management Profile 1, and firmware and software of a Profile 2 Handset must be managed according to requirements of Trusted Device Management Profile 2.

12.5.1 TRUSTED DEVICE MANAGEMENT CORE REQUIREMENTS

In addition to TR1 Core Requirements defined in Chapter 3 "Trusted Environment Core requirements", further core requirements are defined for TDM in this chapter.

REQ. ID	REQUIREMENT	PROFILE 1	PROFILE 2
ATE-TDM-1860	The security properties of TDM Code Assets, Data Assets A.TDM.ACLF.D, A.TDM.ACLS.D, A.TDM.DPF.D, A.TDM.DPS.D and Key Assets SHALL be defended against Threats contained in Threat Group 1 "Hardware Modules Used for Accessing Memories"		✓
ATE-TDM-1870	The Confidentiality of TDM Key Assets SHALL be defended against Threats contained in Threat Group 2 "CLCD Used for Displaying Memories and Interfering With Displayed Data"		✓
ATE-TDM-1880	The security properties of TDM Code Assets, Data Assets A.TDM.ACLF.D, A.TDM.ACLS.D, A.TDM.DPF.D, A.TDM.DPS.D and Key Assets SHALL be defended against Threats contained in Threat Group 3 "Bypass Security by Removal of Battery Power or Removal of External Memory Card"	✓	✓

Table 29 summarises the TDM core requirements that apply to TDM Assets.

Table 29: Trusted Device Management Threat Grouping

Threat Group	1	2	3	4	5	6	7
Threats Assets	Hardware Modules Used for Accessing Memories	CLCD Used for Displaying Memories and Interfering With Displayed Data	Bypass Security by Removal of Battery Power or Removal of External Memory Card	Attack by Replacement of Flash When Power is Off (Pre-Boot)	Extract Secret via Bus Monitoring (Hardware Probes)	Mod Chip Attacks on Data in External RAM	Attack by Replacement of Flash When Power is On (Post-Boot).
A.TDM.DL.C A.TDM.FMA.C A.TDM.SMA.C	2	-	1, 2	1, 2	-	-	-
A.TDM.DISP.D A.TDM.KB.D	-	-	-	-	-	-	-
A.TDM.ACLF.D A.TDM.ACLS.D A.TDM.DPF.D A.TDM.DPS.D	2	-	1, 2	1, 2	-	-	-
A.TDM.SA.K A.TDM.FPA.K A.TDM.FPC.K A.TDM.SPA.K A.TDM.SPC.K A.TDM.ROOT.K	2	2	1, 2	1, 2	2 (C)	2	2

12.5.2 TRUSTED DEVICE MANAGEMENT SPECIFIC REQUIREMENTS

REQ. ID	SECURITY GOAL	TR1 REQUIREMENT	PROF. 1	PROF. 2
ATE-TDM-1890	The security properties of the Trusted Device Management Key Asset A.TDM.SA.K SHALL be guaranteed and A.TDM.SA.K SHALL only be accessible to the Download Agent Asset A.TDM.DL.C, the Firmware Management Agent A.TDM.FMA.C and the Software Management Agent A.TDM.SMA.C	If requiring Confidentiality protection, the Trusted Device Management Key Asset A.TDM.SA.K SHALL be considered as an SST Sensitive Object Asset and stored according to ATE-SST-0890 and SHALL only be accessible to the Download Agent A.TDM.DL.C, the Firmware Management Agent A.TDM.FMA.C and the Software Management Agent A.TDM.SMA.C. If the Trusted Device Management Key Asset A.TDM.SA.K is in the form of a Public Key within a certificate, the Root Public Key A.TDM.ROOT.K used to verify A.TDM.SA.K SHALL be considered as an SST Sensitive Object Asset and stored according to ATE-SST-0890 and the Integrity and validity of A.TDM.SA.K SHALL be verified via a full certificate chain verification before each use	✓	✓

REQ. ID	SECURITY GOAL	TR1 REQUIREMENT	PROF. 1	PROF. 2
ATE-TDM-1900	The security properties of the Trusted Device Management Keys Asset A.TDM.FPA.K SHALL be guaranteed and A.TDM.FPA.K SHALL only be accessible to the Firmware Management Agent A.TDM.FMA.C	The Trusted Device Management Key Asset A.TDM.FPA.K SHALL be considered as an SST Sensitive Object Asset and stored according to ATE-SST-0890 and SHALL only be accessible to the Firmware Management Agent A.TDM.FMA.C. If the Trusted Device Management Key Asset A.TDM.FPA.K is in the form of a Public Key within a certificate, the Root Public Key A.TDM.ROOT.K used to verify A.TDM.FPA.K SHALL be considered as an SST Sensitive Object Asset and stored according to ATE-SST-0890 and the Integrity and validity of A.TDM.FPA.K SHALL be verified via a full certificate chain verification before each use	✓	✓

REQ. ID	SECURITY GOAL	TR1 REQUIREMENT	PROF. 1	PROF. 2
ATE-TDM-1910	The security properties of the Trusted Device Management Keys Asset A.TDM.FPC.K SHALL be guaranteed and A.TDM.FPC.K SHALL only be accessible to the Firmware Management Agent A.TDM.FMA.C	The Trusted Device Management Key Asset A.TDM.FPC.K SHALL be considered as an SST Sensitive Object Asset and stored according to ATE-SST-0890 and SHALL only be accessible to the Firmware Management Agent A.TDM.FMA.C. If the Trusted Device Management Key Asset A.TDM.FPC.K is in the form of a Public Key within a certificate, the Root Public Key A.TDM.ROOT.K used to verify A.TDM.FPC.K SHALL be considered as an SST Sensitive Object Asset and stored according to ATE-SST-0890 and the Integrity and validity of A.TDM.FPC.K SHALL be verified via a full certificate chain verification before each use	✓	✓

REQ. ID	SECURITY GOAL	TR1 REQUIREMENT	PROF. 1	PROF. 2
ATE-TDM-1920	The security properties of the Trusted Device Management Keys Asset A.TDM.SPA.K SHALL be guaranteed and A.TDM.SPA.K SHALL only be accessible to the Software Management Agent A.TDM.SMA.C	The Trusted Device Management Key Asset A.TDM.SPA.K SHALL be considered as an SST Sensitive Object Asset and stored according to ATE-SST-0890 and SHALL only be accessible to the Software Management Agent A.TDM.SMA.C. If the Trusted Device Management Key Asset A.TDM.SPA.K is in the form of a Public Key within a certificate, the Root Public Key A.TDM.ROOT.K used to verify A.TDM.SPA.K SHALL be considered as an SST Sensitive Object Asset and stored according to ATE-SST-0890 and the Integrity and validity of A.TDM.SPA.K SHALL be verified via a full certificate chain verification before each use	✓	✓

REQ. ID	SECURITY GOAL	TR1 REQUIREMENT	PROF. 1	PROF. 2
ATE-TDM-1930	The security properties of the Trusted Device Management Keys Asset A.TDM.SPC.K SHALL be guaranteed and A.TDM.SPC.K SHALL only be accessible to the Software Management Agent A.TDM.SMA.C	The Trusted Device Management Key Asset A.TDM.SPC.K SHALL be considered as an SST Sensitive Object Asset and stored according to ATE-SST-0890 and SHALL only be accessible to the Software Management Agent A.TDM.SMA.C. If the Trusted Device Management Key Asset A.TDM.SPC.K is in the form of a Public Key within a certificate, the Root Public Key A.TDM.ROOT.K used to verify A.TDM.SPC.K SHALL be considered as an SST Sensitive Object Asset and stored according to ATE-SST-0890 and the Integrity and validity of A.TDM.SPC.K SHALL be verified via a full certificate chain verification before each use	✓	✓
ATE-TDM-1940	The security properties of the Download Agent A.TDM.DL.C, the Firmware Management Agent A.TDM.FMA.C and the Software Management Agent A.TDM.SMA.C SHALL be verified before these Applications are executed	The security properties of the Download Agent A.TDM.DL.C, the Firmware Management Agent A.TDM.FMA.C and the Software Management Agent A.TDM.SMA.C SHALL be verified before these Applications are executed. Verification at time of Secure Boot is acceptable if the Agents are Integrity Protected from boot-time to time of use	✓	✓

REQ. ID	SECURITY GOAL	TR1 REQUIREMENT	PROF. 1	PROF. 2
ATE-TDM-1950	The security properties of the FUMO Access Control List Asset A.TDM.ACLF.D SHALL be intrinsically guaranteed OR The Integrity of the FUMO Access Control List Asset A.TDM.ACLF.D SHALL be verified before use	The FUMO Access Control List Asset A.TDM.ACLF.D SHALL be considered as an SST Sensitive Object Asset and SHALL be stored according to ATE-SST-0890	✓	✓
ATE-TDM-1960	The FUMO Access Control List Asset A.TDM.ACLF.D SHALL only be modifiable by the Firmware Management Agent A.TDM.FMA.C	The FUMO Access Control List Asset A.TDM.ACLF.D SHALL be considered as an SST Sensitive Object Asset and SHALL only be modifiable by the Firmware Management Agent A.TDM.FMA.C	✓	✓
ATE-TDM-1970	The security properties of the SCOMO Access Control List Asset A.TDM.ACLS.D SHALL be intrinsically guaranteed OR The Integrity of the SCOMO Access Control List Asset A.TDM.ACLS.D SHALL be verified before use	The SCOMO Access Control List Asset A.TDM.ACLS.D SHALL be considered as an SST Sensitive Object Asset and SHALL be stored according to ATE-SST-0890	✓	✓

REQ. ID	SECURITY GOAL	TR1 REQUIREMENT	PROF. 1	PROF. 2
ATE-TDM-1980	The SCOMO Access Control List Asset A.TDM.ACLS.D SHALL only be modifiable by the software management agent A.TDM.SMA.C	The SCOMO Access Control List Asset A.TDM.ACLS.D SHALL be considered as an SST Sensitive Object Asset and SHALL only be modifiable by the software management agent A.TDM.SMA.C	✓	✓
ATE-TDM-1990	The Data I/O displayed on screen Asset A.TDM.DISP.D MAY be Integrity protected	The Data I/O displayed on screen Asset A.TDM.DISP.D MAY be managed by the SUIO feature and considered as an SUIO Sensitive Object (A.SUIO.DO.D)	✓	✓
ATE-TDM-2000	The Data I/O on keyboard Asset A.TDM.KB.D MAY be Integrity protected	The Data I/O on keyboard Asset A.TDM.KB.D MAY be managed by the SUIO feature and considered as an SUIO Sensitive Object (A.SUIO.DO.D)	✓	✓

