TCG TPM 2.0 Automotive Thin Profile

Family “2.0”

Level 00 Version 1.0

March 16, 2015

Contact: admin@trustedcomputinggroup.org

TCG Published

Disclaimers, Notices, and License Terms

Acknowledgements

TCG acknowledges the following contributors to this specification:

Dean Liberty (AMD), Tom Moulton (Atmel), Stacy Cannady (Cisco), Bill Jacobs (Cisco), Amy Nelson (Dell), Andreas Fuchs (Fraunhofer), Yoshi Hiyama (Fujitsu), Seigo Kotani (Fujitsu), Kouichi Yasaki (Fujitsu), Ira McDonald (High North), Carey Huscroft (HP), Guerney Hunt (IBM), Florian Schreiner (Infineon), Alan Tatourian (McAfee), Rob Spiger (Microsoft), David Wooten (Microsoft), Hisashi Oguma (Toyota InfoTechnology Center)

Table of Contents

Tables

Figures

1 Introduction (Informative)

Automotive vehicle solutions have increasingly leveraged information technology solutions to provide many benefits from entertainment to safety. The typical design consists of numerous interconnected subsystems communicating to external systems through gateway components. Vehicle systems routinely include multiple Electronic Control Unit (ECU). Each ECU consists of components similar to a traditional Personal Computer (PC) Client computer system and mobile phones with a Central Processing Unit (CPU), memory and applications. Each ECU may have RAM and/or ROM based software serviced and/or dynamically changing over the lifetime of the vehicle.

This specification describes how a Trusted Platform Module (TPM) can provide security benefits to the information technology systems in a vehicle. Typical benefits a TPM can provide include integrity reporting of software and cryptographic key creation, storage, management and use. In the automotive vehicle context, this specification describes scenarios of using TPMs for proving an ECU identity, reporting the software in use, and remote deployment of maintenance updates.

The TCG TPM 2.0 Library Specification consists of a library of commands and functionality. Not all TPM capabilities are applicable for all platforms. In the context of automotive ECUs, this specification defines a TPM 2.0 profile called the “Automotive Thin” profile, intended to meet the requirements of ECUs that perform a limited number of scenarios requiring subset of TPM 2.0 capabilities. Standardizing a reduced set of TPM 2.0 capabilities allows implementation and use of the profile without additional cost or complexity for unnecessary TPM capabilities.

2 Overview of Automobile Vehicle Systems (Informative)

There are significant differences between the capabilities of automotive vehicle systems and those found in typical servers, PCs, and mobile phones.

A modern automotive vehicle typically has over 100 separate processors (each with its own OS, RAM, and applications) that are called Electronic Control Unit (ECU) and are configured on three or more separate and isolated networks as shown in Figure 1. Even though the automotive vehicle appears to be single object that is Internet-connected, the vehicle is actually a complex system of separate networks that includes a Head-Unit or Gateway communicating with a Remote Center (a vehicle safety and maintenance center, typically operated by a manufacturer or government agency). The Head Unit or Gateway communicates on behalf of ECUs (specialized processors on internal networks) that face constraints imposed by the demands of high performance real-time machines, and performance requirements from a driver, passengers, and outside highway environmental factors (e.g., road conditions, traffic density, lighting, weather, etc.).

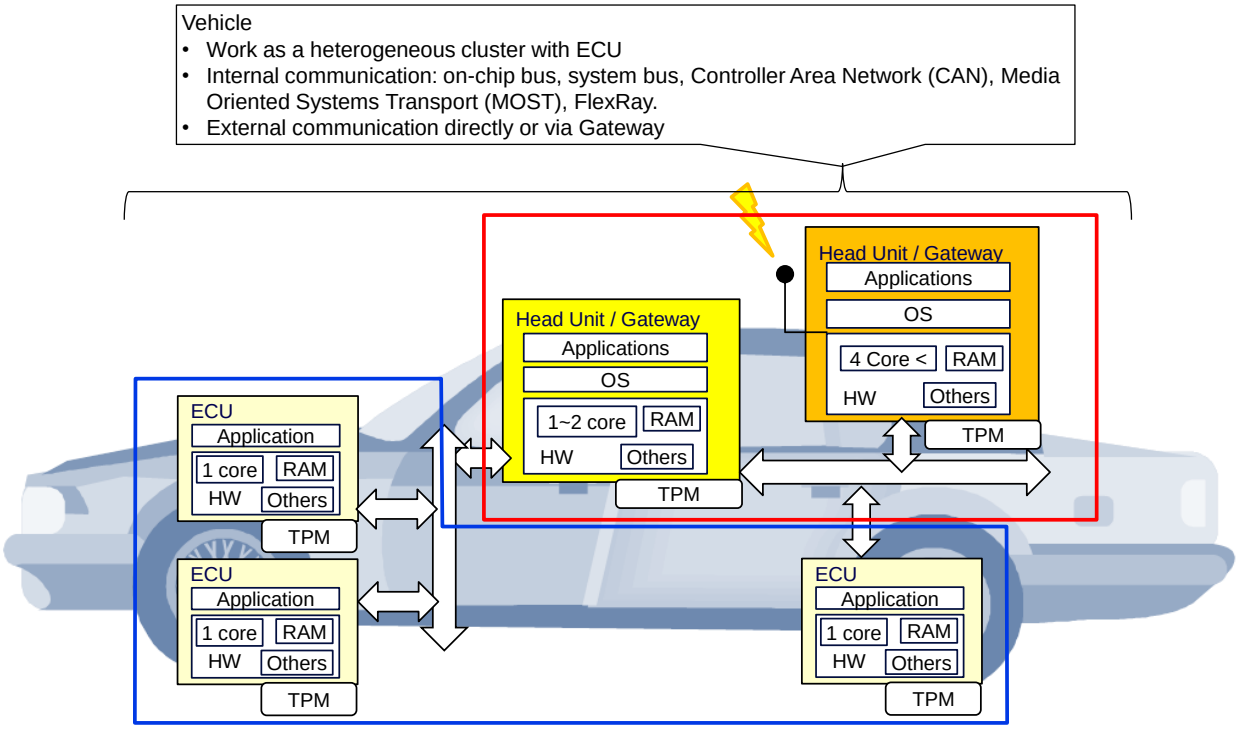

Figure 1: Overview of an automotive vehicle

2.1 Automotive Vehicle Terms

Device: A networked hardware component (which may contain multiple CPUs and areas of ROM, RAM, NVRAM memory), also be known as network equipment or a simply a computer.

