D01.7 – Implementation Report of the Logical TrustZone / TPM integration

<table>
<thead>
<tr>
<th>Project number:</th>
<th>216888</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project acronym:</td>
<td>TECOM</td>
</tr>
<tr>
<td>Project title:</td>
<td>Trusted Embedded Computing</td>
</tr>
<tr>
<td>Start date of the project:</td>
<td>January 1, 2008</td>
</tr>
<tr>
<td>Duration:</td>
<td>36 months</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Deliverable type:</th>
<th>Report - Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deliverable reference number:</td>
<td>216888 / D01.7 / 1.1</td>
</tr>
<tr>
<td>Deliverable title:</td>
<td>Implementation Report of the logical TrustZone / TPM integration</td>
</tr>
<tr>
<td>WP contributing to the deliverable:</td>
<td>WP1</td>
</tr>
<tr>
<td>Due date:</td>
<td>M24</td>
</tr>
<tr>
<td>Actual submission date:</td>
<td>January, 7th, 2010</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Responsible organisation:</th>
<th>Trusted Logic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authors:</td>
<td>TL (Nicolas Ponsini)</td>
</tr>
<tr>
<td>Abstract:</td>
<td>This document details the integration between TrustZone and a TPM.</td>
</tr>
<tr>
<td>Keywords:</td>
<td>TrustZone, Integration</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dissemination level:</th>
<th>PU: Public</th>
</tr>
</thead>
<tbody>
<tr>
<td>Revision:</td>
<td>1.0</td>
</tr>
<tr>
<td>Instrument:</td>
<td>IP</td>
</tr>
<tr>
<td>Thematic Priority:</td>
<td>ICT</td>
</tr>
</tbody>
</table>
# Table of Contents

1  Introduction ..................................................................................................................3
  1.1  Purpose of the Document .........................................................................................3
  1.2  Scope of Application .................................................................................................4

2  TrustZone .......................................................................................................................5
  2.1  TrustZone Technology Overview ..............................................................................5
  2.2  TrustZone Cores ........................................................................................................5
  2.3  TrustZone Software ...................................................................................................6

3  Trusted Computing .........................................................................................................8
  3.1  Functionality .............................................................................................................8
  3.2  Secure Key, data hierarchy and certificate chain ......................................................9
  3.3  Measuring data integrity by TPM .............................................................................10
  3.4  TCG Software Stack .................................................................................................10

4  Implementation Report ...................................................................................................11
  4.1  Secure boot .............................................................................................................11
  4.2  TrustZone as a passive component ........................................................................11
  4.3  Shared memory .......................................................................................................12
  4.4  Secure management of hardware accelerators .......................................................13
  4.5  Secure Storage .......................................................................................................14
  4.6  Monotonic Counter .................................................................................................14
  4.7  Client authentication ...............................................................................................14

5  Conclusion .....................................................................................................................16

6  List of Abbreviations .....................................................................................................17

7  References ....................................................................................................................18
1 Introduction

1.1 Purpose of the Document

In the TECOM project, the main objective of the WP1 is to address two main requirements coming up for Trusted Computing in embedded computing platforms:

- Integration of a trust component into the embedded host processor for cost and device complexity reasons
- Combination of the trust component together with a host processor in one chip or in one covered housing for easy hiding and shielding of typical attack targets like the processor bus, interrupt lines or boot memory manipulations.

This goal mainly relies on the Tasks 1.1 and 1.2 of the Work Package which target to design a VHDL-TPM hardmacro (VHDL: Very High Level Design Language; TPM: Trusted Platform Module) which can be integrated on demand into typical small size embedded computing chips and actually to a corresponding reference chip. In addition Task 1.3 aims to provide the firmware, software and drivers that address this chip and that are optimized for the typical requirements of embedded platforms. At last, the goal of the Task 1.4 is to study the possible architectures to integrate ARM processor with TrustZone technology enabled and a TPM and then to provide a report of an integration of those two components.

