TCG TPM 2.0 Automotive Thin Profile
For TPM Family 2.0; Level 0

Specification Version 1.01
Revision 15
May 31, 2018

Contact: admin@trustedcomputinggroup.org

TCG Published

TCG

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), Graeme Proudler (Independent), Florian Schreiner (Infineon), Taku Tsukamoto (IPA), Rob Spiger (Microsoft), David Wooten (Microsoft), Yuishi Torisaki (Panasonic), Xavier Boussin (STMicroelectronics), Hisashi Oguma (Toyota)

Disclaimers, Notices, and License Terms

Table of Contents

Tables

Figures

1 Introduction (Informative)

1.1 Purpose

Automotive vehicle solutions have increasingly leveraged information technology solutions to provide many benefits ranging 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 satisfy the requirements of ECUs that perform a limited number of scenarios requiring a 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.

1.2 Scope

The Trusted Computing Group TPM 2.0 Library Specification [1] [2] [3] [4] defines a Trusted Platform Module (TPM). This specification of a TPM 2.0 Automotive-Thin Profile defines a TPM that is applicable to vehicle systems. This TPM 2.0 Automotive-Thin Profile specification includes use cases in order to justify the profile. The use cases involve an `Automotive-Rich’ TPM that has more commands and capabilities than an Automotive-Thin TPM. An Automotive-Rich TPM profile may be defined in another specification.

1.3 Keywords

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 RFC 2119.

1.4 Statement Type

Please note a very important distinction between different sections of text throughout this document. You will encounter two distinctive kinds of text: application notes as informative comments and normative statements. Because most of the text in this Automotive-Thin TPM profile is normative statements, the authors have informally defined it as the default and, as such, have specifically called out text which is informative comment. If a section header is marked as informative, the entire section is informative. Conversely, if a section header is not marked as informative, the entire section is normative.

2 Overview of Automotive 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 Units (ECU). Figure 1 illustrates that ECUs in vehicles communicate via separate (isolated) vehicular networks, and that this specification envisages vehicles where each ECU has access to a TPM (i.e. one TPM per ECU).

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 that face constraints, including those imposed by high performance demands of real time machines, drivers, passengers, and outside environmental factors, in addition to security and safety requirements imposed by regulators and drivers. During a vehicle's long life cycle, the vehicle's systems typically require firmware updates to correct faults or add new features. Updates must maintain the integrity of the vehicle's systems. Hence updates require support from services such as secure transmission, remote attestation, firmware measurements, and so on.

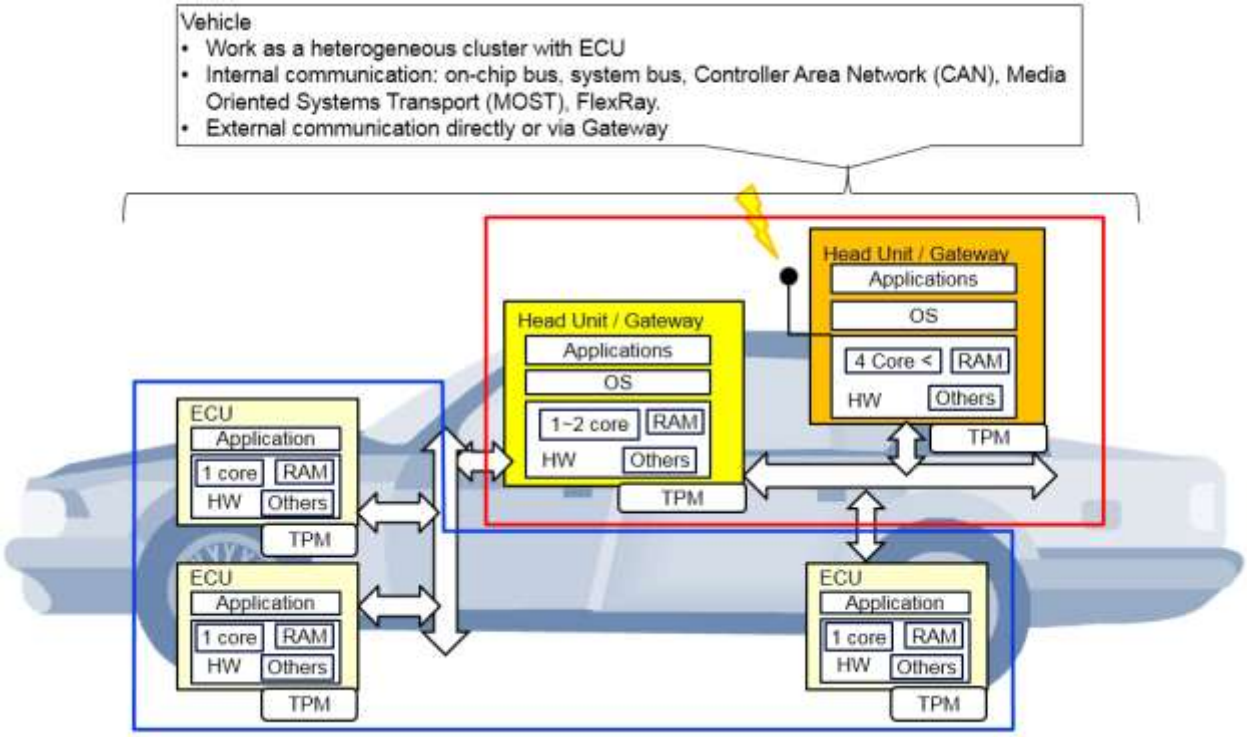

Figure 1: Overview of an automotive vehicle using TPM technology

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 (including System-on-Chip) 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 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 that aids in the interconnection of networks. When two or more networks do not use the same physical and datalink protocols for the purpose of communication, they can interconnect via gateways, using protocol conversion processes.. 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 of SoCs 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. Given the complexity of automotive vehicles, and that usually the computing resources of an ECU are much less than those of a PC, an Automotive Thin TPM is limited to functionality that can be deployed in resource-constrained ([12]) devices.