Firmware: In electronic systems and computing, firmware is the combination of persistent memory and program code and data stored in it.

Controller Area Network (CAN): CAN bus is a vehicle bus standard designed to allow microcontrollers and devices to communicate with each other within a vehicle without a host computer. CAN bus is a message-based protocol, designed specifically for automotive applications but is now also used in many other applications.

Electronic Control Unit (ECU): ECU is a generic term for any embedded system that controls one or more of the electrical systems or subsystems in a motor vehicle

FlexRay: FlexRay is an automotive network communications protocol developed by the FlexRay Consortium to govern on-board automotive computing. It is designed to be faster and more reliable than CAN, but it is also more expensive.

Head-Unit: Typically contained in a radio/CD/entertainment console that includes Internet connectivity (e.g., WiFi) and a communications Gateway for the ECUs in the industrial control operational network(s) of the automotive vehicle

Gateway: A Gateway is an inter-network processor, i.e., a special-purpose processor which aids in the interconnection of networks. When two or more networks do not use the physical and datalink protocols for the purpose of communication, they can be interconnected via gateways, using protocol conversion processes. In addition, gateways require congruent or at least mutually acceptable administrative procedures between the interconnected networks. The duties of a gateway are usually much more complex than those of switches or routers.

Media Oriented Systems Transport (MOST): MOST is a high-speed multimedia network technology optimized by the automotive industry. It can be used for applications inside the car.

Remote Center: A remote vehicle safety and maintenance center, typically operated by manufacturer or government agency

System-on-Chip (SoC): An SoC is an integrated circuit (IC) that integrates all components of a computer or other electronic system into a single chip. It may contain digital, analog, mixed-signal, and often radio-frequency functions - all on a single chip substrate. SoCs are very common in the mobile electronics market because of their low power consumption. A typical application is in the area of embedded systems.

2.2 Automotive Vehicles are Composite Systems

Given the diverse use cases inside the vehicle, it is reasonable to describe a vehicle as a composite industrial control system network with one or more Internet Gateways and one or more human user interfaces. Due to the complexity of this automotive vehicle model, this Library Profile for Automotive Thin is limited to a definition of the functionality of a TPM that can be deployed into each resource-constrained ECU within the vehicle.

Some of the fundamental differences between the PC/tablet/mobile platform model and the automotive model include:

- ECUs have robust physical and performance requirements (temperature, vibration, acceleration, etc.) that are typically far more demanding than they are for PC/tablet/mobile devices.
- ECUs have low availability and low speed of ROM, RAM, and Non-Volatile (NV) memory
- ECUs have sophisticated power management, including how continuously variable low power and standby power states and applications are not normally aware of power transitions.
- Some ECUs do not have a BIOS or a conventional OS (that dispatches distinct application processes) – they may only contain a single thread of firmware that calls a minimal runtime-library. In the past, ECU firmware was typically immutable and also implemented its own integrity verification method, but ECU firmware should be upgradeable via secure methods.

- The expected life cycle of an automotive model, typically twenty or more years, which is much longer than the expected life of other systems that use TPM such as laptop PC or mobile phone.

3 Automotive-Rich Profile and Automotive-Thin Profile – Conceptual Model (Informative)

Based on the automotive model above, a conceptual model composed of two types of TPMs could be suitable for automotive vehicle deployments. One kind of TPM in a Head Unit or Gateway (that communicates directly with the public Internet) could have rich capabilities and be called “Automotive-Rich.” The other kind of TPM built into an ECU could have significantly less capability and be called “Automotive-Thin.” Because most of the ECUs in a vehicle have limited processing, networking, and applications functionality, the Automotive-Thin profile for an individual ECU does not need to be capable of supporting a complex implementation of the TPM 2.0 Library specification [1] [2] [3] [4]. In other words, an Automotive-Thin TPM is intended to sufficient for handling each ECU’s basic hardware root-of-trust needs. In this section, Automotive-Rich and -Thin profiles are described.

3.1 Automotive-Rich Profile – Conceptual Model

In the future, the TCG Embedded Systems Work Group might decide to define a TPM 2.0 Library Profile for Automotive-Rich. However, to facilitate this current TPM 2.0 Library Profile for Automotive-Thin specification, below is a summary of an Automotive-Rich TPM that simply describes the necessary and sufficient capabilities of an Automotive-Rich TPM to support the implementation of Automotive-Thin TPMs in ECUs.

Potential characteristics of an Automotive-Rich TPM for a Head Unit or Gateway include:

- Supports a TPM command list similar to PC Client Platform TPM Profile [7] – rich Head-Unit/Gateway platform functionality would make it practical to use a full TPM 2.0 implementation
- Supports management of multiple Automotive-Thin TPMs (one in each ECU)
- Supports a Gateway between the Remote Center and ECUs on non-Internet industrial control internal networks
- Supports a local certificate store and management services for ECUs with Automotive-Thin TPMs, in cases of lost communications between Remote Center and automotive vehicle
- Supports compatibility with existing vehicle management applications that communicate with Remote Center for vehicle safety and maintenance

4 Scenarios for usage of Automotive-Thin Profile (Informative)

4.1 Introduction

This specification defines a TPM 2.0 Library Profile for Automotive-Thin. Significant characteristics of an Automotive-Thin TPM include:

1. Often deployed in support of resource-constrained ECUs to support their integrity and attestation for remote maintenance services

2. Supports storage of ECU firmware measurements, creation of integrity digests, and creations of signatures on integrity digests

3. After receiving a firmware update or patch, an ECU may use an Automotive-Thin TPM to verify signatures and to help confirms to a Remote Center that an update installation was completed successfully

.

4.2 Example of both Automotive-Rich and Automotive-Thin TPMs in a vehicle

Here we show an example where both Automotive-Rich and Automotive-Thin TPMs are deployed in a vehicle body. Message flows based on this example are described below in Sections 4.5 and 4.6.