This document is the implementation report of such integration. The point is that neither a TPM nor a platform with TrustZone enabled which can address a TPM has been made available for the TECOM project. Thus, this document deals with the integration of a secure execution environment within TrustZone and based on a platform with secure hardware components only accessible by TrustZone. Such platforms already exist on the market: we can mention the Texas Instruments M-Shield OMAP3 platform as an example. In such architecture, a TPM can be seen as a software component executed as a secure service of the secure execution environment. The figure below shows the retained architecture for this report.

![Implemented architecture](image.png)

**Figure 1: Implemented architecture**

Secure Hardware components refer to hardware accelerators, secure RAM, protected keys, secure boot, secure controllers...

Chapter 2 and 3 extracted from the TECOM Architecture document provide a summary of the two technologies: TrustZone and Trusted Computing through the use of a TPM. The Chapter 4 reports the implementation done on a TrustZone enabled platform.
1.2 Scope of Application

This document corresponds to a deliverable of the Task 1.4 of the Work Package 01 as defined in the Description of work of the TECOM project.

![Diagram of TECOM WP01 Tasks]

Figure 2: TECOM WP01 Tasks
2 TrustZone

2.1 TrustZone Technology Overview

ARM TrustZone® technology is a key enabling technology, targeted specifically at securing consumer products such as mobile phones, PDAs, set top boxes or other systems running open operating systems (OS), such as Symbian OS, Linux and Windows Mobile.

ARM TrustZone® technology is implemented within the microprocessor core itself, enabling the protection of on and off-chip memory and peripherals from software attack. Since the security elements of the system are designed into the core hardware, security issues surrounding proprietary, non-portable solutions outside the core are negated. In this way, security is maintained as an intrinsic feature at the heart of every device, with minimal impact to the core area or performance, enabling licensees to build any desired additional security features, such as cryptography, onto a secure hardware foundation.

A new mode, ‘Secure Monitor’, within the core acts as a gatekeeper and reliably switches the system between secure and non-secure states. When the monitor switches the system to the secure state the processor gains additional levels of privilege to run trusted code. It can then handle tasks such as authentication, signature manipulation and the processing of secure transactions.

ARM TrustZone technology tags and partitions secure code and data within the system, and maintains a clear, hardware separation between secure and non-secure information. This separation enables secure code and data to run alongside an OS securely and efficiently, without being compromised or vulnerable to attack. The level of security awareness within a TrustZone technology-enabled system can vary from the simplest security measures implemented just within the core and level-1 memory system, to more extensive security implementations which reach other memory and peripherals, as well as the core.

2.2 TrustZone Cores

The security extensions provided by TrustZone technology allow a secure processor core to be implemented in an area-efficient way without affecting performance. The first implementation of TrustZone technology is in the ARM1176JZ(F)-S, cortex A8 an cortex A9 products.

The presence of the new Secure State, Monitor Mode and SMI instruction are the main additions from the programmers view. In terms of hardware additions there are a number of important features which enable TrustZone secure systems to be implemented with these processors. In particular, from a system perspective, the AMBA 3 AXI protocol allows memory mapped peripherals to be marked as either Secure or non Secure.

'NS' bit - The most obvious addition is the Non-Secure indicator bit, 'NS'. This bit determines whether the processor is in secure or non-secure state, and cannot be written in non-secure state. The processor is in secure state at reset.

Additional CP15 registers - Banked and swapped out when the processor is in secure state. These control access primarily to secure debug and trace features.

Memory Management Unit (MMU) & Caches - To enhance system performance the ARM TrustZone processors have an integral cache memory and MMU. In secure state, access security is determined by an additional security-bit in the processors’ MMU page tables. This bit is crucial to ensure cache coherency is maintained between secure and non-secure states.