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 continuously variable low power and standby power states. Also, ECU software applications are not normally notified of power transitions.
- Some ECUs have neither a boot OS nor any 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. Today ECUs are firmware-updateable.
- The expected life cycle of typical automotive systems is twenty or more years, which is much longer than the expected life of other systems.

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. This section describes Automotive-Rich and -Thin profiles.

3.1 Automotive-Rich Profile – Conceptual Model

In the future, the TCG Embedded Systems Work Group might define a TPM 2.0 Library Profile for Automotive-Rich. This is just a summary of an Automotive-Rich TPM that 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 PC Client-like TPM 2.0 implementation
- Supports management of multiple ECU’s with Automotive-Thin TPMs (one in each ECU) by aggregating several measurements from ECUs
- Supports a Gateway between the Remote Center and ECUs on non-Internet industrial control internal networks
- Supports a local certificate store

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

4.1 Introduction

Significant usages of an Automotive-Thin TPM include:

1. Provides support for resource-constrained ECUs to ensure their firmware integrity
2. Supports storage of ECU firmware measurements, creation of integrity digests, and creation of signatures on integrity digests for attestation for remote maintenance services (see 4.5)
3. After receiving and installing a firmware update or patch, an ECU may use an Automotive-Thin TPM to help provide confirmation 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

Figure 2 shows 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 4.6 and 4.7.

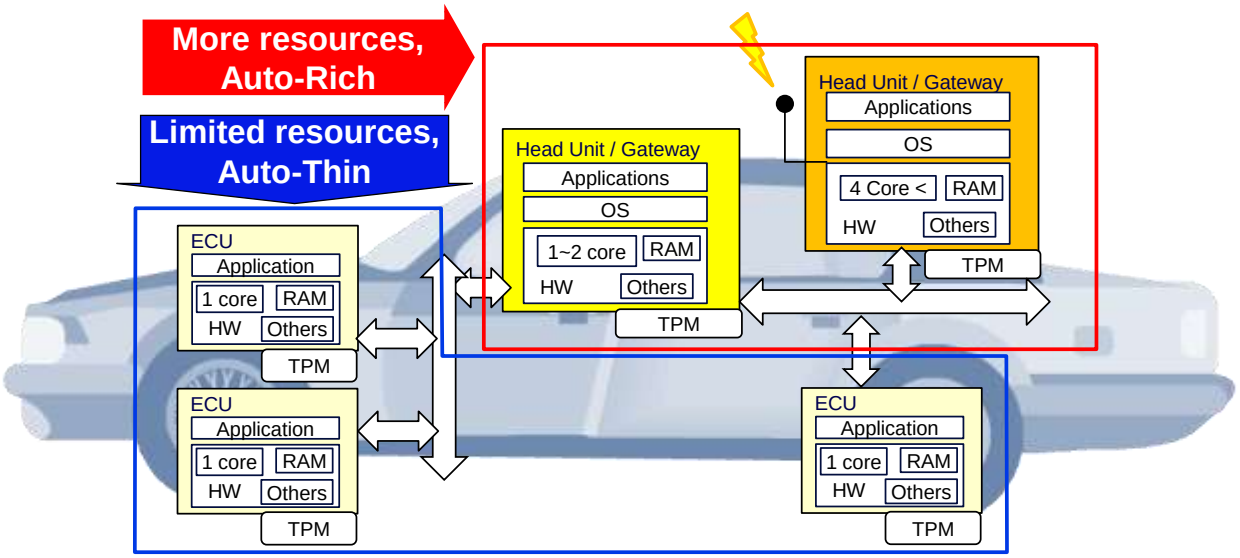

Figure 2: Automotive-Rich and Automotive-Thin TPMs installed in a vehicle

As mentioned in Section 2, 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 store copies of individual Automotive-Thin PCR in its own NVRAM and also aggregate the integrity measurements from the many Automotive-Thin TPMs (see 3.1).

4.3 Example of only Automotive-Thin TPMs in a vehicle

Figure 3 show an example where only Automotive-Thin TPMs are deployed in a vehicle. Message flows based on this example are described in section 4.8 below.