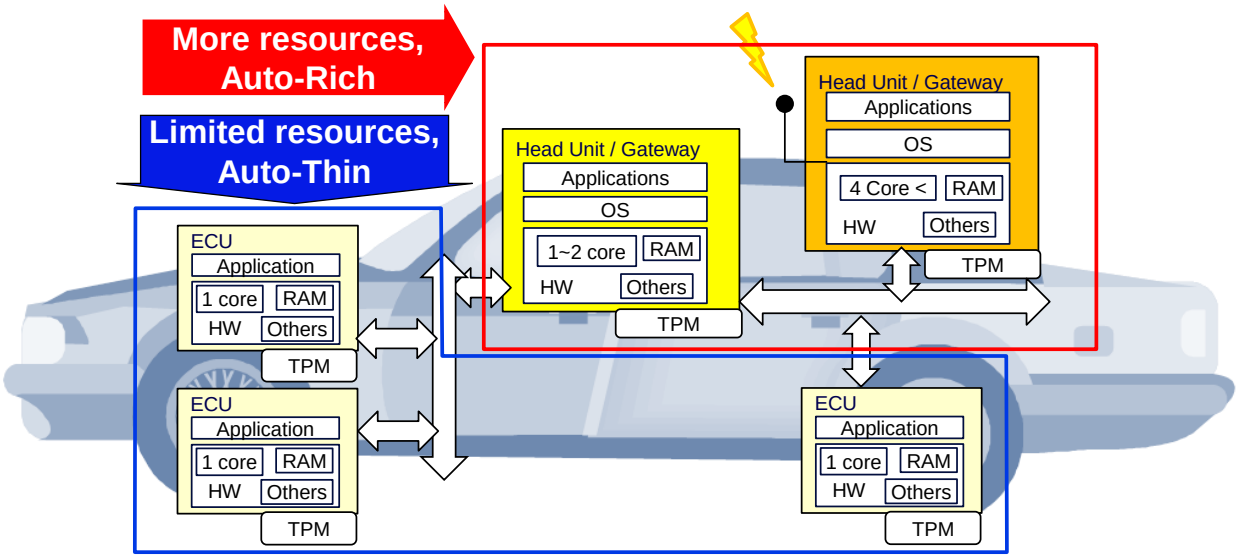

Figure 2: Automotive-Rich and Automotive-Thin TPMs implemented in a vehicle

As mentioned above in Section 2 above, the number of ECUs in a modern vehicle is commonly over 100. For the case where each ECU has its own Automotive-Thin TPM, the number of Automotive-Thin TPMs may be over 100. This is the reason that the Automotive-Rich Profile could have support for individual "shadowing" in NVRAM and also for aggregating of the integrity measurements from the many Automotive-Thin TPMs.

4.3 Example of only Automotive-Thin TPMs in a vehicle

Here we show an example where only Automotive-Thin TPMs are deployed in a vehicle. Message flows based on this example are described in section 4.7 below.

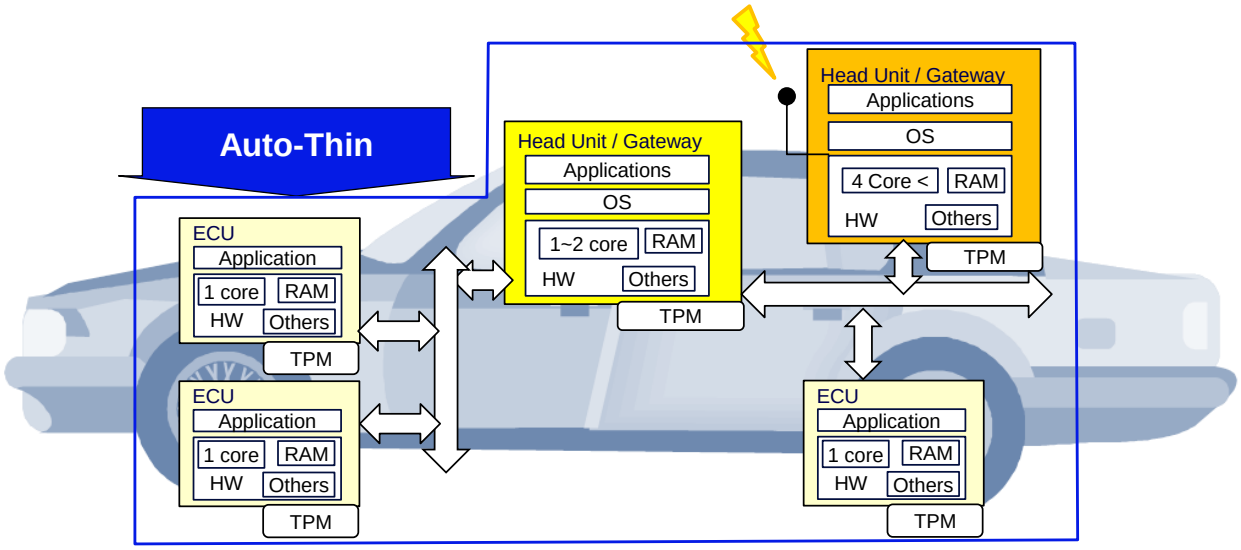

Figure 3: Only Automotive-Thin TPMs implemented in a vehicle

4.4 Message flows for Remote Maintenance

Here we show an example of message flows for a use case of remote maintenance of firmware, where an integrity digest is used to verify an ECU firmware update or patch. Because of this focus, details related only to real time vehicle operations performed by ECUs (brakes, lights, engine, etc.) will be ignored. Remote vehicle maintenance could be done periodically and/or in vehicle off times (i.e. vehicle parked with ignition off). When vehicle recalls could occur based only on software implementation defects, not caused by hardware issues, the remote vehicle maintenance method could be used to solve these recall issues without a dealer or repair shop visit.

Figure 4 shows the message flow for each component (Head Unit/Gateway or ECU) for remote maintenance handled by Automotive-Rich, and -Thins.