There are two sets of page tables in the MMU, one secure and one non-secure. When the processor is in non-secure state the page table entry security-bits are ignored so that all memory accesses generated by the MMU are non-secure.
The MMU is relied upon to propagate the secure/non-secure state to higher levels of memory (e.g. Level 2 cache and system memory). There is no other signalling from the processor to indicate the security state.

### 2.3 TrustZone Software

To complement the security additions to processor and system hardware, ARM has selected Trusted Logic’s Trusted Foundations Software to deliver a highly-secure execution environment for TrustZone-enabled processor cores. This ensures that ARM licensees can count on a complete framework for security including pre-integrated and tested software designed for high security environments.

Trusted Foundations Software:
- Incorporates unique security countermeasures and techniques;
- Allows top-level security at minimal development cost, implementation risk and time-to-market;
- Facilitates differentiation by providing various means of customization.

The architecture of the TFSW is illustrated by the figure below:

**Figure 3: Trusted Foundations Software architecture**

The Trusted Foundations Software technology has been designed to manage exactly two worlds (no more, no less): the normal world stands for the usual execution environment which hosts an operating system or a hypervisor and applications above. The secure world which hosts the TFSW executed in the TrustZone secure mode.

**Security-demanding application**

The security-demanding application (or simply application) is the part of the application that does not manipulate sensitive data and that is executed on top of a rich OS (e.g. Symbian OS, Windows Mobile, Android, or Linux).
Secure services
The parts of an application that manipulate and process sensitive information are called secure services. A typical example is a DRM agent that enforces digital content license rights and decrypts encrypted media. Only software that has been digitally signed and verified can be installed in the Secure Environment. This prevents rogue code from residing in the Secure Environment. The secure services can either be developed in 'C' language or interpreted via the GPD/STIP technology.

The **Secure Service Development Interface (SSDI)** that is made available for secure service developers permits to manage the major mobiles protocols and standards:

- OMA DRM v2.0
- X.509
- SSL V3.0 / TLS
- IPSec
- Payment, ...

Default secure services are provided with the Trusted Foundations Software.

Cryptographic service
The cryptographic API exposed by the cryptographic service is a subset of the PKCS#11 standard API. Through the Cryptographic API, each client application is bound to a private key store: a token where to manage persistent keys. Main functionalities are key management and Digest, symmetric and asymmetric cryptographic operations.

Secure storage service
This service permits to store credentials and sensitive data securely and persistently in a transparent way. This service exposes an ANSI/POSIX file base API to applications. The secure storage API is based on a flat file structure where files are private to an application. It guarantees confidentiality, integrity, authenticity of the data and atomicity of the operations with anti replay protection.

Monotonic Counter service
This service permits each application to read and increment a monotonic counter.

Secure channel
The security-demanding application communicates with secure services via a secure channel which is private between an application and the service it is using. In other words, no other application can access the data that an application is exchanging with a secure service. From the application environment, the interface for this channel is the Trusted Foundations API.

There are different ways an application can establish a connection with a secure service. Three different login methods exist:

- No login i.e. public access (secure service may refuse access to certain functions),
- Client identification: it can be made by an operating system or an hypervisor at the kernel level in order to uniquely identify an application.
- Client authentication: it relies on a Public Key Infrastructure and a Signature File associated to the client application. The Root Key of the PKI is embedded within the TFSW. The Signature

**Trusted Foundations Software framework**
The Trusted Foundations Software framework is a security-centric execution environment. Its kernel enables the execution of the secure services and is fully independent of the OS residing in the application environment. The application environment and the secure environment need to be separated by isolation mechanisms - preferably hardware-based ones - that prevent software in the application environment from interfering with software in the secure environment. The Trusted Foundations Software framework interacts with the secure peripherals via secure drivers.