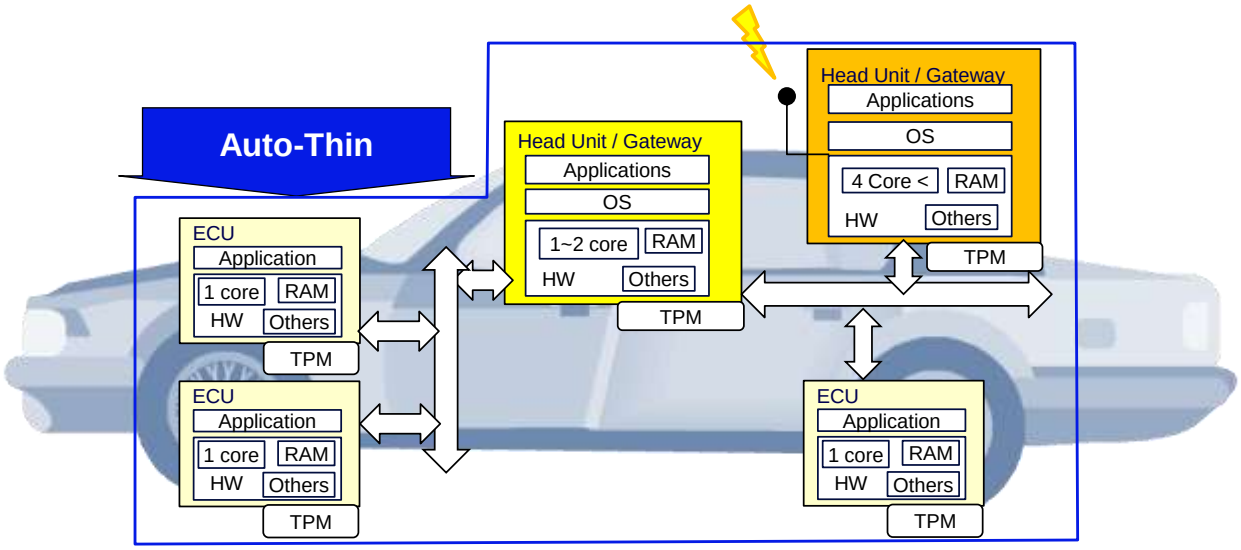

Figure 3: Only Automotive-Thin TPMs installed in a vehicle

4.4 Authorization-based commands usage

Most TPM commands use authorization, thus an authentication value must be provided to TPM by the ECU. This authentication value could be either stored in ECU protected storage or issued by TPM2_Unseal based on PCR values recording the ECU state. Note that high-entropy authorization values are set during manufacturing process.

⇒ TPM2_PolicyPCR & TPM2_Unseal

4.5 Message flows for Remote Maintenance

This section provides an example of message flows for the use case of remote maintenance of firmware, where an integrity digest is used to verify an ECU firmware update or a patch. Details related only to real time vehicle operations performed by ECUs (brakes, lights, engine, etc.) can be ignored. Remote vehicle maintenance could be done periodically and/or in vehicle off times (i.e. vehicle parked with ignition off). Remote vehicle maintenance could be used to replace defective software without visiting a dealer or repair shop. Note that configuration data are not part of this firmware update process and key management is the responsibility of the remote center.

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

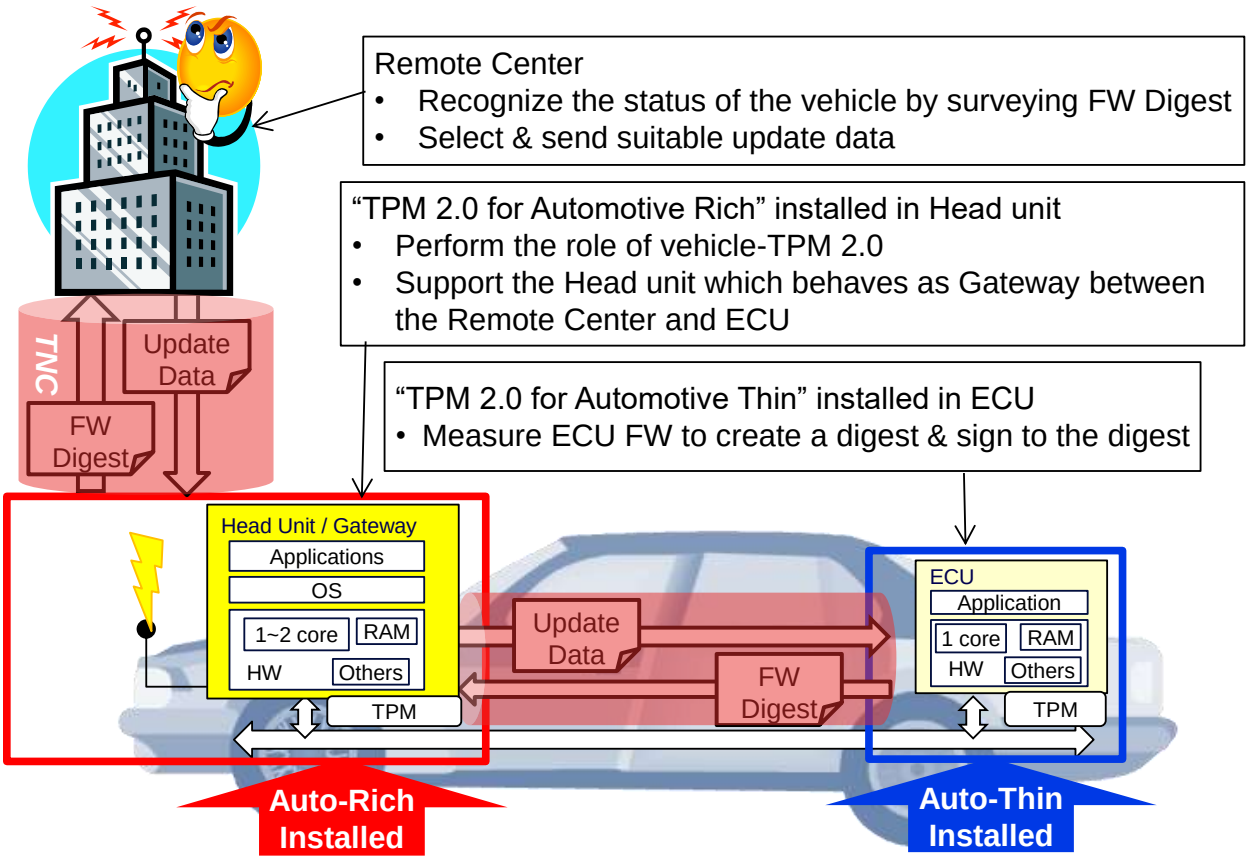

Figure 4: Message Flow for Remote Vehicle Maintenance

4.6 Message flows where Head Unit checks ECU signatures

Figure 5 and Figure 6 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. After checking the signatures in ECU integrity reports with its Automotive-Rich TPM, the Head Unit 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 Automotive-Rich TPM must do more signature generation and validation

Note: The network connections and operation requests flow from right to left in Figure 5 and Figure 6.