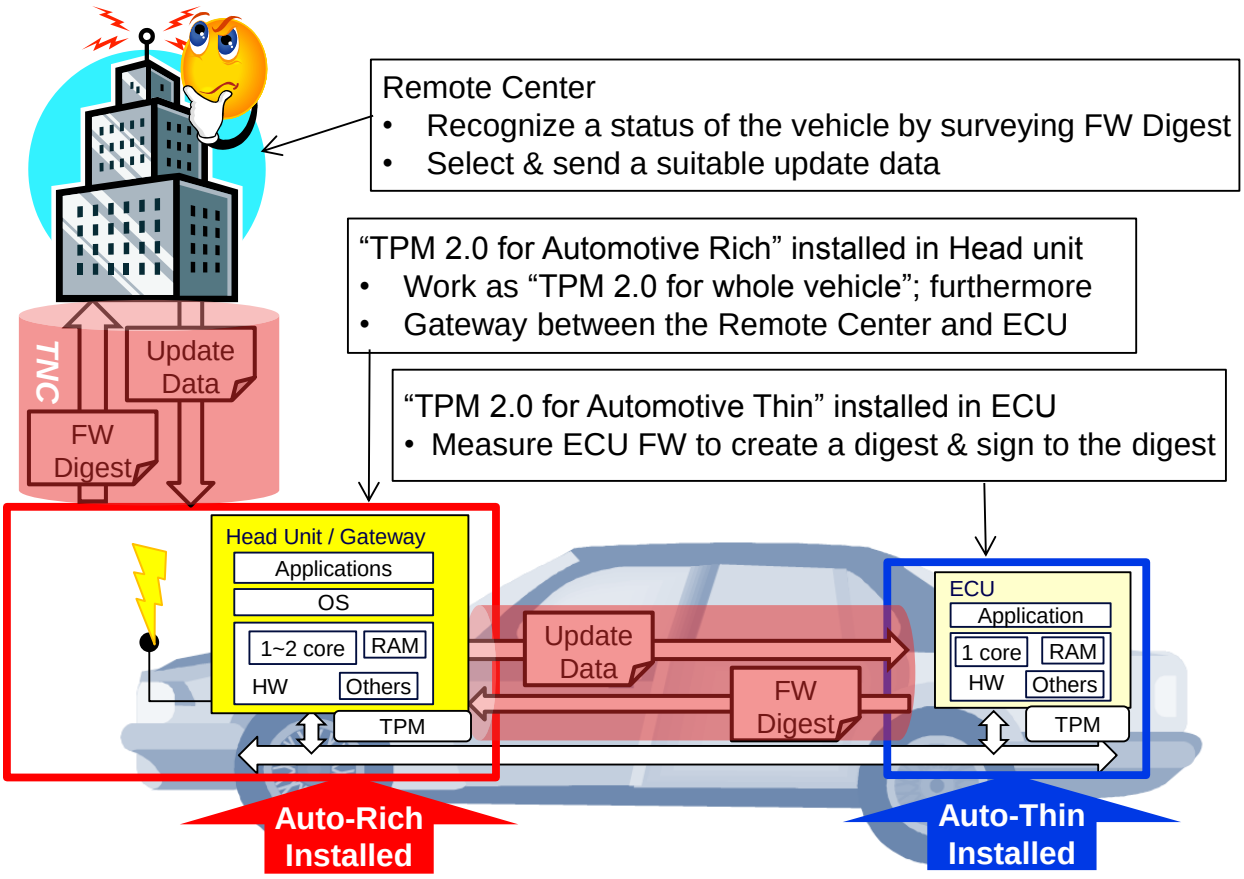

Figure 4: Message Flow for Remote Vehicle Maintenance

In this figure, the role of Vehicle Manufacturing Center (VMC) is also included.

4.5 Message Flows where Head Unit checks ECU signatures

Here we show an example of message flows where the Head Unit or Gateway uses its Automotive-Rich TPM to check the signatures on integrity reports created by each ECU with its own Automotive-Thin TPM. Each ECU's Automotive-Thin TPM has been pre-provisioned with an Endorsement Key (EK) at the time of ECU installation (during vehicle manufacturing or when replaced by a dealer or repair shop). Each ECU's Automotive Thin TPM EK public key has been registered with the Head Unit or Gateway and the Remote Center and the Remote Center has generated an EK certificate at the time of ECU installation in the vehicle.

After checking the signatures in ECU integrity reports with its Automotive-Rich strips the ECU signatures from the original integrity reports and signs the collection of integrity reports with its own Automotive-Rich TPM and sends them to the Remote Center, providing assurance that only well-known ECUs are being reported for the correct vehicle (identified via the Automotive-Rich TPM's EK). If an ECU integrity report signature fails validation by the Automotive-Rich TPM, then the Head Unit or Gateway reports the rogue ECU to the Remote Center.

The summary of Pros and Cons is shown below.

Pros: The number of public signature keys that the Remote Center has to store can be reduced

Cons: The work for signature generation and validation is increased for Automotive-Rich TPM

Note: The network connections and operation requests flow from right to left in these figures.

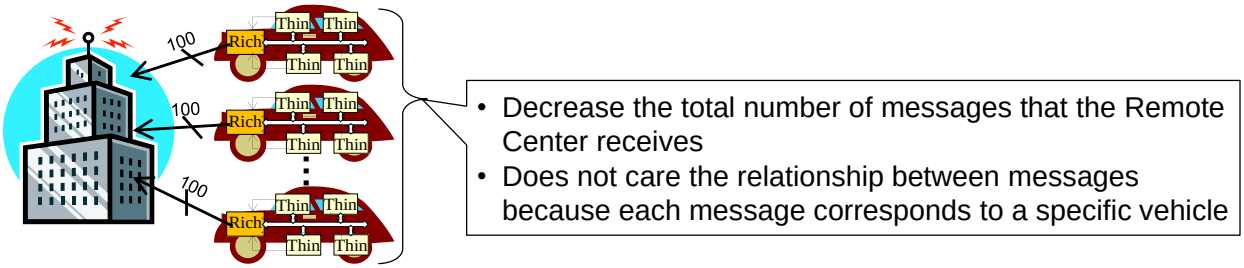

Figure 5: Head Unit that checks ECU signatures summary

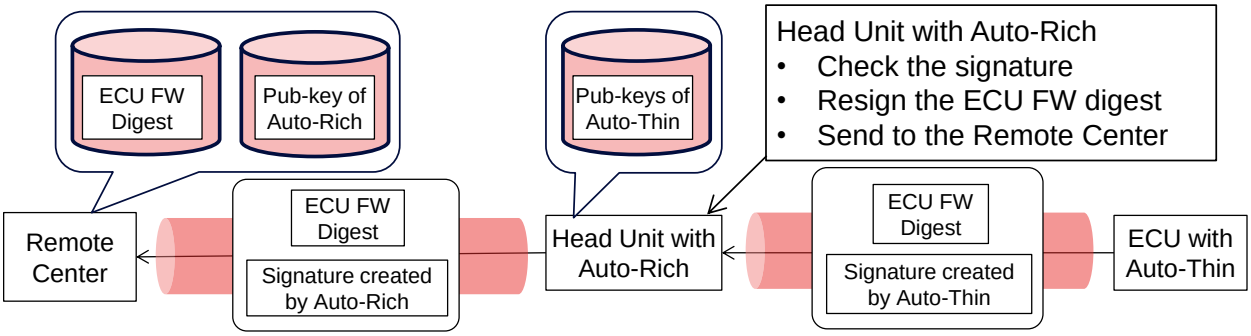

Figure 6: Head Unit that checks ECU signatures details

The Head Unit or Gateway with the Automotive-Rich TPM combines the integrity report messages created by each of the vehicle's installed ECUs with their respective Automotive-Thin TPMs. To minimize the number of signatures that have to be checked at the Remote Center or Vehicle Manufacturing Center (VMC), the Head Unit or Gateway can collect the integrity measurements from multiple ECUs and sign the whole collection before forwarding it to the VMC. Note that each ECUs Automotive-Thin TPM PCR0 value is "shadowed" in an NV index in the Head Unit or Gateway's own Automotive-Rich TPM.