REQ. ID	SECURITY GOAL	TR1 REQUIREMENT	PROF. 1	PROF. 2
ATE-TDM-2010	Once the Firmware Download Package has been downloaded by the Download Agent A.TDM.DL.C: • the Firmware Download Package A.TDM.DPF.D SHALL be considered as Flexible Secure Boot A.FSB.UPD.D Asset • the Firmware Management Agent A.TDM.FMA.C SHALL be considered as Flexible Secure Boot A.FSB.UPD.C Asset • the Firmware Package Authentication Key A.FPA.K SHALL be considered as Flexible Secure Boot A.FSB.VER.K Asset • the firmware update process SHALL conform with FSB Profile 1 Requirements	Once the Firmware Download Package has been downloaded by the Download Agent A.TDM.DL.C: • the Firmware Download Package A.TDM.DPF.D SHALL be considered as Flexible Secure Boot A.FSB.UPD.D Asset • the Firmware Management Agent A.TDM.FMA.C SHALL be considered as Flexible Secure Boot A.FSB.UPD.C Asset • the Firmware Package Authentication Key A.FPA.K SHALL be considered as Flexible Secure Boot A.FSB.VER.K Asset • firmware update process SHALL conform with FSB Profile 1 Requirements	✓	

REQ. ID	SECURITY GOAL	TR1 REQUIREMENT	PROF. 1	PROF. 2
ATE-TDM-2020	Once the Firmware Download Package has been downloaded by the Download Agent A.TDM.DL.C: • the Firmware Download Package A.TDM.DPF.D SHALL be considered as Flexible Secure Boot A.FSB.UPD.D Asset • the Firmware Management Agent A.TDM.FMA.C SHALL be considered as Flexible Secure Boot A.FSB.UPD.C Asset • the Firmware Package Authentication Key A.FPA.K SHALL be considered as Flexible Secure Boot A.FSB.VER.K Asset • the firmware update process SHALL conform with FSB Profile 2 Requirements	Once the Firmware Download Package has been downloaded by the Download Agent A.TDM.DL.C: • the Firmware Download Package A.TDM.DPF.D SHALL be considered as Flexible Secure Boot A.FSB.UPD.D Asset • the Firmware Management Agent A.TDM.FMA.C SHALL be considered as Flexible Secure Boot A.FSB.UPD.C Asset • the Firmware Package Authentication Key A.FPA.K SHALL be considered as Flexible Secure Boot A.FSB.VER.K Asset • the firmware update process SHALL conform with FSB Profile 2 Requirements		✓

REQ. ID	SECURITY GOAL	TR1 REQUIREMENT	PROF. 1	PROF. 2
ATE-TDM-2030	When the Software Download Package A.TDM.DPS.D contains an Application to be installed or updated, the Installation or update process SHALL conform with TEE Application Requirements Profile 1	When the Software Download Package A.TDM.DPS.D contains an Application to be installed or updated, the Installation or update process SHALL conform with TEE Application Requirements Profile 1, i.e. requirements in section 4.4.4 Application Lifecycle Requirements as applicable to Profile 1	✓	
ATE-TDM-2040	When the Software Download Package A.TDM.DPS.D contains an Application to be installed or updated, the Installation or update process SHALL conform with TEE Application Requirements Profile 2	When the Software Download Package A.TDM.DPS.D contains an Application to be installed or updated, the Installation or update process SHALL conform with TEE Application Requirements Profile 2		✓

13 MOBILE COMMERCE

13.1 MOBILE COMMERCE INTRODUCTION

This chapter describes the security requirements related to the Mobile Commerce use case. This chapter does not intend to cover all mobile commerce use cases, but is focused on proximity payment with the most sensitive part of the mobile commerce Application in the UICC connected to a mobile Application in charge to render the UI. This UI link is supported via an HTTP connection between respectively a web server in the UICC and a web browser in the Mobile Equipment.

This particular use case represents for the mobile operators, the state of the art for a mobile commerce architecture, based on the proximity payment architecture as recommended in the GSMA Mobile NFC technical guidelines [28].

The use cases are described in section 13.2.

The list of Actors and Actors' concerns is provided in section 13.3.

The list of Mobile Commerce Assets is defined in section 13.4.

The security requirements are defined in section 13.5.

13.1.1 BUSINESS RATIONALE

Many scenarios could be encompassed in the generic term Mobile Commerce, with very different business models. As the goal of this document is to describe the necessary requirements for e-commerce Applications, we will consider just a few scenarios that expect to be both representative and cover major security concerns.

The first scenario is the offline proximity transaction. Offline means that there is no interaction with a distant server to check the transaction. This scenario fulfils different elements to be considered:

- the mobile, which can include different parts, each of them having their own security properties to be addressed
- an interface (e.g. contactless)
- a reader (e.g. Point-Of-Sale (POS) in a payment use case).

Some use cases that could be considered in this scenario are:

- contactless proximity payment
- ticketing validation (e.g. for transportation etc.)

The second scenario is the online transaction.

"Online" means there is a network link and potential check with a distant server during the transaction.

Some use cases that could be considered in this scenario are:

- online payment transaction (high-value payment where a bank validates the payment, or Internet payment via account data exchange)
- frequent flyer program card embedded in the mobile

For any scenario, the lifecycle of the Applications and associated data shall be considered in this chapter, as there may be a need to install a new payment Application, to load a new version, or to delete the Application and its related data.

The Application set itself will be segmented in two main parts, i.e. two Assets, whatever the scenario: the backward Application part which validates the transaction and the forward Application which manages the interaction with the user. The goal is to distinguish security analysis of these two Application segments.

Note that the backward Application is on the Terminal for proximity transactions and on a back-end server for online transactions

13.2 Key Operator Use Cases

The key operator use cases are listed below.

13.2.1 Use Case 1 – Proximity Payment With User Interaction

Claire wants to use the payment Application stored in her UICC via her wallet Application stored in her Handset. This Application allows Claire to activate a proximity payment via contactless connectivity to a contactless EFTPOS.

Two sub-use cases are considered.

- A first sub-scenario where the user is activating the mobile payment Application thanks to a pin code activation
    - Claire selects her payment Application and her payment card using her wallet Application in her Handset
    - She enters the activation PIN code on her mobile
    - The transaction amount is then displayed on the EFTPOS
    - She places her mobile phone on the contactless hotspot until given a notification (e.g. a sound)
    - An EFTPOS ticket is generated

- A second sub-scenario is where the user needs to authorise the payment only if the amount is higher than a certain level, otherwise no activation PIN code is needed
    - The vendor enters the transaction amount on the EFTPOS
    - Claire places her contactless phone on the hotspot
    - If the amount is lower than a certain level, Claire keeps her mobile on the hotspot until given a notification (e.g. a sound) and the payment is delivered
    - If the amount is higher than this level, Claire has to activate her payment with a PIN validation and again places her contactless phone on the contactless EFTPOS
    - For both cases, a receipt can be generated by the EFTPOS
    - If the financial value over which transactions require user authorisation is checked by the mobile commerce Application on the ME, this value is Integrity protected on the ME and also checked at the server side

The two following diagrams show the synopsis for both sub-use cases.

Figure 8: Mobile Commerce Sub-Use Case 1

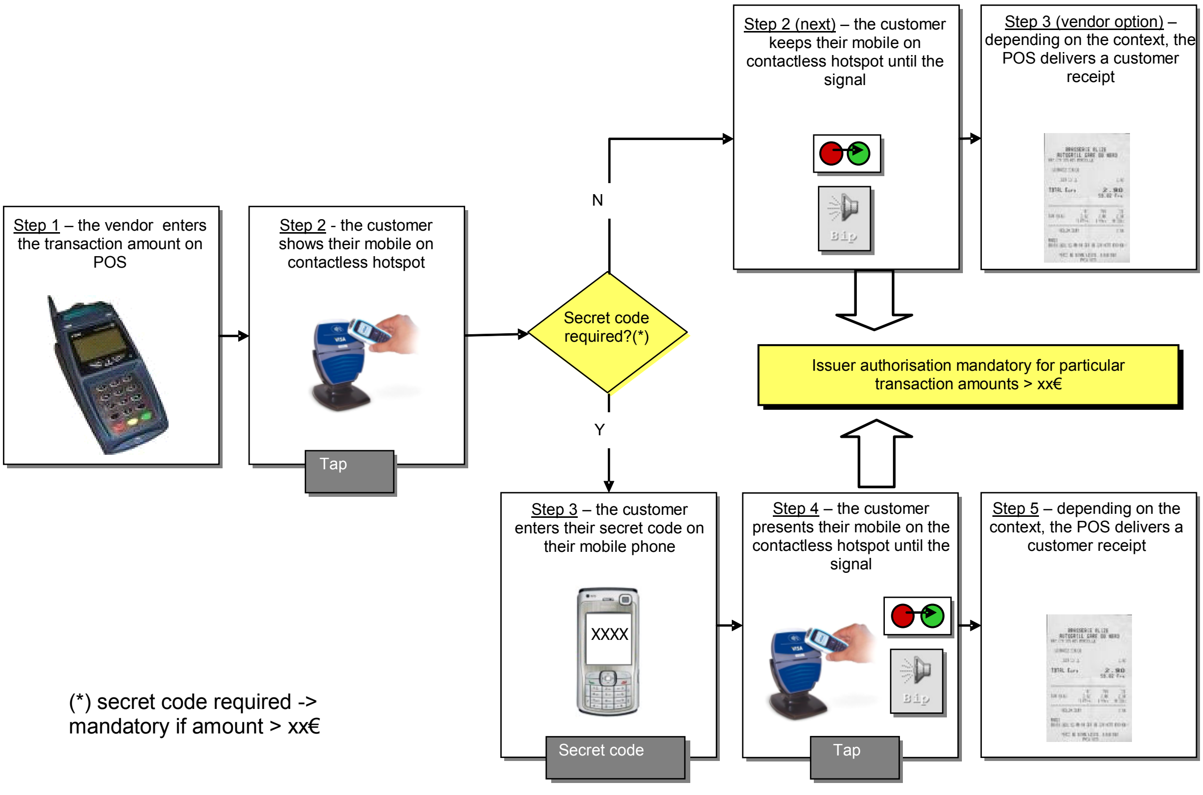
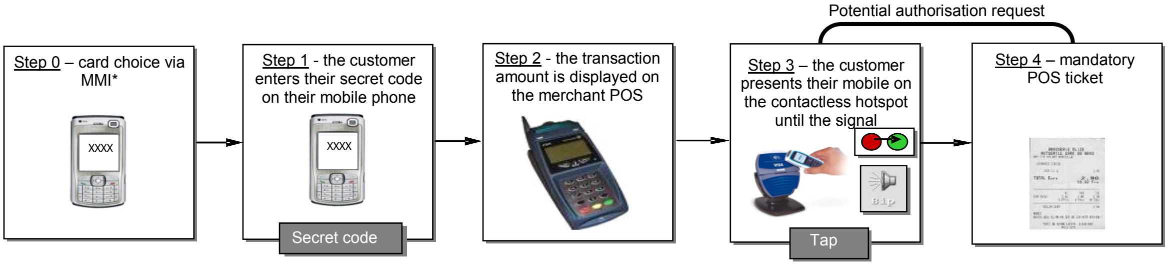