Secure drivers
The **Secure Driver Development Interface (SDDI)** that is made available to secure driver developers permits to abstract any hardware peripheral of the platform. In particular if this hardware is part of the TrustZone address space, it can be only accessed through the TFSW which guaranties a strong access control.
3 Trusted Computing

Trusted Computing technology from the Trusted Computing Group standardization defines a Trusted Platform Module (TPM) which contains functionality for cryptographically measurement of the integrity of data structures (boot sequences, user or system processing data but also code) and make them available for attestation (means digitally signing them and provide means for comparison with already known reference data). Additionally this concept already has digital signing and verification functions, public key infrastructure functionality and integrated conditional access methods (sealing operational data for example with external passwords but also with the result of the measurement and attestation process).

In accordance with the TCG architecture, the TPM provides the security functions requiring particular protection and which are therefore also implemented in a secure hardware environment. The TPM is designed as a passive part. It has no means of actively influencing program execution of the central processor or the boot operation. It receives only control and status measuring data from the central processor which it processes, stores and reads out again from its secure structure, and feeds these results back to the central processor.

3.1 Functionality

![Figure 4: TPM component architecture](image)

The various key classes are stored in a protected manner in the TPM. The access method is selected according to key type (TPM-bound, migratable, signature, identity (AIK), binding keys).

A TPM provides:

- **System authentication**
  
  Authentication and validation of the platform to third parties.

- **Communication of the system’s security status (attestation)**
  
  Trusted communication of the security-relevant (platform-user-defined) configuration.

- **Random number generator**
  
  Generation of hardware-based random numbers for secure key generation.
### File sealing

Binding of data to the system configuration and signing of the data when storing with the hash value of the configuration. Access to the data is then only possible if the configuration remains unchanged: PCR values match the current state.

### Secure saving of configuration changes

Status changes are detected, safeguarded by the SHA-1 hash algorithm.

### 3.2 Secure Key, data hierarchy and certificate chain

For an unequivocal identity as well as for secure storing of key and data material a TPM contains a complete PKI structure for assuring the integrity of stored data. This digital signature capability is used by internal functions like attestation of boot integrity value but can also be used for critical data and keys of the host environment and use scenario.

**Endorsement Key**

During TPM production every TPM will be loaded with a unique public key pair, the so called endorsement key. This key pair and derivatives from it can be used for identifying the TPM as well as data structures, generated for authentication and attestation. Using this feature it is easily possible to identify also the whole system itself for attestation to other trusted systems.

**The Storage Root Key (SRK)** forms the root of a key hierarchy in which other lower-order keys, but also data (blobs), are securely stored, their trustworthiness therefore depending on the SRK. The SRK is automatically generated by the owner in a “Take Ownership” operation. If the owner of a TPM gives up this ownership, this also deletes the SRK and also makes all the keys protected by it completely unusable, which is welcome for data protection purposes.

![TPM Key and certificate hierarchy](image-url)

**Figure 5: TPM Key and certificate hierarchy**
3.3 Measuring data integrity by TPM

The most useful and widely used function is the measuring of data integrity by a built-in standard function. Usually a TPM is connected in parallel to the data bus of the host processor and is capable to copy all the code or data on the bus to an internal hashing processor which computes out of these data a concatenated hash value, named the measuring value of the data stream.

This capability has a special importance for the boot process, as this is the initial value of the whole system boot process and a precondition for a correct and not manipulated start of the system. As the boot process is usually a complex sequence of data, the measuring is done in different parts and segments on data blocks. Every block is firstly measured, the measured value is then securely stored by the TPM’s internal digital signing system and after that the data block is executed. This is done block by block and at the end a sequence of digitally signed measurement values is available inside the TPM which can be compared against external or internal reference values (attestation). As this is the initial procedure which is executed to gain information about the status of trust for the whole system it is called Core Root of Trust Measurement (CRTM).

![Figure 6: Data Integrity measurement](image)

![Figure 7: Measuring the integrity of the boot sequence block by block](image)

The main difference of TPM supported booting and generation of CRTM against ARM TrustZone and similar processors is that the TPM booting is defined in the standard and the basic functionality is part of the TPM specification while TrustZone boot implementations have to be realized by the system designer itself. Additionally TPM functionality is realized in a separate chip which therefore cannot be manipulated even by attacking the system program.