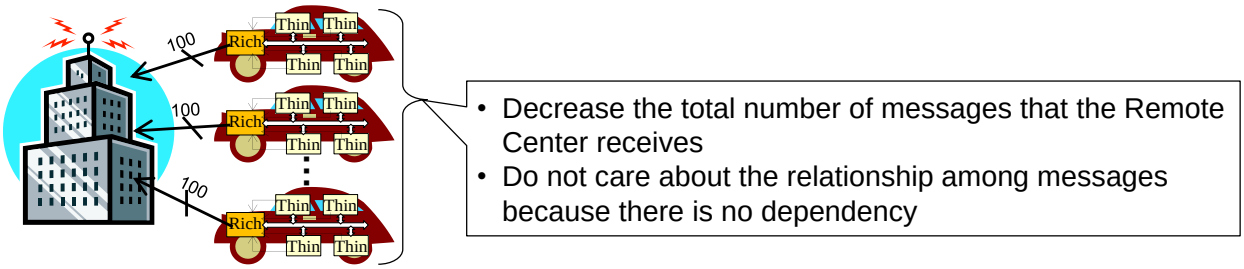

Figure 5: Head Unit that checks ECU signatures (summary)

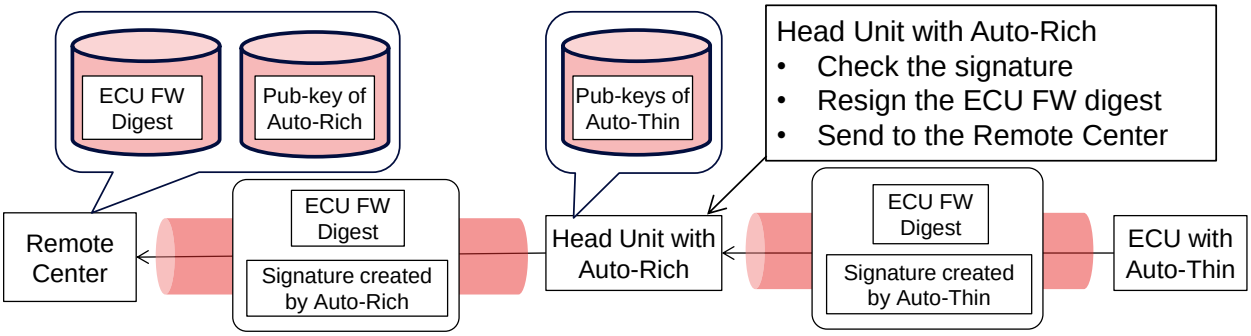

Figure 6: Head Unit that checks ECU signatures (details)

As shown in Figure 6, 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 ECU's Automotive-Thin TPM PCR0 value is stored in an NV index in the Head Unit or Gateway's own Automotive-Rich TPM. A practical example of this procedure is described below (each action is followed by an arrow pointing to the TPM commands used by the Automotive-Thin TPM in that action).

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 that is 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 generate the ECU's firmware digest. A Root-of-Trust-for-Measurement (RTM) in each ECU measures the ECU's firmware and records the measurement in the Thin-TPM at the first boot or after the firmware update is complete. The remote center checks the measurements.

⇒ TPM2_HashSequenceStart & TPM2_SequenceUpdate & TPM2_EventSequenceComplete

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 signs 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. Head Unit or Gateway can also use its Automotive-Rich TPM to verify the digest and signature from each ECU's Automotive-Thin TPM if desired.

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 it 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 original Remote Center signature on the firmware patch with its Automotive-Rich TPM, based on factory provisioning of public keys and certificates for each ECU. Then Head Unit sends the firmware patch to ECU for recommended signature verification with its own Automotive-Thin TPM.

⇒TPM2_LoadExternal & TPM2_VerifySignature

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. This operation will extend the new ECU measurement into the PCR in the Head Unit that records all firmware measurements of all ECUs in the vehicle. The Remote Center expects then to receive a digest from that PCR.

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

4-1. (Same method shown above in 1- Provisioning steps during vehicle manufacturing) At ownership change time, each ECU uses its Automotive-Thin TPM to generate a child key of its EK that is 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 signing 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.

⇒ TPM2_EvictControl and TPM2_FlushContext

4.7 Message flows where Head Unit does not check ECU signatures

Figure 7 shows an example of message flows where the Head Unit or Gateway with the Automotive-Rich TPM instead does not check the signatures sent by each ECU from its own Automotive-Thin TPM. The Head Unit or Gateway creates a TNC [9] 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 cost of the Head Unit and its Automotive-Rich TPM can be minimized
- Cons: The Remote Center must do more signature validation

Note: The network connections and operation requests flow from right to left in Figure 7.