Step 1 – the vendor enters the transaction amount on POS
Step 2 - the customer shows their mobile on contactless hotspot
Tap
Secret code required?(*)
N
Y
Step 2 (next) – the customer keeps their mobile on contactless hotspot until the signal
Bip
Step 3 (vendor option) – depending on the context, the POS delivers a customer receipt
Issuer authorisation mandatory for particular transaction amounts > xx€
Step 3 – the customer enters their secret code on their mobile phone
XXXX
Secret code
Step 4 – the customer presents their mobile on the contactless hotspot until the signal
Bip
Tap
Step 5 – depending on the context, the POS delivers a customer receipt
(*) secret code required -> mandatory if amount > xx€

Figure 9: Mobile Commerce Sub-Use Case 2


Potential authorisation request
Step 0 – card choice via MMI*
XXXX
Step 1 - the customer enters their secret code on their mobile phone
XXXX
Secret code
Step 2 - the transaction amount is displayed on the merchant POS
Step 3 – the customer presents their mobile on the contactless hotspot until the signal
Bip
Tap
Step 4 – mandatory POS ticket
(*)if multi-card holder

Some abuse cases

- The link between web browser and UICC is Spoofed by a Handset Application and displays false information on the display
- A Man-In-The-Middle Attack is organised by a Handset Application which stores secrets entered by the user when he wants to enter sensitive information.

13.2.2 POSSIBLE FUTURE OPERATOR USE CASES

The following use cases are out-of-scope. They may however be addressed in a future release of this document.

13.2.3 USE CASE 1 – PROXIMITY PAYMENT WITHOUT USER INTERACTION/ONLINE PAYMENT TRANSACTION WITHOUT SECURE ACCESS TO PERIPHERALS

This use case is not considered in the mobile commerce chapter.

13.2.4 USE CASE 2 - SECURE TICKETING

This use case could cover:

- the management of the tickets
  - secure download of the tickets in the Device
  - rights management (date validity, occurrence, rights, ticket lifecycle).
- The access and the usage of the ticket by the end user

13.3 MOBILE COMMERCE ACTORS AND THEIR CONCERNS

Table 30 describes all Actors involved in Mobile Commerce, their main concerns, and the trust link between them.

Table 30: Mobile Commerce Actors and Their Concerns

Actor	Main Concerns (in this context)
Operators (O)	Operators wish to be able to offer payment Applications and services. Operators will trust the manufacturer on the security properties that are required from the lifecycle of the Application to its Execution
Financial institutions (FI)	Financial institutions will want any sensitive data managed by their Application to be protected from malicious software or hardware Attacks. They will also be concerned about the Availability of the Application. Financial institutions will have to trust the whole mobile architecture
Banks (remote server)	Banks will want any sensitive data on payment Applications to be protected from malicious software or hardware Attacks. They will also be concerned about the Availability of the Application. Banks could have particular demands about the provability and certification of the installed mobile payment Applications. Banks will have to trust the mobile architecture
Merchants (Mc)	Merchants want to be sure of the Authenticity and validity of the transaction, as they need to have no rejection on the payment. Merchants will also be concerned also about the Availability of the payment Application

Actor	Main Concerns (in this context)
End User (U)	The end user wants to be sure of the Confidentiality of their payment transaction. The end user will also be concerned by the Non-Replay of their payment, and by the Availability of the Application. The end user will have to trust all the banks which authorise transaction systems, including the intrinsic security properties of transaction systems including mobile, information display, transaction data, confirmation of the transaction and PIN access
Manufacturer (M)	The Handset manufacturer implements the Execution Environment to enable payment Applications to be installed, managed and executed
Attacker (A)	An Attacker may want to launch various Attacks against payment Applications; from sensitive user information disclosure, to Replay Attacks, for their own interests, through to Denial-of-Service Attacks
UICC Manufacturer (UM)	The UICC manufacturer is concerned about the protection of their sensitive keys, code and data (e.g. for intellectual property protection)

13.4 MOBILE COMMERCE ASSETS

13.4.1 ASSETS TO BE PROTECTED FOR MOBILE COMMERCE SERVICE

The following high level "Asset model" is assumed in developing the list of Assets:

- Certificates are defined in order to enable the mobile commerce Application (located on the ME) to authenticate the UICC and for the UICC to authenticate the mobile commerce Application; this Asset is named A.MCO.CERTIF.D
- The Certificates A.MCO.CERTIF.D are chained back to a Root Public Key named A.MCO.ROOT.K
- The mobile commerce Application is shared between its code (A.MCO.APP.C), its data (A.MCO.APP.D) and its keys (A.MCO.APP.K). This mobile commerce Application in the ME renders the UI of the payment Application and provides the interface to the mobile commerce Application in the UICC, if present.

- Data is exchanged between the user and the mobile commerce Application via the mobile keyboard (A. MCO.IN.D), and via the mobile display (A. MCO.OUT.D).
- Typical mobile commerce transactions will require a temporary session key to ensure secure communication between the mobile commerce Application and the UICC; this session key is called A.MCO.SESSION.K.
- The payment operation could be activated by user validation via (for example) a PIN entry code. This data/key required to activate the payment is named A.MCO.ACTPAY.D. A.MCO.ACTPAY.D is considered as more sensitive than the A.MCO.IN.D (e.g. protection against Replay Attack can be envisaged for this Asset).

13.4.2 MOBILE COMMERCE ASSET TABLE

The following Assets in the implementation of Mobile Commerce use case require protection.

Table 31: Mobile Commerce Assets

Asset (A) or Asset Container (AC)	**Description**	**Owner†**	**Protection Level /Sensitivity**	**Security Properties Required**
A.MCO.xxx with y=K for Key, y=D for Data, y=C for Code, y=H for Hardware		*See Actors List*	*Sensitive (S) Operational (O)*	*Integrity (I) Confidentiality (C). Authenticity (Au) Non-Replay (NR)*
A.MCO.ROOT.K	Root Public Keys used by the Mobile Commerce Application	O FI	S	I, Au
A.MCO.CERTIF.D	Digital certificates related to the payment system that allow the mobile to authenticate the UICC	O	S	I, Au
A.MCO.APP.C	Mobile Commerce Application Code in the ME (manages input/output between the user and the UICC)	O FI	S	I, Au
A.MCO.APP.D	Mobile Commerce Application Data in the ME (including data used to verify UICC identity)	O FI	S	I, Au

Asset (A) or Asset Container (AC)	Description	Owner†	Protection Level /Sensitivity	Security Properties Required
A.MCO.APP.K	Key used to prove mobile commerce Application identity	O FI	S	I, C, Au
A.MCO.SESSION.K	Mobile Commerce Application session key used to secure communication between the Application and UICC	O FI	S	I, C
A.MCO.ACTPAY.D	Data/Key required to activate the payment (e.g. pin code)	O FI U	S	I, C, NR
A. MCO.IN.D	Data I/O exchanged on Keyboard between the Mobile commerce Application and the UICC	M	S	I, Au, C, NR
A. MCO.OUT.D	Data output from the UICC to be displayed on screen	M	S	I, Au, NR

Note:

† Denotes that the given set of owners of a specific Asset is only indicative.

13.4.3 ASSETS GROUPING

In order to facilitate requirement understanding, the following grouping of Assets is defined:

- **MCO Code Assets** group contains all MCO code Assets: A.MCO.APP.C
- **MCO Data Assets** group contains all MCO data Assets: A.MCO.CERTIF.D, A.MCO.APP.D, A.MCO.ACTPAY.D, A. MCO.IN.D and A. MCO.OUT.D
- **MCO Key Assets** group contains all MCO key Assets: A.MCO.ROOT.K, A.MCO.APP.K and A.MCO.SESSION.K.

13.5 MOBILE COMMERCE REQUIREMENTS

In addition to TR1 Core Requirements defined in Chapter 3 “Trusted Environment Core requirements”, further requirements are defined for MCO below.