As a practical example of this procedure is illustrated below,

1. **Provisioning steps during vehicle manufacturing:**

1-1. Manufacturer ECU provisioning software sends Automotive-Thin TPM commands to generate an EK and a child key of the EK as a signing key, and then tells the Automotive-Thin TPM to keep the signing key persistently loaded in the Automotive-Thin TPM.

⇒ TPM2_CreatePrimary & TPM2_Create & TPM2_Load & TPM2_EvictControl

1-2. ECU reads out public part of the new signing key, then sends it to Remote Center.

⇒ TPM2_ReadPublic

1-3. Remote Center generates the signing key certificate, and then sends it to Head Unit or Gateway with the Automotive-Rich TPM for each ECU...

2. **Measuring ECU firmware and confirming ECU firmware update completion:**

2-1. Manufacturer ECU provisioning software sends commands to each ECU to read out the ECU firmware, and then the ECU Automotive-Thin TPM generates its digest.

⇒ TPM2_HashSequenceStart & TPM2_SequenceUpdate & TPM2_SequenceComplete

⇒ TPM2_PCR_Extend

2-2. Each ECU signs its own firmware digest with the signing key stored in its own Automotive-Thin TPM, and then sends both digest and signature to the Head Unit or Gateway.

⇒ TPM2_Quote

2-3. Head Unit or Gateway uses its Automotive-Rich TPM to verify the digest and signature from each ECU's Automotive-Thin TPM, strips the original ECU signatures, and then resigns the ECU firmware integrity reports with the Head Unit or Gateway's own Automotive-Rich TPM signature key, and Head Unit sends the collection of firmware integrity reports to the Remote Center.

2-4. Remote Center verifies the signature from Head Unit, and attests the condition of the firmware in each installed ECU. In the case that the Remote Center determines that a specific ECU's firmware should be updated, the subsequent procedure is the following.

3. **Installing an ECU firmware patch:**

3-1. Remote Center selects a suitable update patch and signs with the Remote Center's private signature key, and sends it to Head Unit. The Head Unit trusts all software signed by the Remote Center (by law in Japan and Europe).

3-2. Head Unit verifies the signature with its Automotive-Rich TPM on the firmware patch, based on factory provisioning of public keys and certificates for each ECU, strips the original Remote Center signature from the patch, and resigns it with the Head Unit's private signature key, and then Head Unit sends the firmware patch to ECU for signature verification with its own Automotive-Thin TPM.

3-3. ECU applies the received and verified firmware patch and confirms success to the Head Unit.

3-4. ECU with Automotive-Thin, Head Unit with Automotive-Rich and Remote Center each re-measure ECU firmware in the same way as from 2-1 to 2-4 to confirm the successful update completion.

4. **Rekeying ECUs when vehicle is sold to another owner:**

4-1. (Same method shown above in 1) At ownership change time, each ECU uses its Automotive-Thin TPM to generate a child key of its EK as a new signing key and reads out the public key part.

4-2. Each ECU uses its Automotive-Thin TPM to sign the public part of the new signing key by the old signing key, and sends it with the signature to Remote Center via the Head Unit proxy.

⇒ TPM2_Certify

4-3. Remote Center verifies each message based on the old signing key, and generates the new sign key certificate, then sends it to Head Unit for storage in its Automotive-Rich TPM and Head Unit sends acknowledgment to ECU

4-4. ECU deletes the old signing key from its Automotive-Thin TPM.

4.6 Message Flows where Head Unit that does not check ECU signatures

Here we show an example of message flows where the Head Unit or Gateway with the Automotive-Rich TPM does not check the signatures sent by each ECU from its own Automotive-Thin TPM. The Head Unit or Gateway creates a TNC [10] connection to the Remote Center and simply forwards the integrity reports that each ECU has signed with its own Automotive-Thin TPM to the Remote Center.

The summary of Pros and Cons is shown below.

Pros: The total manufacturing costs of the Head Unit and its Automotive-Rich TPM can be minimized

Cons: The work for signature validation is increased in the Remote Center

Note: The network connections and operation requests flow from right to left in these figures.

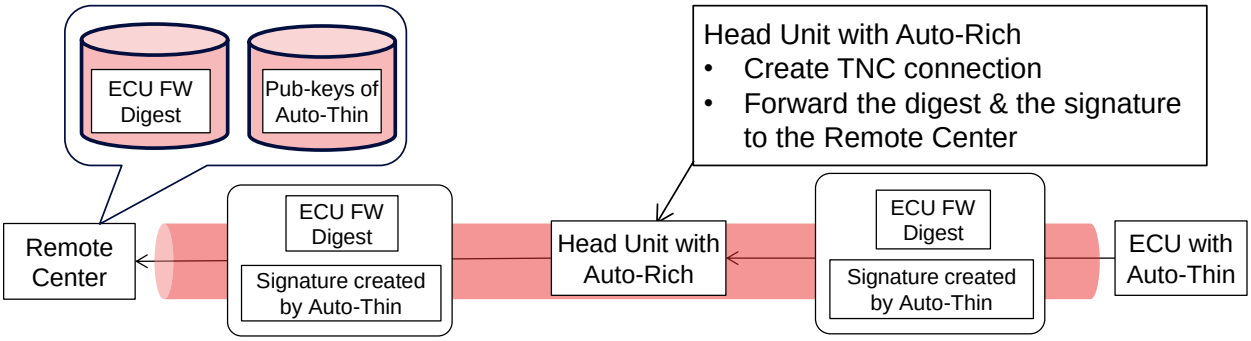

Figure 7: Head Unit that does not check ECU signatures

4.7 Messages flows for Remote Maintenance with only Automotive-Thin TPMs

Here we show an example of message flows for an alternative use case of remote maintenance using only Automotive-Thin TPMs in a vehicle (even in the Head Unit or Gateway). The Vehicle Manufacturing Center has direct Internet connections to both the Head Unit and all of the ECUs in the vehicle.