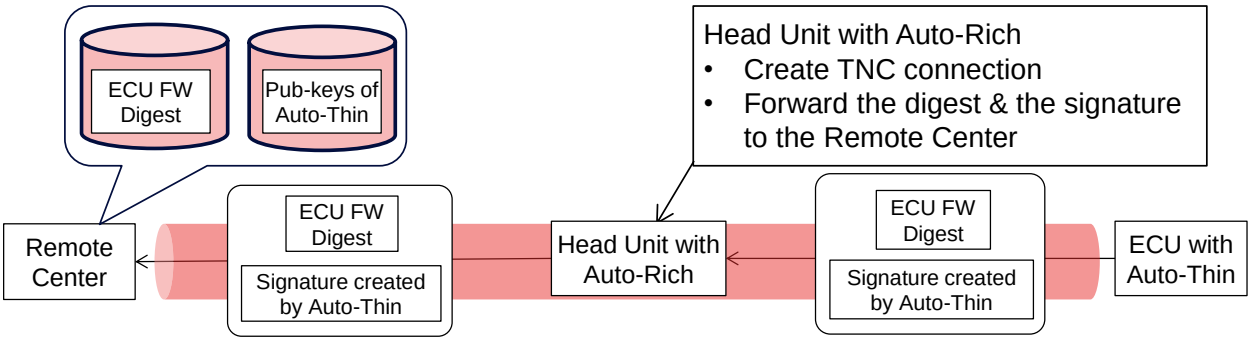

Figure 7: Head Unit that does not check ECU signatures

4.8 Message flows for Remote Maintenance with only Automotive-Thin TPMs

Figure 8 and Figure 9 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.

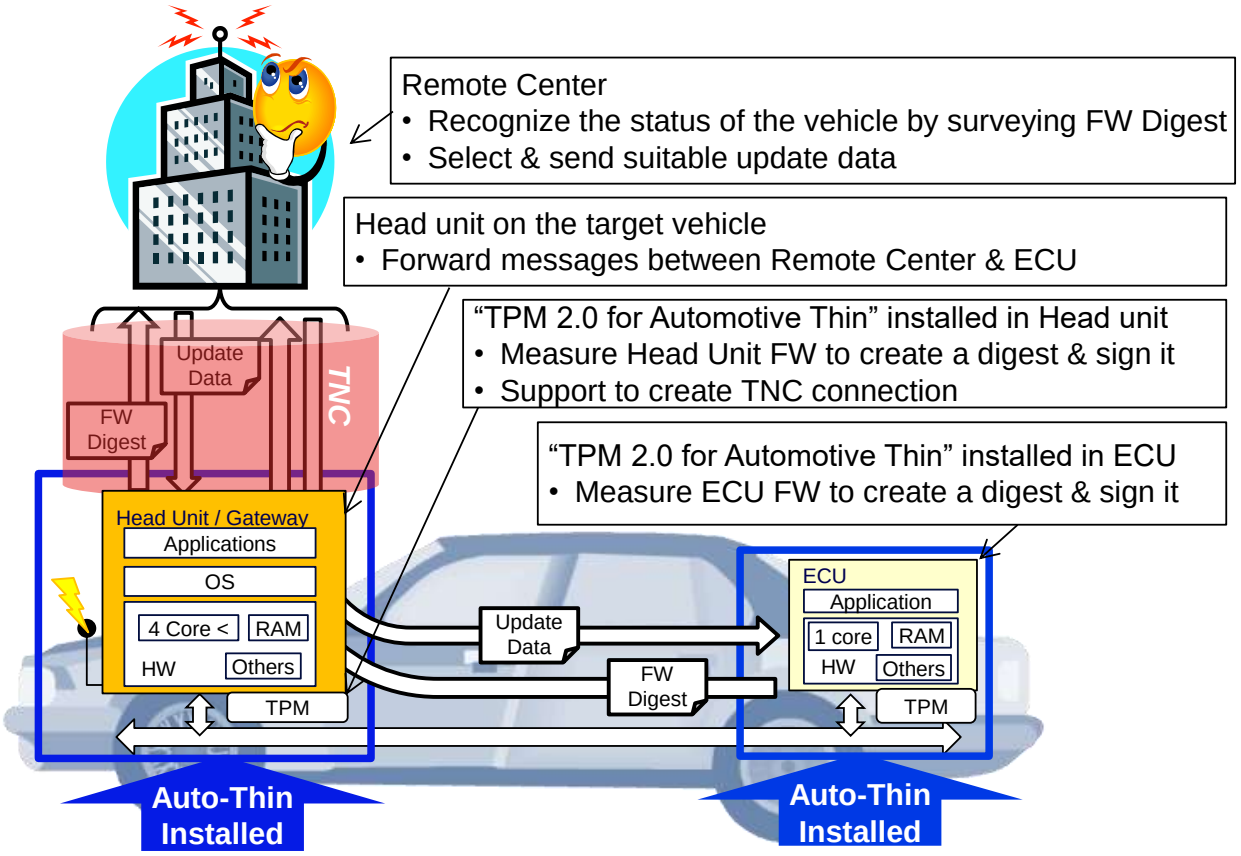

Figure 8: Message Flow for Remote Maintenance with only Automotive-Thin TPMs (summary)

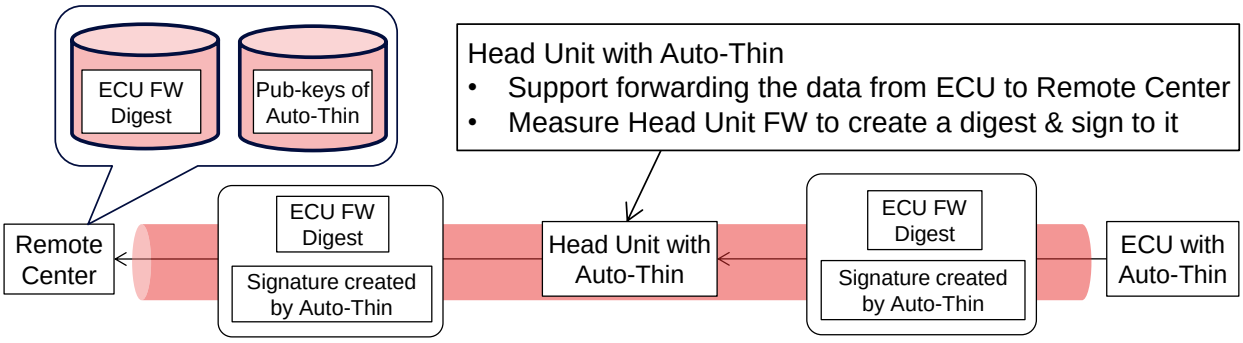

Figure 9: Message Flow for Remote Maintenance with only Automotive-Thin TPMs (details)

The potential role of an Automotive-Thin TPM in a Head Unit is shown in Figure 8 and 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.