REQ. ID	SECURITY GOAL	TR1 REQUIREMENT	PROF. 1	PROF. 2
ATE-MCO-2050	A.MCO. ROOT.K, A.MCO.APP.C, A.MCO.APP.D and A.MCO.APP.K SHALL be protected against the Threats contained in Threat Group 3 and 4	A.MCO.APP.D SHALL be considered as A.EE.APP.D, A.MCO.APP.C SHALL be considered as A.EE.APP.C, A.MCO.APP.K and A.MCO.ROOT.K SHALL be considered as A.EE.APP.K, and these Assets SHALL be protected by a TEE	✓	✓
ATE-MCO-2060	A.MCO. ROOT.K, A.MCO.APP.C, A.MCO.APP.D and A.MCO.APP.K SHALL be protected against the Threats contained in Threat Group 1, 2, 5, 6 and 7	A.MCO.APP.D SHALL be considered as A.EE.APP.D, A.MCO.APP.C SHALL be considered as A.EE.APP.C, A.MCO.APP.K and A.MCO.ROOT.K SHALL be considered as A.EE.APP.K, and these Assets SHALL be protected by a TEE Profile 2		✓

REQ. ID	SECURITY GOAL	TR1 REQUIREMENT	PROF. 1	PROF. 2
ATE-MCO-2070	The Confidentiality of Mobile Commerce Application keys (A.MCO.APP.K) SHALL be defended against the Threats contained in Threat Group 2 “CLCD Used for Displaying Memories and Interfering With Displayed Data”. The Mobile Commerce Application data (A.MCO.OUT.D) Assets SHALL only be displayable at the request of the Mobile Commerce Application and SHALL otherwise be defended against the Threats contained in Threat Group 2, “CLCD Used for Displaying Memories and Interfering With Displayed Data”	Mobile Commerce Application keys (A.MCO.APP.K) SHALL be managed by a TEE and considered as Application Keys (A.EE.APP.K) requiring Confidentiality protection. A.MCO.APP.K SHALL be stored by a SST and considered as an SST Sensitive Object (SST.DO.D(C)). Data I/O from the UICC to be displayed on screen (A.MCO.OUT.D) SHALL be protected by a Secure Channel (SUM) when transmitted from the UICC to the A.MCO.APP.C		✓
ATE-MCO-2080	The security properties of the Mobile Commerce Application key (A.MCO.APP.K) SHALL be defended against the Threats contained in Threat Group 5 “Extract Secret via Bus Monitoring (Hardware Probes)”	Mobile Commerce Application key (A.MCO.APP.K) SHALL be defended against the Threats contained in Threat Group 5 “Extract Secret via Bus Monitoring (Hardware Probes)”		✓

REQ. ID	SECURITY GOAL	TR1 REQUIREMENT	PROF. 1	PROF. 2
ATE-MCO-2090	The security properties of the Mobile Commerce Application key (A.MCO.APP.K) SHALL be defended against the Threats contained in Threat Group 6 “Mod Chip Attacks on Data in External RAM”	Mobile Commerce Application key (A.MCO.APP.K) SHALL be defended against the Threats contained in Threat Group 6 “Mod Chip Attacks on Data in External RAM”		✓

Table 32 summarises the MCO core requirements that apply to MCO Assets.

Table 32: Mobile Commerce Threat Groups

Threat Group	1	2	3	4	5	6	7
Threats Assets	Hardware Modules Used for Accessing Memories	CLCD Used for Displaying Memories and Interfering With Displayed Data	Bypass Security by Removal of Battery Power or Removal of External Memory Card	Attack by Replacement of Flash When Power is Off (Pre-Boot)	Extract Secret via Bus Monitoring (Hardware Probes)	Mod Chip Attacks on Data in External RAM	Attack by Replacement of Flash When Power is On (Post-Boot).
A.MCO.APP.C A.MCO.APP.D A.MCO.ROOT.K	2	-	-	-	-	-	-
A.MCO.APP.K	2	2	-	-	1, 2	1, 2	-
A.MCO.SESSION.K A.MCO.MCO.ACTPAY.D A.MCO.IN.D A.MCO.CERTIF.D	-	-	-	-	-	-	-
A.MCO.OUT.D	-	2	-	-	-	-	-

REQ. ID	SECURITY GOAL	TR1 REQUIREMENT	PROF. 1	PROF. 2
ATE-MCO-2100	MCO data output displayed on screen (A.MCO.OUT.D) SHALL be Confidentiality and Integrity protected from other Applications	MCO data output displayed on screen (A.MCO.OUT.D) SHALL be managed by the SUIO feature and considered as an SUIO Sensitive Object (A.SUIO.DO.D), if the MCO is configured to require management by the SUIO feature	✓	✓
ATE-MCO-2110	MCO data input from Keyboard (A.MCO.IN.D) SHALL be Confidentiality and Integrity protected from other Applications	MCO data input from keyboard (A.MCO.IN.D) SHALL be managed by the SUIO feature and considered as an SUIO Sensitive Object, if the MCO is configured to require management by the SUIO feature	✓	✓
ATE-MCO-2120	The Assets A.MCO.IN.D, A.MCO.OUT.D, A.SESSION.K and A.MCO.ACTPAY.D SHALL be stored in a Security Preserving Memory and erased when no longer required.	The Assets A.MCO.IN.D, A.MCO.OUT.D, A.MCO.SESSION.K (if A.SESSION.K is used over more than one communications session of the Application) and A.MCO.ACTPAY.D SHALL be managed by an SST and considered as an SST Sensitive Object (SST.DO.D)	✓	✓

REQ. ID	SECURITY GOAL	TR1 REQUIREMENT	PROF. 1	PROF. 2
ATE-MCO-2130	MCO digital certificates related to the payment system that allow the mobile to authenticate the UICC (A.MCO.CERTIF.D) SHALL be Integrity protected against any unauthorised Application	MCO digital certificates related to the payment system that allow the mobile to authenticate the UICC (A.MCO.CERTIF.D) SHALL be verified for Integrity, Authenticity and validity before use	✓	✓
ATE-MCO-2140	Mobile Commerce Application (A.MCO.APP.C,) SHALL be checked at Installation and SHOULD be checked immediately prior to Execution	A.MCO.APP.C and A.MCO.APP.D MAY be checked by the Run-time Integrity Check feature	✓	✓

14 LOOKING FORWARD

OMTP continually develops and maintains its recommendations and requirements. Future versions of this document will address the following areas:

- Document maintenance
- Further use cases, for example:
    - **Application Security Framework Protection:** the protection of the mechanisms to restrict access to Device APIs as defined in OMTP Application Security Framework [3].
    - Others to be identified.
- Updates and additions to requirements as referenced standards evolve.
- More precise statements regarding enablers and use cases that require further investigations, like the protection of Generic Bootstrapping Architecture implementations.
- More precise statements in case of Generic Bootstrapping Architecture could cover the appliance of the OMTP Application Security Framework [3] to fulfil the requirements ATE-GBA-1360 and ATE-GBA-1380 to ATE-GBA-1420 inclusive for instance.

14.1 FORWARD LOOKING REQUIREMENTS

There are some requirements that were elicited in the generation of this recommendation which do not yet have standards to support them, but which will require standardisation to be actively implemented within mobile Devices. The following list of requirements are marked as 'FL' (Forward Looking) and are intended to give a clear indication of what will be expected in the future. For this reason, they are not included in the main document and are not mandatory. As standards become available to meet the forward looking requirements, they will be reassessed and may be formalised within maintenance releases and the 'FL' statement removed.

14.1.1 Forward Looking SST Application Assets Manager Requirements

Req. ID	Requirement	Profile 1	Profile 2
FL-ATE-SST-2150	The SST Application Assets Manager SHALL have the possibility to store Sensitive Objects in the UICC if requested by the Application calling it. Unless the Sensitive Objects are already protected by the SST Application Assets Manager, Sensitive Objects SHALL be sent to the UICC through a Secure Channel (as defined in chapter 10) using: - the TRANSACT DATA command (see section 7.4 and section 9.1 of ETSI TS 102 484 [21]) in the case of a secured APDU protocol OR - the TLS record protocol defined in RFC 4346 [30] in the case of a TLS protocol. After the establishment of a Secure Channel, all relevant traffic with the USIM would be encrypted to guarantee secure storage. Note: The TRANSACT DATA command is defined in ETSI TS 102 221 [31], section 11.1.21	✓	✓
FL-ATE-SST-2160	The SST Application Assets Manager SHALL be able to use a (compatible) UICC as an alternate Security-Preserving Memory.		

REQ. ID	REQUIREMENT	PROFILE 1	PROFILE 2
FL-ATE-SST-2170	An Application running on the UICC SHALL have the possibility to call the SST Application Assets Manager by requesting for storage of Sensitive Object in Security-Preserving Memories (either AC.SST.IPM.H, AC.SST.ICPM.H or AC.SST.ICM.H) on the ME. The Sensitive Objects SHOULD be sent from the UICC to the SST Application Assets Manager through a Secure Channel (as defined in chapter 10)	✓	✓
FL-ATE-SST-2180	The SST Application Assets Manager SHALL be able to validate, via the Execution Environment, the identity of the UICC Application calling it. Mechanisms defined in chapter 10 SHOULD be used	✓	✓

14.1.2 Forward Looking SST Key Manager Requirements

REQ. ID	REQUIREMENT	PROFILE 1	PROFILE 2
FL-ATE-SST-2190	When required by the Application calling the Secure Storage facility, the SST Key Manager SHALL be able to uniquely bind the Sensitive Objects to the UICC	✓	✓

14.1.3 Forward Looking GBA Requirements

Req. ID	Requirement	Profile 1	Profile 2
FL-ATE-GBA-2200	If there are applicable ACPs stored in the UICC and the Terminal, then ACPs stored in the UICC SHALL take precedence over ACPs stored in the Terminal, as requested in 3GPP TR 33.905 –'Recommendations for Trusted Open Platform (Release 7)'[19]. ACP in UICC is defined in OMA TS SCWS section 12, and can be accessed from the Terminal using HTTP GET command	✓	✓

14.1.4 Forward Looking SUIO Requirements

Req. ID	Requirement	Profile 1	Profile 2
FL-ATE-SUIO-2210	The SUIO SHALL be able to apply a Terminal Binding of sensitive user input/output data to the UICC	✓	N/A
FL-ATE-SUIO-2220	The SUIO SHALL be able to apply a UICC Binding of sensitive user input/output data to the UICC	✓	N/A
FL-ATE-SUIO-2230	An Application running on the UICC SHALL have the possibility to call the SUIO by requesting use of secure user input and output devices. The Sensitive Objects SHOULD be transfer from/to the UICC to/from the SUIO through a Secure Channel (as defined in chapter 10)	✓	N/A
FL-ATE-SUIO-2240	The SUIO Application Assets Manager SHALL be able to validate, via the Execution Environment, the identity of the UICC Application calling it. Mechanisms defined in chapter 10 SHOULD be used	✓	N/A