3.4 TCG Software Stack

In the architecture promoted by TCG, there is a TSS (TCG Software Stack) which is the supporting software above the TPM. This TSS is split in several layers:

- The **TPM Device Driver (TDD)** which directly communicate with the embedded TPM hardware. It translates low-level API calls into hardware commands and reciprocally. This layer is OS and vendor dependant.
- The **TPM Device Driver Library (TDDL)** implements an OS and vendor independent library to access and communicate with the embedded TPM. All TDDLs implement a given and fixed TDDL interface (TDDL). The TDDL is limited to only a handful set of primitives: `Tddli_Open`, `Tddli_Close`, `Tddli_Cancel`, `Tddli_Get/SetCapability`, `Tddli_GetStatus` and `Tddli_TransmitData`.
- The **TSS Core Services (TCS)** is more oriented towards trusted computing functionalities a TPM can provide. It offers an API capable of reading TPM’s PCRs - `Tcsip_PcrRead`, quoting - `Tcsip_Quote`, sealing - `Tcsip_Seed`, etc.
- The **TSS Service Provider (TSP)** is the highest TSS layer. It provides high trusted computing functionalities as cryptography and storage capabilities.
4 Implementation Report

This report refers to the integration of a secure execution environment within TrustZone and based on a platform with secure hardware components only accessible by TrustZone. The secure execution environment is the Trusted Foundations Software and the platform is a Texas Instruments M-Shield OMAP3 platform. The operating system used for the normal world was Linux.

In addition, some references are made to the TPM as a secure service running above the Trusted Foundations Software and the benefits of such integration will be highlighted. The figure below summarizes the architecture of the integration:

![Figure 8: Secure TPM service architecture](image)

The normal world stack is composed of a TDDLI stub which is an implementation of the TDDLI as defined by the TCG above the TFSW API and TFSW Driver. Those two components permits to format commands and switch to the secure world to address the TPM secure service which execute the command effectively. Then the answer is handled again by the TFSW driver and TFSW API and forwarded to the TDDLI stub.

4.1 Secure boot

Using TrustZone, it is possible to boot in secure mode. This is an important feature especially to achieve secure boot measurements: code isolated from the normal world and usually in ROM is in charge of verifying the code to be loaded in TrustZone and in the public area. Once the TPM secure service is loaded, it retrieves the measurement values verified at boot time so that the root of trust for measurement is up and ready.

4.2 TrustZone as a passive component

In this integration, the principle of the Trusted Foundations software as a passive execution environment has been adopted. The principle is that the normal world execution environment hosting the rich operating system is the default execution environment. The switch to TrustZone is triggered either by the Normal World requiring a service to the secure world either by the interrupt controller while an interruption is dedicated to the secure world. In that, the integration of the TPM as a secure service entirely respects the TPM principle as a passive component within the platform environment.
Note that in this configuration, the secure world is scheduled as any other process of the operating system and interrupted by the OS when required.

### 4.3 Shared memory

TrustZone enabled processor offers two set of MMU tables one reserved for the trustZone and one for the public environment. The RAM is split in secure RAM and public RAM. The secure world can map and access public and secure RAM whereas the normal world is limited to the public RAM as shown in the figure below:

![Platform](image)

**Figure 9: RAM mapping**

During this integration, the notion of shared memory in between the two worlds has been implemented: buffers of memory in the client memory space can be shared with a secure service of the TFSW without having to copy it. Within the TFSW driver, the virtual address of the memory segment permits to retrieve the physical address and bits configuration such as the type of memory and the cache configuration. This information once passed to the TFSW permits to map the memory segment within the secure world setting the NS bit to 1.

This feature is useful to exchange large data buffers. Those shared memory blocks are private between the security-demanding application and the secure service (linked to a session) and access flags permit to specify a Read or Read/Write access on client and service side. Shared memory is either allocated via the TFSW API or managed by clients and registered to the TFSW API.