Note: In this example, it might be necessary that all of the ECUs use Ethernet-like cabling to directly support the Internet TCP/IP protocol suite.

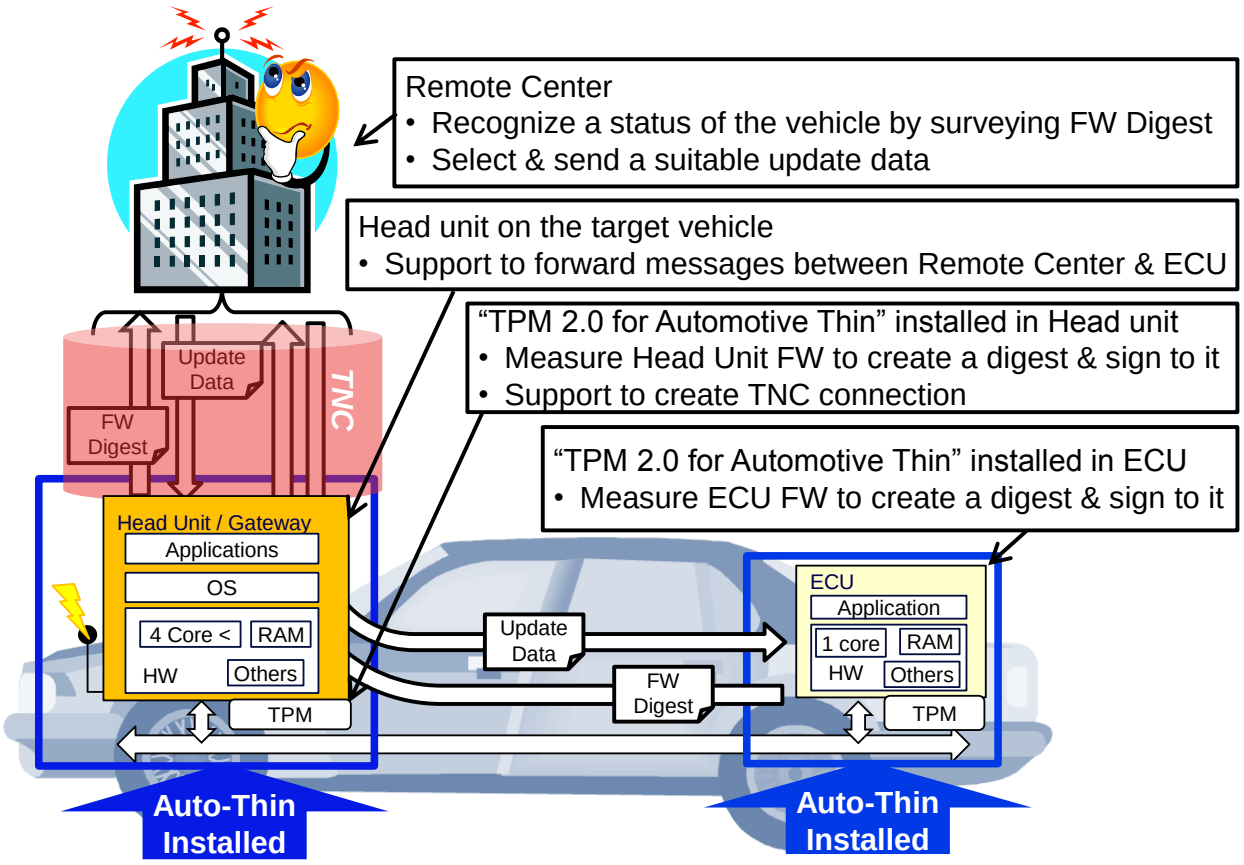

Figure 8: Message Flow for Remote Maintenance with only Automotive-Thin TPMs summary

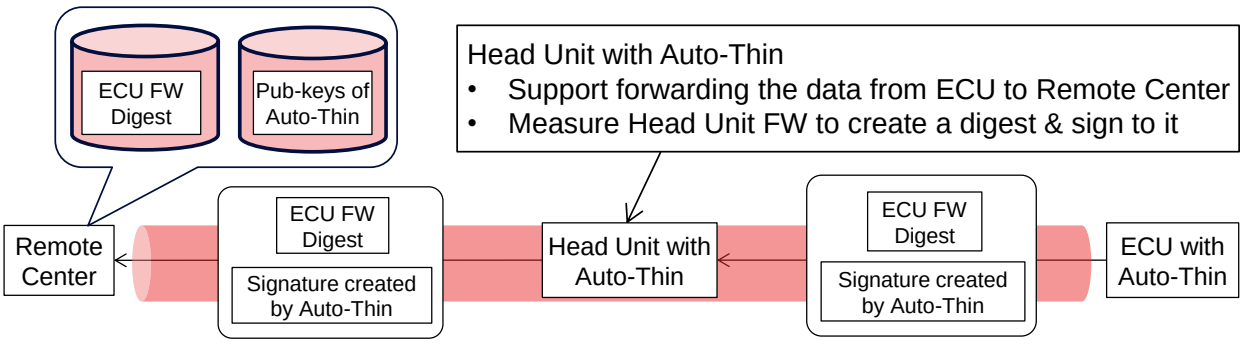

Figure 9: Message Flow for Remote Maintenance with only Automotive-Thin TPMs details

The potential role of an Automotive-Thin TPM in a Head Unit is shown in Figure 9. One of the primary responsibilities of the Automotive-Thin TPM in this Head Unit is in supporting the forwarding of messages between the Remote Center and each ECU. In addition, the sequence of operations for remote maintenance is similar to what was described in Section 4.6 except that there is no Automotive-Rich TPM involved.

5 Definition of Automotive-Thin Profile

This section describes the platform-specific requirements for a TPM 2.0 implementation compliant with this Automotive-Thin Profile.

5.1 Mandatory TPM 2.0 Library Specification Version

A TPM 2.0 implementation compliant with this Automotive-Thin Profile SHALL be compliant with the *TPM Library Specification, Family 2.0, Level 00, Revision 01.16* or later TCG published version.