15 Definition of Terms

Term	Description
Access Control	Security mechanism which guarantees Asset security properties by restricting the ability to use (read, execute, modify or delete) an Asset
Actor	Entity playing a role in the ME value chain. It can be for example an operator, a service or a content provider, the end user, the Terminal or UICC manufacturer
Application	This term is used to cover both active software components such as executables and libraries, as well as more passive components such as content and scripts
Application Assets Manager	The software component of the Secure Storage facility responsible for managing Sensitive Objects
Asset	Resource whose security properties need to be protected
Asset Container	Resource used to store or transfer other Assets
Attack	An intentional act attempting to violate the UE Security Policy
Attacker	Anyone or anything that attempts to perform an Attack
Authentication Vector	A triplet/quintet of parameters for user authentication in GSM/UMTS
Authenticity	The property that a legitimate user can correctly identify an Asset as being genuine and attributable to its authors or caretakers (definition updated from ‘OMTP Trusted Environment: OMTP TR0’ [1])
Authorised Party	An entity that is authorised to perform a specific operation on Assets. This authorisation depends on the Security Policy of the Asset’s owner (definition updated from ‘OMTP Trusted Environment: OMTP TR0’ [1])

TERM	DESCRIPTION
AVAILABILITY	The property of an Asset that it is accessible when required by an Authorised Party
BUILT-IN SELF-TEST	Any mechanisms within the ME that allow diagnostic or repair operations to take place
BUFFER OVERFLOW	A Buffer Overflow is an erroneous condition where a process attempts to store data beyond the boundaries of a fixed-length buffer. The result is that the extra data overwrites adjacent memory locations
CREDENTIAL MANAGEMENT	Software component of the Execution Environment that provides facilities to identify users and processes and assigns them credentials and permissions so that Access Control policies may be enforced
CONFIDENTIALITY	The property of an Asset that information about its value can only be acquired by an Authorised Party (definition updated from 'OMTP Trusted Environment: OMTP TR0' [1])
CRYPTOGRAPHICALLY-PROTECTED DATA	Data whose security properties are fully or partially protected via cryptographic mechanisms
DEFRAGMENTATION	The process by which fragmented areas of technology and specification are brought together to ensure a set of unified, solid and coherent requirements, forming part of a recommendation
DENIAL-OF-SERVICE	An Attack on a computer system or network that causes a loss of service to users
DEVICE	Equivalent to Handset, Mobile Equipment or Terminal
EMERGENCY CODE BASE	In FSB, within a configuration where there are two distinct code images that can be executed on the Terminal, the first code image is the main code base that is normally executed on the Terminal; the second code image is the Emergency Code Base. The Emergency Code Base will be executed if there is a reason that prevents the main code base from being executed
EXECUTION	Occurs after Installation and Loading. The point in time begins when the ME starts the software

TERM	DESCRIPTION
EXECUTION ENVIRONMENT	A set of hardware and software components that provide facilities (e.g. Computing, Memory Management, input/output, etc.), necessary to support Applications
FIRMWARE OVER THE AIR	The ability to modify the operating software of the ME via an over-the-air mechanism
FLASH	A type of memory device which offers non-volatile storage. Also, the process of re-programming a system (see also Flashing)
FLASHING	The process of re-programming a system
FLEXIBLE SECURE BOOT	Refers specifically to the OMTP FSB function as well as generically to any Secure Boot process that initiates a secure primary RIC
GBA CLIENT	Functional entity in the UE communicating as a gateway with the UICC and with the BSF during the bootstrapping procedure
GBA MASTER SECRET	The key Ks obtained at the end of the bootstrapping procedure, used to derive the NAF-specific key
GBA SERVER	Functional entity in the UE communicating as a gateway with the GBA Server and with the NAF during the bootstrapping procedure
GENERAL PURPOSE MEMORY	Any memory that is not a Security-Preserving Memory (defined as non-secure memory in 'OMTP Trusted Environment: OMTP TR0' [1])
HANDSET	Equivalent to Device, Mobile Equipment or Terminal
HARDWARE UNIQUE KEY	A unique ME-specific key used for internal security mechanisms, as defined in the 'OMTP Trusted Environment: OMTP TR0' [1]
HASH	A small value which is the resultant of data which has had a mathematical function applied to it

TERM	DESCRIPTION
HOME SUBSCRIBER SERVER	UMTS Database containing the subscription-related information, performing authentication and authorisation of the user, and providing information about the physical location of user (similar to the GSM HLR and AUC)
INTERNATIONAL MOBILE EQUIPMENT IDENTITY	A unique number given to every single Terminal typically found behind the battery. IMEI numbers of cellular phones connected to a GSM network are stored in a database register showing the allocation to a particular manufacturer for use with a particular model. The uniqueness of the IMEI allows the identification of individual MEs allowing operators to control access to GSM and UMTS networks based on ME types or individual units. The IMEI is used for security purposes in case of theft or if there are technical problems with a mobile network. When a mobile phone is reported stolen or is not type-approved, the IMEI number can be blacklisted
INSTALLATION	The point when a package of software (that may contain one or more items of code and data) is integrated with the EE prior to Loading and Execution. Typically the software package will be resident in low speed memory, such as a file system, after Installation
INTEGRITY	The property of an Asset that it has been modified only by an Authorised Party (definition updated from the ‘OMTP Trusted Environment: OMTP TR0’ [1])
INTEGRITY-CHECKED MEMORY	Any memory whose Integrity is checked at run-time, but not necessarily before every memory access. There can be a delay between the time when the memory Integrity is compromised and the time of detection
INTEGRITY AND CONFIDENTIALITY-PROTECTED MEMORY	Any memory that protects the Integrity and Confidentiality of its content against the Attacks described in the Threat Model, for the minimum time specified by the owner of contents stored in it (defined as Secure Memory in ‘OMTP Trusted Environment: OMTP TR0’ [1])

TERM	DESCRIPTION
INTEGRITY-PROTECTED MEMORY	Any memory that protects the Integrity of its contents against the Attacks described in the Threat Model, for the minimum time specified by the owner of contents stored in it
INTEGRITY-PROTECTION SECURE BOOT	In FSB, a method of ensuring the Integrity of the code base by storing the code base in Integrity-Protected Memory
INTEGRITY-VERIFICATION SECURE BOOT	In FSB, a method of ensuring the Integrity of the code base by verifying the Integrity of the code base at system boot time
KEY ESTABLISHMENT	The security features and mechanisms that provision a shared key between a UICC and a Terminal that may host the UICC or be connected to the device hosting the UICC via a local interface
KEY LOGGER	A program that runs in background that captures the user's keystrokes. They can be very hard to detect, if not listed in the process list. Key Loggers are able to detect and store sensitive information such as login details, passwords and credit card numbers
KEY MANAGER	Application in charge of managing cryptographic keys. The responsibilities of a Key Manager could include the provisioning, exchange, safeguarding, use, vetting and replacement of keys
LOADING	The process of moving installed code into high speed Execution memory. This could occur a long time prior to Execution. In some Terminals this phase is just a matter of preparing the EE with the Execution memory that is the same as the Installation memory
MAN-IN-THE-MIDDLE ATTACK	A method of active eavesdropping in which the Attacker makes separate connections with the targets and relays messages between them. This allows the Attacker to have complete control of the communication between the targets
MOBILE EQUIPMENT	Equivalent to Device, Terminal or Handset

TERM	DESCRIPTION
MOD CHIP	Equivalent to Modification Chip. Small electronic device used to modify or disable built-in restrictions and limitations of a system
NONCE	A number or string used only once
NON-REPLAY	The property of an Asset of being resistant to replacement by a previous version of that Asset
PEER APPLICATIONS	Applications running on different EEs with an established communication channel between them
PEER EXECUTION ENVIRONMENTS	Distinct EEs between which a communication channel is established between for Applications running on these EEs
PHARMING	An Attack attempting to steal personal/private information using a malicious web site that impersonates a legitimate one (by sending false information on Internet DNS servers)
PHISHING	An Attack attempting to acquire sensitive information by masquerading as a trustworthy entity in an electronic communication, typically email or instant messaging
PHYSICAL MEMORY	Installed hardware memory
PLATFORM SOFTWARE	Software that will be executed following a successful Secure Boot
PRESENCE	A Presence service allows a user to make their status available to others and the statuses of others available to them
PRIVATE KEY	The Private Key portion of a key pair (Public Key, Private Key) used in an asymmetric cryptography system. The Private Key is kept secret, while the Public Key may be widely distributed. The keys are related mathematically, but the Private Key cannot be practically derived from the Public Key

TERM	DESCRIPTION
PROFILE 1	Profile 1 enablers require defence against software Attacks, some basic hardware Attacks, and some hardware probing Attacks for confidential key material. Profile 1 enablers can be used to support basic security services such as an Application Security Framework such as is described in the 'OMTP Application Security Framework' [3] or to manage I/O in security-demanding use cases (as for instance defined in Part II)
PROFILE 2	Profile 2 enablers require defence against the Attacks considered in Profile 1 as well as more sophisticated hardware Attacks such as probing on external memory. Profile 2 has to be considered from a security point of view as a superset of the Profile 1. Profile 2 enablers would typically be used to protect or provide security facilities for high value services such as M-commerce
PUBLIC KEY	The Public Key portion of a key pair (Public Key, Private Key) used in an asymmetric cryptography system. The Private Key is kept secret, while the Public Key may be widely distributed. The keys are related mathematically, but the Private Key cannot be practically derived from the Public Key
REMOVABLE DEVICE MANUFACTURER	Manufacturer of removable storage devices such as SD Cards or Memory Sticks
REPLAY ATTACK	A form of network Attack in which a valid data transmission is maliciously or fraudulently repeated or delayed
RIGHTS OBJECT	A collection of permissions and other attributes which are linked to protected content
ROBUSTNESS	Strength of a security function, mechanism, service or solution, and the confidence that it is implemented and functioning correctly
ROOT KEY	Public Key pair (public/private) of the certificate authority
ROOT PUBLIC KEY	Public Key attached to the Root Key