After applying updates or patches, the ECUs use the method described in action 3-4 of Section 5.5 to attest their new configuration to the Remote Maintenance Center. If the RMC finds that the ECUs have applied unsuitable updates or patches, the Remote Maintenance Center should send remedial updates or patches to the ECUs.

4.9 Audit and accountability

Traceability and transparency are important not only for the design, development, and testing processes. They are also essential for the deployment process. Actions taken which have a definite impact on safety and security of Automated Driving Systems must be tracked in a reliable, auditable manner so that root cause analysis can be performed to find and fix any safety failures. Only in this manner can accountability be established.

When software and firmware updates are employed to fix automotive safety problems, the importance of reliable, auditable safety and security measures increases yet again. With such measures, automakers can reliably and demonstrably show which updates have been deployed to which vehicles. In this manner, the safety of these vehicles can be established. Recall compliance rates can also be easily calculated and shown for Over-The-Air updates.

TCG supports "audit and accountability" from vehicles to third parties based on a total ecosystem including a chip as Hardware Root of Trust (HRoT=TPM), a security network attestation protocol (TNC-Trusted Network Communications), and central key management (PKI-Public Key Infrastructure) with Remote Center.

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 01.38* 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_PT_PS_FAMILY_INDICATOR	0x00000009	TPM_PS_EMBEDDED, as defined in TPM 2.0 Library Part 2: Structures, Table 26
TPM_PT_PS_LEVEL	0x00000000	The level of the TPM 2.0 Automotive-Thin Specification
TPM_PT_PS_REVISION	101	The revision of the TPM 2.0 Automotive-Thin Specification
TPM_PT_PS_DAY_OF_YEAR	192	The day of the year of the TPM 2.0 Automotive-Thin Profile
TPM_PT_PS_YEAR	2017	The year of the TPM 2.0 Automotive-Thin Profile

5.3 Mandatory Algorithms and Curves

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

- At least one of RSA 2048 bits or ECC 256 bits (TPM_ECC_NIST_P256, TPM_ECC_BN_P256). Additional asymmetric algorithms and key sizes are allowed
- AES 128, other symmetric algorithms are optional
- 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 profile SHALL support RSA key sizes of 2048 bits and key constants specified in Table 2.

Table 2 – Conditionally Mandatory RSA Algorithm Constants

Name	Value	Comments
RSA_KEY_SIZES_BITS	{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 support TPM_ECC_NIST_P256 and TPM_ECC_BN_P256 ECC curves and the ECC constants described in tables 7 and 10 of the TCG Algorithm Registry [8]. Other curves may be implemented.

5.6 Supported 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 3. Commands listed as recommended (R) SHOULD be implemented. All other TPM 2.0 commands (O) are optional.

Commands that enable the Remote Center to verify the state of a vehicle are Mandatory. Commands that enable an ECU to verify firmware supplied by the Remote Center are Recommended. Commands that enable compliance testing and security evaluation are Mandatory.

Table 3 – Mandatory and Recommended TPM 2.0 Commands

Commands	M/R/O	Comments
Signals / Indications		
_TPM_INIT	M	Necessary for TPM initialization
_TPM_Hash_Start	O	
_TPM_Hash_Data	O	
_TPM_Hash_End	O	
Startup		
TPM2_Startup	M	Necessary for TPM initialization
TPM2_Shutdown	M	Necessary for deterministic TPM power-down
Testing		
TPM2_IncrementalSelfTest	O	
TPM2_SelfTest	M	Necessary for proper TPM management
TPM2_GetTestResult	M	Necessary for proper TPM management
Session Commands		
TPM2_StartAuthSession	M	Necessary to protect exchanges between ECU and TPM with

Commands	M/R/O	Comments
		HMAC authentication.
TPM2_PolicyRestart	O	
Object Commands		
TPM2_Create	M	Necessary for Automotive-Thin TPM use case
TPM2_Load	M	Necessary for Automotive-Thin TPM use case
TPM2_LoadExternal	R	Recommended for Automotive-Thin TPM use case
TPM2_ReadPublic	M	Necessary for Automotive-Thin TPM use case
TPM2_ActivateCredential	O	
TPM2_MakeCredential	O	
TPM2_Unseal	R	Recommended for Automotive-Thin TPM use case
TPM2_ObjectChangeAuth	O	
TPM2_CreateLoaded	O	
Duplicate Commands		
TPM2_Duplicate	O	
TPM2_Rewrap	O	
TPM2_Import	M	Necessary for TPM compliance test suite
Asymmetric Primitives		
TPM2_RSA_Encrypt	O	
TPM2_RSA_Decrypt	O	
TPM2_ECDH_KeyGen	O	
TPM2_ECDH_ZGen	O	
TPM2_ECC_Parameters	O	
TPM2_ZGen_2Phase	O	
Symmetric Primitives		
TPM2_EncryptDecrypt	O	Replaced by TPM2_EncryptDecrypt2
TPM2_EncryptDecrypt2	R	Used to perform symmetric encryption or decryption of a single data buffer, with chaining support for bulk symmetric encryption/decryption.
TPM2_Hash	M	Necessary for Automotive-Thin use cases based on hash