5.2 Mandatory Platform Constants

A TPM 2.0 implementation compliant with this Automotive-Thin Profile SHALL supports the required platform-specific constants in Table 1:

Table 1 – Mandatory Platform Constants

Property	Value	Comment
TPM_PS_FAMILY_INDICATOR	0x00000009	Embedded Platform TPM Specification, which is defined in TPM 2.0 Part 2-Structures [2]
TPM_PS_LEVEL	0x00000000	The level of the TPM 2.0 Automotive-Thin Specification
TPM_PS_REVISION	0x00000100	The revision of the TPM 2.0 Automotive-Thin Specification
TPM_PS_DAY_OF_YEAR	0x00000000	The day of the year of the implemented TPM 2.0 Automotive-Thin Profile publication
TPM_PS_YEAR	0x00000000	The year of the implemented TPM 2.0 Automotive-Thin Profile publication

5.3 Mandatory Algorithms

A TPM 2.0 implementation compliant with this Automotive-Thin Profile SHALL supports the following mandatory algorithms:

- At least one of RSA 2048 or ECC P256. Additional asymmetric algorithms and key sizes are allowed.
- At least one symmetric algorithm. AES 128 is recommended, others are allowed.
- CFB mode.
- SHA-256. Other hash algorithms are allowed.
- HMAC.

5.4 Conditionally Mandatory RSA Constants

If RSA is implemented, then a TPM 2.0 implementation compliant with this Automotive-Thin TPM SHALL support RSA key sizes of 2048 bits and key constants specified in Table 2. Other RSA key sizes of 1028 bits may be implemented.

Table 2 – Conditionally Mandatory RSA Algorithm Constants

Name	Value	Comments
RSA_KEY_SIZES_BITS	{1024, 2048}	braces because this is a list value
MAX_RSA_KEY_BITS	2048	
MAX_RSA_KEY_BYTES	((MAX_RSA_KEY_BITS + 7) / 8)	

5.5 Conditionally Mandatory ECC Constants

If ECC is implemented, then a TPM 2.0 implementation compliant with this Automotive-Thin TPM SHALL supports NIST_P256 and BN_P256 ECC curves and the following ECC constants in Table 3. Other curves listed in the TCG Algorithm Registry may be implemented.

Table 3 – Conditionally Mandatory ECC Algorithm Constants

Name	Value	Comments
ECC_CURVES	{TPM_ECC_NIST_P256, TPM_ECC_BN_P256}	
ECC_KEY_SIZES_BITS	{256}	this is a list value with length of one
MAX_ECC_KEY_BITS	256	
MAX_ECC_KEY_BYTES	((MAX_ECC_KEY_BITS + 7) / 8)	

5.6 Mandatory and Recommended TPM 2.0 Commands

A TPM 2.0 implementation compliant with this Automotive-Thin Profile SHALL implement the commands listed as mandatory (M) in Table 4. Commands listed as recommended (R) SHOULD be implemented for interworking. All other TPM 2.0 commands are optional.

Table 4 – Mandatory and Recommended TPM 2.0 Commands

Name	Profile Support
TPM2_Startup	M
TPM2_Shutdown	M
TPM2_SelfTest	M
TPM2_GetTestResult	M
TPM2_StartAuthSession	M
TPM2_Create	M
TPM2_Load	M
TPM2_ReadPublic	M
TPM2_Hash	M
TPM2_Certify	M
TPM2_Quote	M

Name	Profile Support
TPM2_Sign	M
TPM2_PCR_Extend	M
TPM2_PCR_Read	M
TPM2_PCR_Reset	O
TPM2_CreatePrimary	M
TPM2_EvictControl	M
TPM2_GetCapability	M
TPM2_HashSequenceStart	R
TPM2_SequenceUpdate	R
TPM2_SequenceComplete	R
TPM2_EventSequenceComplete	R
TPM2_IncrementalSelfTest	O
TPM2_PolicyRestart	O
TPM2_LoadExternal	R
TPM2_ActivateCredential	O
TPM2_MakeCredential	O
TPM2_Unseal	O
TPM2_ObjectChangeAuth	O
TPM2_Duplicate	O
TPM2_Rewrap	O
TPM2_Import	O
TPM2_RSA_Encrypt	O
TPM2_RSA_Decrypt	O
TPM2_ECDH_KeyGen	O
TPM2_ECDH_ZGen	O
TPM2_ECC_Parameters	O
TPM2_ZGen_2Phase	O
TPM2_EncryptDecrypt	R
TPM2_HMAC	R
TPM2_GetRandom	O
TPM2_StirRandom	O
TPM2_HMAC_Start	O
TPM2_CertifyCreation	O
TPM2_GetSessionAuditDigest	O

Name	Profile Support
TPM2_GetCommandAuditDigest	O
TPM2_GetTime	O
TPM2_Commit	O
TPM2_EC_Ephemeral	O
TPM2_VerifySignature	R
TPM2_SetCommandCodeAuditStatus	O
TPM2_PCR_Event	O
TPM2_PCR_Allocate	O
TPM2_PCR_SetAuthPolicy	O
TPM2_PCR_SetAuthValue	O
TPM2_PolicySigned	O
TPM2_PolicySecret	O
TPM2_PolicyTicket	O
TPM2_PolicyOR	O
TPM2_PolicyPCR	O
TPM2_PolicyLocality	O
TPM2_PolicyNV	O
TPM2_PolicyCounterTimer	O
TPM2_PolicyCommandCode	O
TPM2_PolicyPhysicalPresence	O
TPM2_PolicyCpHash	O
TPM2_PolicyNameHash	O
TPM2_PolicyDuplicationSelect	O
TPM2_PolicyAuthorize	O
TPM2_PolicyAuthValue	O
TPM2_PolicyPassword	O
TPM2_PolicyGetDigest	O
TPM2_PolicyNvWritten	O
TPM2_HierarchyControl	O
TPM2_SetPrimaryPolicy	O
TPM2_ChangePPS	O
TPM2_ChangeEPS	O
TPM2_Clear	O
TPM2_ClearControl	O