TERM	DESCRIPTION
SECRET KEY	A shared Secret Key value held by two (or more) communicating parties for use in a symmetric cryptosystem
SECURE BOOT	The establishment of a chain-of-trust to prevent modified code by an Attacker being loaded on a Terminal at start-up
SECURE CHANNEL	In cryptography, a method or technique assumed to provide a means by which data can be transferred from one place or user to another without risk of interception or tampering
SECURE STORAGE	A facility that receives Sensitive Objects and securely stores them as long as is required in memories, in such a way that it maintains the security properties (Integrity and/or Confidentiality) of these Assets
SECURITY POLICY	The set of rules and procedures for the protection of Assets' security properties
SECURITY-PRESERVING MEMORY	Encompasses Integrity-Checked Memory, Integrity-Protected Memory and Integrity and Confidentiality-Protected Memory
SENSITIVE OBJECTS	A data object that has security properties that need to be maintained by the Secure Storage facility
SERVICE LOCATION FUNCTION	See Subscriber Locator Function
SMART CARD	Tamper-resistant device (including trusted-by-the-operator memory and a trusted-by-the-operator Execution Environment) that can communicate with the Terminal through its interface. The operators issue Smart Cards in the form of Security or User Identification modules. Examples of Smart Cards are: SIM (GSM), R-UIM (CDMA), UICC (UMTS) with the USIM as an application
SUBSCRIBER LOCATOR FUNCTION	Subscription Locator Function or Service Location Function is used as a resolution mechanism to enable network entities to find the address of the HSS that holds the subscriber data for a given user identity

TERM	DESCRIPTION
SPOOF	The imitation of one identity by another
TERMINAL	Used as an alternative term for a cellular telephone or Handset
TERMINAL BINDING	Unique association of a data to an ME or ME sub-system
THREAT	Capabilities, intentions and Attack methods of adversaries, or any circumstance or event with the potential to violate or bypass the UE Security Policy
THREAT AGENT	Any human user or information technology (IT) product or system, which may attempt to violate or bypass the UE Security Policy and perform an unauthorised operation with the UE
THREAT MODEL	The list of Threats attempting to violate security properties of given Assets, and associated risk assessment
TRUSTED ENVIRONMENT	An Environment which meets a set of defined security criteria to protect its Assets and Execution. These security criteria are defined in the 'OMTP Trusted Environment: TR0' [1] and this document, the OMTP Advanced Trusted Environment: TR1. The meeting of these requirements provides a level of trust in the operation of that Environment
TRUSTED EXECUTION ENVIRONMENT	A Profile 1 or 2 Trusted Execution Environment is an Execution Environment that meets TR1 Profile 1 or 2 requirements
TRUSTED IDENTITIES	Application identities received either: over a Secure Channel from a TEE, or from inside the same TEE as the SST Application Assets Manager
TYPE-SAFE	Code that accesses only the memory locations it is authorised to access, and only in well-defined, allowable ways
UICC BINDING	Unique association of a data to an UICC

TERM	DESCRIPTION
UNINSTALLATION	The removal of a package of software (that may contain one or more items of code and data) from an integrated state with the EE, such that it is no-longer ready to be loaded and executed. It may, or may not, include the removal of Device specific information created by the original Installation or the Execution of the package (e.g. game saves). It may, or may not, include removal of the software package from the file system
UNTRUSTED IDENTITIES	Application identities received from another EE or TEE over an insecure channel
UPGRADE	Typically performed by the partial Uninstallation of a software package, and the Installation of all or some of a replacement package covering all or some of the total package components
VIRTUAL MACHINE	A piece of software that creates a virtualized Environment between the platform and its operating system, so that software can be executed on an abstract machine
WHAT YOU SEE IS WHAT YOU SIGN	A requirement for electronic signature software meaning that the software must show you the content in its entirety before you sign it

16 ABBREVIATIONS

ABBREVIATION	DESCRIPTION
2G	2nd Generation mobile phone technology (e.g. GSM)
3GPP	3rd Generation Partnership Project
AC	Asset Container
ACL	Access Control List
ACP	Access Control Policy
AES	Advanced Encryption Standard
AKA	Authentication and Key Agreement
API	Application Programming Interface
ASF	Application Security Framework
AUC	AUthentication Centre
AUTN	Authentication Token – used in UMTS for network authentication
AV	Authentication Vector
BC	BroadCast
BCA	BroadCAst
BCAST	BroadCAST Protocol
BIST	Built-in Self Test
BM-SC	MBMS Broadcast-Multicast Service Centre
BSF	Bootstrapping Server Function
BSM	BCAST Subscription Management
BSP	Broadcast Service Provider
BTID	Bootstrapping Transaction Identifier

ABBREVIATION	DESCRIPTION
CDMA	Code Division Multiple Access
CEK	Content Encryption Key
CK	Cipher Key
CLCD	Colour LCD controller
COR	COre Requirements
CP	Content Provider
CPU	Central Processing Unit
CRAM	Challenge Response Authentication Mechanism
CSIM	Cdma2000 Subscriber Identity Module
DM	Device Management
DMA	Direct Memory Access
DNS	Domain Name Server
DoS	Denial-of-Service
DRAM	Dynamic Random Access Memory
DRM	Digital Rights Management
ECC	Elliptic Curve Cryptography
EE	Execution Environment
EFTPOS	Electronic Funds Transfer at Point Of Sale
EICTA	European Information & Communications Technology Industry Association
ESG	Electronic Service Guide
ESN	Electronic Serial Number
ETSI	European Telecommunications Standard Institute
FOTA	Firmware Over The Air

ABBREVIATION	DESCRIPTION
FQDN	Fully Qualified Domain Name
FSB	Flexible Secure Boot
FTA	Free-To-Air
FUMO	Firmware Update Management Object
GAA	Generic Authentication Architecture
GBA	Generic Bootstrapping Architecture
GBA_ME	ME-based GBA
GBA_U	GBA with UICC-based enhancements
GSM	Global System for Mobile Communications
GSMA	GSM Association
HLR	Home Location Register
HMAC	Hash function based Message Authentication Code
HSS	Home Subscriber Server
HTTP	Hyper Text Transport Protocol
HTTPS	Secure HTTP
HUK	Hardware Unique Key
IK	Integrity Key
IMEI	International Mobile Equipment Identity
IMS	IP Multimedia Subsystem
I/O	Input/Output
IP	Internet Protocol
IPC	Inter-Process Communication
ISA	Instruction Set Architecture

ABBREVIATION	DESCRIPTION
ISIM	IP Multimedia Services Identity Module
JTAG	Joint Test Action Group
Kc	Ciphering Key – the session key for the air interface encryption in GSM
LED	Light-Emitting Diode
LTKM	Long Term Key Message
MBMS	Multimedia Broadcast Multicast Service
MCO	Mobile Commerce
ME	Mobile Equipment
MNO	Mobile Network Operator
MMU	Memory Management Unit
MO	Management Object
MRK	MBMS Request Key
MUK	MBMS User Key
NAF	Network Application Function
NFC	Near Field Communication
NR	Non-Replay
OMA	Open Mobile Alliance
OMTP	Open Mobile Terminal Platform
OS	Operating System
OSGi	Open Services Gateway Initiative
OTA	Over-The-Air
PCB	Printed Circuit Board
PEK	Program Encryption Key

ABBREVIATION	DESCRIPTION
PIN	Personal Identification Number
POS	Point-of-Sale
PSK	Pre-Shared Key
RAM	Random Access Memory
RAND	RANDom number (used for authentication)
RES	RESponse (in GBA)
RIC	Run-time Integrity Checking
RO	Rights Object
ROM	Read-Only Memory
RSA	Rivest-Shamir-Adleman: a Public Key cipher, named for its inventors, that can be used both for encrypting messages and making digital signatures
R-UIM	Removable User Identity Module
SCOMO	Software Component Management Object
SCK	Smart Card Key
SCP	Smart Card Platform (ETSI group)
SCWS	Smart Card Web Server
SEK	Service Encryption Key
SHA-256	Secure Hash Algorithm with a 256 bit footprint
SIM	Subscriber Identity Module
SLF	Subscription Locator Function or Service Location Function
SMK	Subscriber Management Key
SoC	System-on-Chip
SP	Service Provider

ABBREVIATION	DESCRIPTION
SRES	Signed RESponse
SRK	Subscriber Request Key
SST	Secure Storage Facility
STK	SIM Toolkit
STKM	Short Term Key Message
SUIO	Secure User Input and Output
SUM	Secure Interaction of UICC with Mobile
SW	Software
SYNCML	Synchronization Markup Language
TCG	Trusted Computing Group
TBK	Terminal Binding Key
TDM	Trusted Device Management
TEE	Trusted Execution Environment
TEK	Traffic Encryption Key
TLS	Transport Layer Security
TR	Trusted Environment
TR0	OMTP Trusted Environment: OMTP TR0 [1]
TR1	OMTP Advanced Trusted Environment: OMTP TR1
UA	User Agent
UART	Universal Asynchronous Receiver Transmitter
UE	User Equipment
UI	User Interface
UIM	User Identity Module

ABBREVIATION	DESCRIPTION
UICC	Universal Integrated Circuit Card
UMTS	Universal Mobile Telecommunications Service
UIO	User Input/Output
USIM	Universal Subscriber Identity Module
VGA	Video Graphics Array
WYSIWYS	What You See Is What You Sign
XRES	EXpected user RESponse

17 Referenced Documents

No.	Document	Author	Date
1	OMTP Trusted Environment: OMTP TR0 v1.2 http://www.omtp.org/Publications/Display.aspx?Id=f45e6775-2ecf-40fa-8cf2-dbe182ee9b58	OMTP	May 2009
2	Security Threats on Embedded Consumer Devices v1.1 http://www.omtp.org/Publications/Display.aspx?Id=39fc8a36-b440-4096-abcb-bcf5ae221e3f	OMTP	May 2009
3	OMTP Application Security Framework v2.2 http://www.omtp.org/Publications/Display.aspx?Id=c4ee46b6-36ae-46ae-95e2-cfb164b758b5	OMTP	Mar 2008
4	RFC 2119 - Key words for use in RFCs to Indicate Requirement Levels http://www.ietf.org/rfc/rfc2119.txt	IETF	Mar 1997
5	GPD/STIP device Application Security Management http://www.globalplatform.org	GlobalPlatform	
6	NIST, “Guideline for implementing cryptography in the federal government”, SP 800-21. http://csrc.nist.gov/publications/nistpubs/800-21-1/sp800-21-1_Dec2005.pdf	NIST	Dec 2005
7	NIST, “Recommendation for Random Number Generation Using Deterministic Random Bit Generators”, SP 800-90, Draft. http://csrc.nist.gov/publications/drafts/sp800-90_draft_dec2005.pdf	NIST	Dec 2005