For instance, Shared memory mechanisms are especially interesting for the Crypto Service of the TECOM Embedded Security Layer as specified in the TECOM architecture document. This service is subject to manage a large amount of data in input/output to perform symmetric cryptographic operations or digest operations. In those cases, the shared memory will be used. A typical use case is the DRM use case.

Besides, registered shared memory also permits to optimize performance as the buffer of the client application space is mapped to the corresponding Secure Service without having to copy it. In the context of a TPM service executed within TrustZone, shared memory blocks are interesting in term of performance for large buffers to be hashed using SHA-1.
4.4 Secure management of hardware accelerators

During this implementation, the cryptographic operations have been achieved using hardware cryptographic accelerators for asymmetric (modular exponentiation), symmetric and digest operations. Programming those accelerators only accessible by the secure world has permit to obtain better performances than software operations. Besides, coupled with the programming of the secure DMA controller we managed to bypass the CPU load overhead for symmetric and digest algorithms enabling high bandwidth for streaming operations.

However, the TFSW is a passive component scheduled as any other process of the operating system and that can be interrupted by public interruptions. This can be penalizing with real time operating systems interrupted each tick of the clock. In such configuration, while performing cryptographic operations on large buffers, the time passed to process the buffer is hugely overloaded by the time passed to switch to the normal world and to go back to the secure world after the interrupts. Because of this, specific emphasis on optimisation techniques to improve cryptographic performances has been implemented. Those techniques have been based on protection mechanisms also known as firewalls which permit to relax the fact that hardware accelerators registers are only accessible by the secure world. In our case, we used this feature to configure the symmetric hardware accelerators so that the key and configuration registers are still only accessible by the secure world but the input/output registers and the DMA controller registers are also accessible by the normal world. The secure world is the only that can configure and revoke this mode. The figure below shows this configuration:

![Shared accelerators architecture](image)

**Figure 10: Shared accelerators architecture**

By sharing non sensitive registers of crypto hardware accelerators, we managed to offer a mode of operations which permit to avoid back and forth exchanges and context switches in between the two worlds and thus to reach a very good level of performance. This mode is especially dedicated for streaming or Virtual Private Network use cases. Note that this mode has also been adapted to the digest accelerators (specific case with no cryptographic key). This is interesting for the TPM service while manipulating large amount of data when performing SHA-1 digest operations.
4.5 Secure Storage

When code is executed within TrustZone there is no possible access to the flash memory and it is not possible to store non volatile data. However, a secure storage service (based on on-chip device master key) is made available to secure services by our implementation. To achieve this, we have implemented a delegation service and a delegation daemon as shown in the figure below:

![Secure Storage Delegation Architecture](image)

**Figure 11: Secure Storage delegation architecture**

In this architecture, the secure storage daemon is executed in the user space and has access to the file systems primitives made available by the operating system. The secure storage daemon is always connected to a delegation service waiting for a command from it to execute. This configuration is little bit different of the usual case because the daemon at the application layer is waiting for commands triggered by the delegation service whereas in the usual configuration a secure service is waiting for commands triggered by applications.

All sensitive data and operations are managed and performed within the delegation service within TrustZone. The service drives the operations performed by the daemon within the normal world which are none sensitive and only permit to access the file system of the operating system.

The secure storage service serves for the TPM secure service to emulate non volatile memory.

4.6 Monotonic Counter

In our implementation, the TFSW provides a monotonic counter service based on non volatile registers only accessible within the secure world. This monotonic counter service is directly accessed by the TPM secure service to implement monotonic counters managed by a TPM.