Name	Profile Support
TPM2_HierarchyChangeAuth	O
TPM2_DictionaryAttackLockReset	O
TPM2_DictionaryAttackParameters	O
TPM2_PP_Commands	O
TPM2_SetAlgorithmSet	O
TPM2_FieldUpgradeStart	O
TPM2_FieldUpgradeData	O
TPM2_FirmwareRead	O
TPM2_ContextSave	O
TPM2_ContextLoad	O
TPM2_FlushContext	R
TPM2_ReadClock	O
TPM2_ClockSet	O
TPM2_ClockRateAdjust	O
TPM2_TestParms	O
TPM2_NV_DefineSpace	O
TPM2_NV_UndefineSpace	O
TPM2_NV_UndefineSpaceSpecial	O
TPM2_NV_ReadPublic	O
TPM2_NV_Write	R
TPM2_NV_Increment	R
TPM2_NV_Extend	R
TPM2_NV_SetBits	O
TPM2_NV_WriteLock	O
TPM2_NV_GlobalWriteLock	O
TPM2_NV_Read	R
TPM2_NV_ReadLock	O
TPM2_NV_ChangeAuth	O
TPM2_NV_Certify	O

5.7 Mandatory PCR Support

A TPM 2.0 implementation compliant with this Automotive-Thin Profile SHALL support PCR0, whose default reset state is 0..0. PCR0 may only be reset by TPM2_Startup().

Support for any other PCR is optional. The default reset state for all optional PCR is 0..0.

5.8 Mandatory Locality Support

A TPM 2.0 implementation compliant with this Automotive-Thin Profile SHALL support locality 0 and may support other localities.

5.9 Recommended NV Storage Capabilities

A TPM 2.0 implementation compliant with this Automotive-Thin Profile SHOULD supports the non-volatile (NV) storage recommendations in Table 5.

The following table includes minimum NV storage recommendations

Table 5 – Recommended NV Storage Capabilities

Recommendation	Value	Comments
Minimum size for NV area	1.6KBytes minimum	This indicates the minimum amount of total NV space that can be used for the NV commands. This does not include NV storage for pre-defined TPM internal data.
Minimum number of counter indices	4	Corresponds to the TPMA_NV_COUNTER bit.
Minimum number of reserved PCR-style indices	4	Corresponds to the TPMA_NV_EXTEND bit.
Minimum number of reserved bit fields	0	Corresponds to the TPMA_NV_BITS bit.
Minimum number of hybrid indices	0	Corresponds to the TPMA_NV_ORDERLY bit.
Minimum number of persistent objects	7	Corresponds to TPM_PT_HR_PERSISTENT_MIN.

The following table is used to indicate reserved indices. The ranges are reserved by the TCG Technical Committee (TC) in the TCG Registry of Reserved TPM 2.0 Handles and Localities.

Table 6 – Recommended Reserved NV Index Storage

Reserved Indices	Recommendation	Value	Comments
EK Certificates	1.6K Bytes minimum	0x01C00000	There may be one or more “EK” cert defined based on algorithms and support by vendors.

NOTE For EK Certificates in Table 6, please refer to Figure 6. To support the checking process of the signature of Automotive-Thin TPM in Automotive-Rich TPM, the EK Certificates stored in Automotive-Rich TPM and Automotive-Thin TPM should be useful.

5.10 Mandatory Reserved Handles

TPM 2.0 implementations of the Automotive-Thin Profile SHALL reserves specific handle for keys such as an EK. This is used to store an “endorsement key” that can be used for platform authentication and identification.

5.11 Mandatory Resource Minimums and Maximums

TPM 2.0 implementations of the Automotive-Thin Profile SHALL meet the following constraints in Table 7 for minimum and maximum resources supported.

Table 7 – Mandatory Resource Minimums and Maximums

Resource Type	Minimum	Maximum	Comments
Active Sessions	MUST be at least 3	None specified	Recommend implementations be consistent with PC-Client specification requirements
Concurrent loaded sessions	MUST be at least 3	None specified	Recommend implementations be consistent with PC-Client specification requirements
Concurrent loaded objects	MUST be at least 2	None specified	Recommend implementations be consistent with PC-Client specification requirements

5.12 Mandatory Hierarchy Support

TPM 2.0 implementations of the Automotive-Thin Profile SHALL support the Platform Hierarchy and the Endorsement Hierarchy and may support the Storage Hierarchy and Null Hierarchy.

5.13 Required Implementation Values

There are no required implementation values for a TPM 2.0 implementation of the Automotive-Thin Profile.

5.14 Dictionary Attack Parameters

There are no minimum or maximum dictionary parameters for TPM 2.0 implementations compliant with this Automotive-Thin Profile.

5.15 Mandatory Platform Interface Indication Support

TPM 2.0 implementations of the Automotive-Thin Profile SHALL support the platform interface indications in Table 8.

Table 8 – Mandatory Platform Interface Indications

Platform Interface Indication	Required	Requirements
_TPM_Init	Y	This interface indication shall be supported. There are no requirements on how this interface indication is provided to the TPM.

6 References

[1] Trusted Computing Group, *Trusted Platform Module Library, Part 1: Architecture*, Family 2.0, current TPM 2.0 specification level

[2] Trusted Computing Group, *Trusted Platform Module Library, Part 2: Structures*, Family 2.0, current TPM 2.0 specification level

[3] Trusted Computing Group, *Trusted Platform Module Library, Part 3: Commands*, Family 2.0, current TPM 2.0 specification level

[4] Trusted Computing Group, *Trusted Platform Module Library, Part 4: Supporting Routines*, Family 2.0, current TPM 2.0 specification level

[5] Internet Engineering Task Force, *Guidelines for Writing RFC Text on Security Considerations*, RFC 3552, July 2003

[6] Internet Engineering Task Force, *Key words for use in RFCs to Indicate Requirement Levels*, RFC 2119, March 1997

[7] Trusted Computing Group, *PC Client Platform TPM Profile (PTP) Specification,* http://www.trustedcomputinggroup.org/resources/pc_client_platform_tpm_profile_ptp_specification

[8] Trusted Computing Group, *Virtualized Trusted Platform Architecture Specification*, http://www.trustedcomputinggroup.org/resources/virtualized_trusted_platform_architecture_specification

[9] Trusted Computing Group, *TCG Algorithm Registry*, current specification level, http://www.trustedcomputinggroup.org/resources/tcg_algorithm_registry

[10] Trusted Computing Group, *TCG IF-TNCCS (Trusted Network Connect Client-Server) Specification,* current specification level (see also technically equivalent IETF RFC 5793) http://www.trustedcomputinggroup.org/resources/tnc_iftnccs_specification