No.	Document	Author	Date
8	NIST, "Recommendation for key management – Part 1: General Guideline", SP 800-57. http://csrc.nist.gov/CrytpotToolkit/tkkeymgmt.html	NIST	Apr 2005
9	Austrian recommendations for electronic signatures, RTR-GmbH, « Empfohlene Algorithmen und Parameter fur elektronische Signaturen », http://www.a-sit.at/pdfs/rtr-algorithms-20050301-de.pdf	A-SIT	Mar 2005
10	Federal Network Agency for Electricity, Gas, Telecommunications, Post and Railway, "Notification in accordance with the Electronic Signatures Act and the Electronic Signatures Ordinance (overview of suitable algorithms)". http://www.bundesnetzagentur.de/media/archive/5171.pdf	The German Federal Network Agency	Jan 2006
11	SGDN/DCSSI, « Règles et recommandations concernant le choix et le dimensionnement des mécanismes cryptographiques de niveau de robustesse standard », Version 1.02, N° 2791/SGDN/DCSSI/SDS/. http://www.ssi.gouv.fr/fr/politique_produit/catalogue/inventaire/pdf/Meca_crypto_v1.02.pdf	DCSSI	Nov 2004
12	OMTP UICC/(U)SIM Defragmentation and Requirements: OMTP UICC v2.2 http://www.omtp.org/Publications/Display.aspx?Id=4f9ec3d3-c0a7-4875-9458-0156cb9df3c9	OMTP	Aug 2008
13	3GPP TS 33.220 v7.7.0 "Generic Authentication Architecture (GAA); Generic bootstrapping architecture (Release 7)"	3GPP	Mar 2007

No.	Document	Author	Date
14	3GPP TS 33.102 v7.1.0 “3G Security; Security architecture (Release 7)”	3GPP	Dec 2006
15	3GPP TS 33.246 v7.3.0 “3G Security; Security of Multimedia Broadcast/Multicast Service (MBMS) (Release 7)”	3GPP	Mar 2007
16	3GPP TS 33.110 v7.1.0 “Key establishment between a Universal Integrated Circuit Card (UICC) and a terminal (Release 7)”	3GPP	Mar 2007
17	OMA Candidate Version 1.0 “Service and Content Protection for Mobile Broadcast Services” OMA-TS-BCAST_SvcCntProtection-V1_0, October 2008	OMA BCAST	October 2008
18	3GPP TS 33.222 v7.2.0 “Generic Authentication Architecture (GAA); Access to network Application functions using Hypertext Transfer Protocol over Transport Layer Security (HTTPS) (Release 7)”	3GPP	Oct 2006
19	3GPP TR 33.905 v7.7. 0 “Recommendations for Trusted Open Platform (Release 7)”	3GPP	Mar 2007
20	ETSI TS 102.412: Smart Card Platform Requirements	ETSI	Mar 2007
21	ETSI TS 102 484: Secure Channel between a UICC and an End Point Terminal v.7.3.0	ETSI	Jan 2009

No.	Document	Author	Date
22	Open Mobile Alliance Device Management v1.2 http://www.openmobilealliance.org/release_program/dm_v1_2.html	OMA	Jun 2006
23	Open Mobile Alliance Firmware Update Management Object V1.0 http://www.openmobilealliance.org/release_program/fumo_v1_0A.html	OMA	Feb 2007
24	Open Mobile Alliance Software Component Management Object Candidate version 1.0 http://www.openmobilealliance.org/release_program/docs/CopyrightClick.asp?pck=RD&file=OMA-RD-SCOMO-V1_0-20070731-C.pdf	OMA	Jul 2007
25	Open Mobile Alliance Client Provisioning v1.1 http://www.openmobilealliance.org/release_program/cp_v1_1.htm	OMA	May 2005
26	OSGi R4 Mobile V4.0 Specification http://www.osgi.org	OSGi	Sep 2006
27	GlobalPlatform Device Specifications http://www.globalplatform.org/specificationview.asp?id=device	GlobalPlatform	
28	GSMA Mobile NFC technical guidelines: http://www.gsmworld.com/documents/gsma_nfc_tech_guide_vs1.pdf	GSMA	Apr 2007
29	ETSI TS 102 223 Card Application Toolkit (CAT) Release 8, v8.1.0	ETSI	Oct 2008
30	RFC 4346 The Transport Layer Protocol (TLS) v1.1 http://www.ietf.org/rfc/rfc4346.txt	IETF	
31	ETSI TS 102 221 Smart Cards; UICC-Terminal Interface; Physical and Logical Characteristics v.7.14.0	ETSI	April 2009

18 APPENDIX 1- CONSOLIDATED TR1 ASSETS TABLE

ASSET (A) OR ASSET CONTAINER (AC)	DESCRIPTION	CHAPTER
A.EE.HUK.K	Hardware Unique Key	Trusted Execution Environments
A.EE.C	Execution Environment Code	Trusted Execution Environments
A.EE.D	Execution Environment Data (multiple Application running contexts: registers, etc.)	Trusted Execution Environments
A.EE.K	Execution Environment Keys	Trusted Execution Environments
A.EE.APP.C	Applications Code	Trusted Execution Environments
A.EE.APP.D	Applications Data	Trusted Execution Environments
A.EE.APP.K	Applications Keys	Trusted Execution Environments
A.SST.DO.D	Sensitive Objects: Objects provided to the SST by the calling Application. It can represent any information (data, code, keys), but is treated as opaque data by the Secure Storage facility. Requirements on this Asset will apply when this Asset under the control of the SST	Secure Storage
A.SST.AAM.C	SST Application Assets Manager: Code of the SST that manages Sensitive Objects	Secure Storage
A.SST.AAM.D	SST Application Assets Manager Data	Secure Storage
A.SST.CPDO.D	Cryptographically-Protected Data Objects: Data objects provided by the SST once they have been cryptographically protected	Secure Storage
A.SST.KM.K	SST Keys: Keys used to protect the security properties of Sensitive Objects or other keys, by transforming them into. Cryptographically-Protected Data Objects	Secure Storage
A.SST.KM.C	SST Key Manager: Code of the SST that manages keys	Secure Storage
A.SST.KM.D	SST Key Manager Data	Secure Storage
AC.SST.IPM.H	SST Integrity-Protected Memory	Secure Storage
AC.SST.ICPM.H	SST Integrity and Confidentiality-Protected Memory	Secure Storage
AC.SST.ICM.H	SST Integrity-Checked Memory	Secure Storage

ASSET (A) OR ASSET CONTAINER (AC)	DESCRIPTION	CHAPTER
A.FSB.ISB.C	Initial Secure Boot code that initiates the Secure Boot procedure	Flexible Secure Boot
A.FSB.ASB.C	Additional Secure Boot code that is called by the initial Secure Boot to execute part of the Secure Boot procedure	Flexible Secure Boot
A.FSB.BIND.D	Binding data used to verify the Authenticity of the Integrity-Protected Memory. The specific nature of this data is determined by the technical solution used. A.FSB.BIND.D may include cryptographic keys.	Flexible Secure Boot
A.FSB.VER.K	Verification keys that are used to verify the signature and Authenticity of the code base, both during the boot process, and when code base updates are received (if required)	Flexible Secure Boot
A.FSB.HASH.D	Hash digest values that verify the Integrity of the code base	Flexible Secure Boot
AC.FSB.IPM.H	Integrity-Protected Memory where code base is stored	Flexible Secure Boot
A.FSB.CB.D	Code base data – the ME program code and data that is verified by the Secure Boot procedure. It is treated by the Secure Boot as data	Flexible Secure Boot
A.FSB.UPD.C	Update code base Application – the Application that updates the code base with the code base image	Flexible Secure Boot
A.FSB.UPD.D	Update code base image – the code base image that is downloaded to replace some or all of the code base	Flexible Secure Boot
A.FSB.CBL.D	Code base list – data structure that defines what program code and data must be verified by the Secure Boot procedure	Flexible Secure Boot
A.GBA1.AUTN.D	Authentication value for user authentication to the network (UICC GBA-related data). Authentication value which authenticates HSS/BSF to the UICC (UICC GBA-related data)	Generic Bootstrapping Architecture
A.GBA1.RAND.D	Random value for user authentication to the network (UICC GBA-related data)	Generic Bootstrapping Architecture
A.GBA1.BTID.D	Identifier used to bind the subscriber identity to the keying material (UICC GBA-related data)	Generic Bootstrapping Architecture
A.GBA2.SIM.C	Code of SIM Application (GBA-related Applications on UICC)	Generic Bootstrapping Architecture
A.GBA2.USIM.C	Code of USIM Application (GBA-related Applications on UICC)	Generic Bootstrapping Architecture
A.GBA2.ISIM.C	Code of ISIM Application (GBA-related Applications on UICC)	Generic Bootstrapping Architecture

ASSET (A) OR ASSET CONTAINER (AC)	**DESCRIPTION**	**CHAPTER**
A.GBA3.CK.K	Keying material (Main UICC GBA-related keys)	Generic Bootstrapping Architecture
A.GBA3.IK.K	Keying material (Main UICC GBA-related keys)	Generic Bootstrapping Architecture
A.GBA3.Ks.K	Keying material (Main UICC GBA-related keys)	Generic Bootstrapping Architecture
A.GBA3.KsIntNAF.K	Resulting key (Main UICC GBA-related keys)	Generic Bootstrapping Architecture
A.GBA3.KsExtNAF.K	Resulting key (Main UICC GBA-related keys)	Generic Bootstrapping Architecture
A.GBA4.Kc.K	Keying material for 2G-GBA (Main ME GBA-related keys, 2G GBA & GBA_ME)	Generic Bootstrapping Architecture
A.GBA4.Ks.K	Keying material (Main ME GBA-related keys, 2G GBA & GBA_ME)	Generic Bootstrapping Architecture
A.GBA4.KsNAF.K	Resulting key (Main ME GBA-related keys, 2G GBA & GBA_ME)	Generic Bootstrapping Architecture
A.GBA5.NAFId.D	FQDN concatenated with protocol id (GBA Server Assets (data, code))	Generic Bootstrapping Architecture
A.GBA5.KsInput.D	Random value for key derivation (GBA Server Assets (data, code))	Generic Bootstrapping Architecture
A.GBA5.CertBSF.D	BSF certificate for TLS connection with UE in 2G-GBA (GBA Server Assets (data, code))	Generic Bootstrapping Architecture
A.GBA5.BTID.D	BTID is used to bind the subscriber identity to the keying material (GBA Server Assets (data, code))	Generic Bootstrapping Architecture
A.GBA5.Server.API.C	Code of the GBA Server interface (GBA Server Assets (data, code))	Generic Bootstrapping Architecture
A.GBA5.Server.Crit.C	Code of the GBA Server (GBA Server Assets (data, code))	Generic Bootstrapping Architecture
A.GBA5.IMSI/IMPI.D	Identifier used to define the subscribers (GBA Server Assets (data, code))	Generic Bootstrapping Architecture
A.GBA5.Keylifetime.D	Key lifetime (GBA Server Assets (data, code))	Generic Bootstrapping Architecture
A.RIC.RIC.H1	Primary Integrity checking function	Run-time Integrity Checking