4.7 Client authentication

Connection to a secure service may require client identification and/or client authentication. Three different login methods have been implemented:

- No login i.e. public access. A secure service may refuse access to certain functions depending on the access control policy.
- Client identification using the full path name of the demanding application. In this login mode, the implementation transmits the client identifier to the secure service.
Client authentication. It relies on a Public Key Infrastructure and a Signature File associated to the client application. The Root Key of the PKI is embedded within the TFSW. The Signature File contains the following elements:

- A manifest file associated with the application. It must contain the application Universal Unique Identifier (UUID), which must be generated by the application signer. The application UUID uniquely identifies the application. This means that the TFSW will not distinguish two applications with the same UUID. The manifest must also contain the digest of the application binary executable. Since the whole manifest is signed, the application binary is therefore indirectly authenticated.
- A signature of the manifest file. The signing key is an RSA private key.
- Optionally a chain of certificates that authenticates the signing key.

The signature file is generated using a signature process similar to the one shown below:
5 Conclusion

This integration permits to understand how a platform based on secure hardware components and with TrustZone processor enabled can serve to build a trusted execution environment with features such as secure storage, shared memory, monotonic counter or secure cryptographic operations. As already explained within the introduction, this integration has been adopted as the material for integration with a full TPM hardware was not available. However, the integration realized shows that a trusted execution environment can be the host for a TPM service and a good alternative to a full TPM hardware only mapped and accessible by the secure world: the same level of security can be reached depending on the secure peripherals available on the platform. Actually, the case of a full TPM hardware is simple to implement as in this case the TPM secure service is just a pass through to the hardware protected by TrustZone but the cost to add a complete hardware module in an embedded device is important.
6 List of Abbreviations

API  Application Programming Interface
MMU  Memory Management Unit
OS   Operating System
RAM  Random Access Memory
ROM  Read Only Memory
TC   Trusted Computing
TCG  Trusted Computing Group
TDD  TPM Device Driver
TDDL TPM Device Driver Library
TDDL1 TPM Device Driver Library Interface
TFAPI Trusted Foundations API
TFSW Trusted Foundations Software
TPM  Trusted Platform Module
TSS  TCG Software Stack
UUID Universal Unique Identifier
VHDL Very High Level Design Language
WP   Work Package
7 References

<table>
<thead>
<tr>
<th>TCG</th>
<th>Trusted Computing Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>TECOM</td>
<td>D05.2 TECOM Architecture –</td>
</tr>
<tr>
<td>Architecture</td>
<td><a href="https://svn.tecom-project.eu/Deliverables/D05.2-Architecture/TECOM-D05.2-Architecture.doc">https://svn.tecom-project.eu/Deliverables/D05.2-Architecture/TECOM-D05.2-Architecture.doc</a></td>
</tr>
<tr>
<td>TECOM TrustZone / TPM architectures</td>
<td>D01.6 TrustZone enabled processor with a TPM - <a href="https://svn.tecom-project.eu/Deliverables/D01.6-Architecture-TZ-TPM/TECOM-D01.6-TrustZoneEnableProcessorWithTPM.pdf">https://svn.tecom-project.eu/Deliverables/D01.6-Architecture-TZ-TPM/TECOM-D01.6-TrustZoneEnableProcessorWithTPM.pdf</a></td>
</tr>
<tr>
<td>TPM</td>
<td>TPM specifications</td>
</tr>
<tr>
<td></td>
<td><a href="http://www.trustedcomputinggroup.org/resources/tpm_main_specification">http://www.trustedcomputinggroup.org/resources/tpm_main_specification</a></td>
</tr>
<tr>
<td>TrustZone</td>
<td>TrustZone Technology overview</td>
</tr>
<tr>
<td>TSS</td>
<td>TCG Software Stack</td>
</tr>
<tr>
<td></td>
<td><a href="http://www.trustedcomputinggroup.org/developers/software_stack/specifications">http://www.trustedcomputinggroup.org/developers/software_stack/specifications</a></td>
</tr>
<tr>
<td>UUID</td>
<td>Universal Unique Identifier - RFC 4122</td>
</tr>
</tbody>
</table>