Commands	M/R/O	Comments
TPM2_HMAC	R	Recommended for Automotive-Thin TPM use case
Random Number Generator		
TPM2_GetRandom	O	
TPM2_StirRandom	O	
Hash/HMAC/Event Sequences		
TPM2_HMAC_Start	R	Recommended for Automotive-Thin TPM use case
TPM2_HashSequenceStart	M	Necessary for Automotive-Thin TPM use case
TPM2_SequenceUpdate	M	Necessary for Automotive-Thin TPM use case
TPM2_SequenceComplete	R	Recommended for Automotive-Thin TPM use case
TPM2_EventSequenceComplete	M	Necessary for Automotive-Thin TPM use case
Attestation Commands		
TPM2_Certify	M	Necessary for Automotive-Thin TPM use case
TPM2_CertifyCreation	O	
TPM2_Quote	M	Necessary for Automotive-Thin TPM use case
TPM2_GetSessionAuditDigest	O	
TPM2_GetCommandAuditDigest	O	
TPM2_GetTime	O	
Anonymous Attestation		
TPM2_Commit	O	
TPM2_ECC_Ephemeral	O	
Signature Verification		
TPM2_VerifySignature	R	Recommended for Automotive-Thin TPM use case
TPM2_Sign	R	Recommended for Automotive-Thin TPM use case
Command Audit		
TPM2_SetCommandCodeAuditStatus	O	
Integrity Collection (PCR)		
TPM2_PCR_Extend	M	Necessary for Automotive-Thin TPM use case
TPM2_PCR_Event	M	Necessary for Automotive-Thin TPM use case (for use with TPM2_EventSequenceComplete to extend the digest list)

Commands	M/R/O	Comments
TPM2_PCR_Read	M	Necessary for Automotive-Thin TPM use case
TPM2_PCR_Allocate	O	
TPM2_PCR_SetAuthPolicy	O	
TPM2_PCR_SetAuthValue	O	
TPM2_PCR_Reset	O	No need as PCR0 is reset when the vehicle switches on
Enhanced Authorization (EA)		
TPM2_PolicySigned	O	
TPM2_PolicySecret	O	
TPM2_PolicyTicket	O	
TPM2_PolicyOR	O	
TPM2_PolicyPCR	R	To be consistent with recommended support of TPM2_Unseal
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_PolicyTemplate	O	
TPM2_PolicyAuthorizeNV	O	
Hierarchy Commands		
TPM2_CreatePrimary	M	Necessary for Automotive-Thin TPM use case

Commands	M/R/O	Comments
TPM2_HierarchyControl	O	
TPM2_SetPrimaryPolicy	O	
TPM2_ChangePPS	O	
TPM2_ChangeEPS	O	
TPM2_Clear	O	
TPM2_ClearControl	O	
TPM2_HierarchyChangeAuth	M	Necessary for TPM security evaluation
Dictionary Attack Functions		
TPM2_DictionaryAttackLockReset	O	
TPM2_DictionaryAttackParameters	O	
Miscellaneous Management Functions		
TPM2_PP_Commands	O	
TPM2_SetAlgorithmSet	O	
Field Upgrade		
TPM2_FieldUpgradeStart	R	Both of these commands are required if either is implemented
TPM2_FieldUpgradeData		
TPM2_FirmwareRead	O	
Context Management		
TPM2_ContextSave	O	
TPM2_ContextLoad	O	
TPM2_FlushContext	M	Necessary for Automotive-Thin TPM use case and TPM compliance test suite
TPM2_EvictControl	M	Necessary for Automotive-Thin TPM use case
Clocks and Timers		
TPM2_ReadClock	O	
TPM2_ClockSet	O	
TPM2_ClockRateAdjust	O	
Capability Commands		
TPM2_GetCapability	M	Necessary for proper TPM management

Commands	M/R/O	Comments
TPM2_TestParms	O	
Non-volatile Storage		
TPM2_NV_DefineSpace	R	To be consistent with support of TPM2_NV_Increment and TPM2_NV_Extend and table 5
TPM2_NV_UndefineSpace	R	Necessary to free NV storage if TPM2_NV_DefineSpace was used
TPM2_NV_UndefineSpaceSpecial	O	
TPM2_NV_ReadPublic	M	Necessary for TPM compliance test suite
TPM2_NV_Write	R	Necessary to write NV indices, which are recommended functionality
TPM2_NV_Increment	R	Necessary to increment NV counters, which are recommended functionality
TPM2_NV_Extend	R	Necessary to extend NV PCRs, which are recommended functionality
TPM2_NV_SetBits	O	
TPM2_NV_WriteLock	O	
TPM2_NV_GlobalWriteLock	O	
TPM2_NV_Read	M	Necessary to read the EK certificate, which is part of the Thin-TPM use case, and necessary to read NV counters and NV PCRs, which are recommended functionality
TPM2_NV_ReadLock	O	
TPM2_NV_ChangeAuth	O	
TPM2_NV_Certify	O	
Vendor Specific		
TPM2_Vendor_TCG_Test	O	Mandatory if Vendor Proprietary commands that can be tested are implemented

5.7 Mandatory PCR Support

A TPM 2.0 implementation compliant with this Automotive-Thin Profile SHALL support SHA-256 PCR0 in a single bank.

Support for any other PCR is optional.

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 minimum size support

A TPM 2.0 implementation compliant with this Automotive-Thin Profile SHOULD support the non-volatile (NV) storage recommendations in Table 4.