ASSET (A) OR ASSET CONTAINER (AC)	DESCRIPTION	CHAPTER
A.RIC.RIC.D1	Primary Integrity checking function verification data (May include reference Hashes, initialisation values, parameters, etc.)	Run-time Integrity Checking
A.RIC.RIC.H2	Secondary Integrity checking function	Run-time Integrity Checking
A.RIC.RIC.D2	Secondary Integrity checking function verification data (May include reference Hashes, initialisation values, parameters, etc.)	Run-time Integrity Checking
A.RIC.KEYS.K	RIC key material	Run-time Integrity Checking
A.RIC.INI.C1	Primary initialisation function	Run-time Integrity Checking
A.RIC.INI.C2	Secondary initialisation function	Run-time Integrity Checking
A.RIC.ERR.C1	Primary Error Handler	Run-time Integrity Checking
A.RIC.ERR.C2	Secondary Error Handler	Run-time Integrity Checking
A.RIC.APP.D1	Primary Application Code/Data segment to be checked	Run-time Integrity Checking
A.RIC.APP.D2	Secondary Application Code/Data segment to be checked	Run-time Integrity Checking
A.RIC.STS.H1	Primary Monitored Hardware status	Run-time Integrity Checking
A.RIC.STS.H2	Secondary Monitored Hardware status	Run-time Integrity Checking
A.SUIO.DO.D	Sensitive object provided to the UI by the Application or by the UI to the Application calling it. It can represent any information, but is treated as opaque data by the SUIO. Requirements on this Asset will apply when this Asset is under the control of the SUIO	Secure Access to User Input/Output Facility
A.SUIO. C	Code of the Secure UI software components	Secure Access to User Input/Output Facility
A.SUIO.D	Data required by A.SUIO.C (e.g. access rights)	Secure Access to User Input/Output Facility
A.SUIO.IO.C	Code of the Secure UI software components for managing secure input/output devices (SUIO secure input/output drivers)	Secure Access to User Input/Output Facility
A.SUIO.IO.D	Data of the Secure UI SW components for managing secure input/output devices (SUIO secure input/output drivers)	Secure Access to User Input/Output Facility
A.SUM.UAUTH.K	SC UICC Key: Keys used to authenticate UICC to ME	Secure Interaction of UICC with Mobile

ASSET (A) OR ASSET CONTAINER (AC)	DESCRIPTION	CHAPTER
A.SUM.TAUTH.K	SC ME Keys: Keys used to authenticate ME to UICC	Secure Interaction of UICC with Mobile
A.SUM.SC.K	SC Session Keys: Session keys used to protect channel communications	Secure Interaction of UICC with Mobile
A.SUM.SC.D	SC Data: Exchanged data in the SC	Secure Interaction of UICC with Mobile
A.SUM.SC.MNG.C	Critical SUM Secure Channel Manager: ME software component managing a SC (including key management)	Secure Interaction of UICC with Mobile
A.SUM.SC.MNG.D	Critical SUM Secure Channel Manager Data: Data used by A.SUM.SC. MNG.C to authenticate UICC and define policy	Secure Interaction of UICC with Mobile
A.SUM.SC.DIS.{C,D}	General SUM Secure Channel Manager: ME software component managing discovery process and communications protocols	Secure Interaction of UICC with Mobile
A.BC.SCK.K	Smart Card Key (Assets held entirely in the UICC)	Broadcast Service Protection
A.BC.SMK.K	Subscriber Management Key (Assets held entirely in the UICC)	Broadcast Service Protection
A.BC.LTKM.D	Long Term Keystream Messages, decrypted (Assets held entirely in the UICC)	Broadcast Service Protection
A.BC.SEK.K	Service Encryption Key (Assets held entirely in the UICC)	Broadcast Service Protection
A.BC.PEK.K	Program Encryption Key (Assets held entirely in the UICC)	Broadcast Service Protection
A.BC.SRK.K	Subscriber Request Key (Assets in the ME)	Broadcast Service Protection
A.BC.BCA.BTID.D	Bootstrapping Transaction Identifier (Assets in the ME)	Broadcast Service Protection
A.BC.STKM.D	Short Term Keystream, Messages, decrypted (Assets in the ME)	Broadcast Service Protection
A.BC.TEK.K	Traffic Encryption Keys (Assets in the ME)	Broadcast Service Protection
A.BC.TBK.K	Terminal Binding Key (Assets in the ME)	Broadcast Service Protection
A.BC.BCA.C	Broadcast Service Protection Application code, running on the ME (Assets in the ME)	Broadcast Service Protection
A.BC.BCA.D	Data used by the Broadcast Service Protection Application (Assets in the ME)	Broadcast Service Protection
A.BC.CON.D	Broadcast content, decrypted (Assets in the ME)	Broadcast Service Protection

ASSET (A) OR ASSET CONTAINER (AC)	DESCRIPTION	CHAPTER
A.BC.FLA.D	Access criteria flags contained in STKMs (Assets in the ME)	Broadcast Service Protection
A.BC.ESG.D	Electronic Service Guide (Assets in the ME)	Broadcast Service Protection
A.BC.MEK.K	Device or Broadcast Application keys used to authenticate ME to BSP (NAF) for TBK retrieval or Secure Channel key agreement (Assets in the ME)	Broadcast Service Protection
A.TDM.SA.K	Server Authentication Keys – the credentials used to authenticate the management server during the download process at the transport layer or the SyncML command (Assets Shared by the two use cases UC1 and UC2)	Trusted Device Management
A.TDM.DL.C	Download Agent (Assets Shared by the two use cases UC1 and UC2)	Trusted Device Management
A.TDM.DISP.D	Data I/O displayed on Screen – User prompts (Assets Shared by the two use cases UC1 and UC2)	Trusted Device Management
A.TDM.KB.D	Data I/O on Keyboard – User Approval (Assets Shared by the two use cases UC1 and UC2)	Trusted Device Management
A.TDM.FMA.C	Firmware Management Agent – Code in charge of upgrading the firmware of the ME (Assets Specific to UC1 (Firmware Management))	Trusted Device Management
A.TDM.FPA.K	Firmware Package Authentication Keys – cryptographic keys used to ensure the Authenticity of the downloaded package A.TDM.DPF.D (Assets Specific to UC1 (Firmware Management))	Trusted Device Management
A.TDM.FPC.K	Firmware Package Confidentiality Protecting Keys – cryptographic keys used to ensure the Confidentiality of the downloaded package A.TDM.DPF.D (Assets Specific to UC1 (Firmware Management))	Trusted Device Management
A.TDM.ACLF.D	Access Control List of the FUMO Management Tree – Data Structure that defines which management commands each management server is allowed to perform on the nodes of the OMA DM FUMO sub-tree (Assets Specific to UC1 (Firmware Management))	Trusted Device Management
A.TDM.DPF.D	Download Package containing the Firmware Update Code (Assets Specific to UC1 (Firmware Management))	Trusted Device Management
A.TDM.SMA.C	Software Management Agent – Code in charge of installing, upgrading and uninstalling software of the ME (Assets Specific to UC2 (Software Management))	Trusted Device Management
A.TDM.SPA.K	Software Package Authentication Keys – cryptographic keys used to ensure the Authenticity of the downloaded package	Trusted Device Management

ASSET (A) OR ASSET CONTAINER (AC)	**DESCRIPTION**	**CHAPTER**
	A.TDM.DPS.D (Assets Specific to UC2 (Software Management))	
A.TDM.SPC.K	Software Package Confidentiality Protecting Keys – cryptographic keys used to ensure the Confidentiality of the downloaded package A.TDM.DPS.D (Assets Specific to UC2 (Software Management))	Trusted Device Management
A.TDM.ACLS.D	Access Control List of the SCOMO Management Tree – Data Structure that defines which management commands each management server is allowed to perform on the nodes of the OMA DM SCOMO sub-tree (Assets Specific to UC2 (Software Management))	Trusted Device Management
A.TDM.DPS.D	Download Package containing the Software Installation Code or Software Update Code (Assets Specific to UC2 (Software Management))	Trusted Device Management
A.TDM.ROOT.K	Root Public Keys used by the TDM (Assets Specific to UC2 (Software Management))	Trusted Device Management
A.MCO.ROOT.K	Root Public Keys used by the Mobile Commerce Application	Mobile Commerce
A.MCO.CERTIF.D	Digital certificates related to the payment system that allow the mobile to authenticate the UICC	Mobile Commerce
A.MCO.APP.C	Mobile Commerce Application Code in the ME (manages input/output between the user and the UICC)	Mobile Commerce
A.MCO.APP.D	Mobile Commerce Application Data in the ME (including data used to verify UICC identity)	Mobile Commerce
A.MCO.APP.K	Key used to prove mobile commerce Application identity	Mobile Commerce
A.MCO.SESSION.K	Mobile Commerce Application session key used to secure communication between the Application and UICC	Mobile Commerce
A.MCO.ACTPAY.D	Data/Key required to activate the payment (e.g. pin code)	Mobile Commerce
A. MCO.IN.D	Data I/O exchanged on Keyboard between the Mobile commerce Application and the UICC	Mobile Commerce
A. MCO.OUT.D	Data output from the UICC to be displayed on screen	Mobile Commerce

-------- END OF DOCUMENT --------