The following table includes minimum NV storage recommendations:

Table 4 – Recommended NV Storage minimum size support

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	3	Corresponds to TPM_PT_HR_PERSISTENT_MIN. Based on [7], this is the minimum number of persistent objects that can be held in TPM NV Memory. When calculating this number, the following example allocations were used: • 1 slot intended for root keys (EK) • 2 slots intended for OS/application usage

Table 5 indicates reserved indices. The ranges are reserved by the TCG Technical Committee (TC) in the TCG Registry of Reserved TPM 2.0 Handles and Localities [11].

There may be one or more EK certificate and EK template defined based on algorithms and support by TPM vendors.

5.10 NV Storage handles (informative)

Table 5 contains the list of handles recommended for use by the Automotive Industry in an Automotive Thin TPM.

Table 5 – NV Index handles example

Reserved Indices	Value
EK Certificate (RSA)	0x01C00022
EK template (RSA)	0x01C00024
EK Certificate (ECC)	0x01C0002A
EK template (ECC)	0x01C0002C

NOTE For an example of the use of EK certificates, please refer to the section “Message Flows where Head Unit checks ECU signatures”.

5.11 Mandatory Reserved Handles

TPM 2.0 implementations of the Automotive-Thin Profile SHALL reserve specific handles for keys such as an EK. These are used to point to “Endorsement Keys” that can be used for platform authentication and identification.

5.12 Mandatory Default Template for EK

The Automotive-Thin TPM profile doesn’t require support for policies, only HMAC and/or password authorization. Therefore neither the ordinary EK template nor the ordinary EK handles can be used. See TCG EK Credential Profile specification [10]. New RSA templates are defined in Table 6. New ECC templates are defined in Table 7. New NV handles are defined in Table 5.

5.12.1.1 RSA Template

Table 6 – Automotive-Thin default RSA EK Public Area Template (TPMT_PUBLIC)

Parameter	Type	Content
type	TPMI_ALG_PUBLIC	TPM_ALG_RSA
nameAlg	TPMI_ALG_HASH	TPM_ALG_SHA256
objectAttributes	TPMA_OBJECT	fixedTPM = 1 stClear = 0 fixedParent = 1 sensitiveDataOrigin = 1 userWithAuth = 1 adminWithPolicy = 0 noDA = 1 encryptedDuplication = 0 restricted = 1 decrypt = 1 sign = 0
authPolicy	TPM2B_DIGEST	
size	UINT16	32
buffer	BYTE	All 0
parameters	TPMS_RSA_PARMS	
symmetric->algorithm	TPMI_ALG_SYM_OBJECT	TPM_ALG_AES
symmetric->keyBits	TPMI_AES_KEY_BITS	128
symmetric->mode	TPMI_SYM_MODE	TPM_ALG_CFB
symmetric->details		NULL
scheme->scheme	TPMI_ALG_ASYM_SCHEME	TPM_ALG_NULL
scheme->details		NULL
keyBits	TPMI_RSA_KEY_BITS	2048
exponent	UINT32	0
unique	TPM2B_PUBLIC_KEY_RSA	
size	UINT16	256
buffer	BYTE	All 0

5.12.1.2 ECC Template

Table 7 – Automotive-Thin default ECC EK Public Area Template (TPMT_PUBLIC)

Parameter	Type	Content
type	TPMI_ALG_PUBLIC	TPM_ALG_ECC
nameAlg	TPMI_ALG_HASH	TPM_ALG_SHA256
objectAttributes	TPMA_OBJECT	fixedTPM = 1 stClear = 0 fixedParent = 1 sensitiveDataOrigin = 1 userWithAuth = 1 adminWithPolicy = 0 noDA = 1 encryptedDuplication = 0 restricted = 1 decrypt = 1 sign = 0
authPolicy	TPM2B_DIGEST	
size	UINT16	32
buffer	BYTE	All 0
parameters	TPMS_ECC_PARMS	
symmetric->algorithm	TPMI_ALG_SYM_OBJECT	TPM_ALG_AES
symmetric->keyBits	TPMI_AES_KEY_BITS	128
symmetric->mode	TPMI_SYM_MODE	TPM_ALG_CFB
symmetric->details		NULL
scheme->scheme	TPMI_ALG_ECC_SCHEME	TPM_ALG_NULL
scheme->details		NULL
curveID	TPMI_ECC_CURVE	TPM_ECC_NIST_P256
kdf->scheme	TPMI_ALG_KDF	TPM_ALG_NULL
kdf->details		NULL
unique	TPMS_ECC_POINT	
x->size	UINT16	32
x->buffer	BYTE	All 0
y->size	UINT16	32
y->buffer	BYTE	All 0

5.13 Mandatory Resource Minimums and Maximums

TPM 2.0 implementations of the Automotive-Thin Profile SHALL satisfy the constraints in Table 8 for minimum and maximum resources supported.

Table 8 – 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 [7] requirements
Concurrent loaded sessions	MUST be at least 3	None specified	Recommend implementations be consistent with PC-Client specification [7] requirements
Concurrent loaded objects	MUST be at least 2	None specified	Recommend implementations be consistent with PC-Client specification [7] requirements

5.14 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.

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, *TCG Algorithm Registry*, current specification level, http://www.trustedcomputinggroup.org/resources/tcg_algorithm_registry

[9] 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

[10] Trusted Computing Group, TCG EK Credential Profile for TPM Family 2.0, current TPM 2.0 specification level

[11] Registry of Reserved TPM 2.0 Handles and Localities, http://www.trustedcomputinggroup.org/resources/registry-reserved-tpm-2-0-handles-localities

[12] Terminology for Constrained-Node Networks, https://tools.ietf.org/html/rfc7228

