R
E
F
E
R
E
N
C
E

TCG Guidance for Securing Resource-Constrained Devices

Version 1.0
Revision 22
March 13, 2017

Contact: admin@trustedcomputinggroup.org

TCG

TCG Published

Disclaimers, Notices, and License Terms

Acknowledgements

The TCG wishes to thank all those who contributed to this reference document. This document builds on considerable work done in the various work groups in the TCG.

Special thanks to the members of the IoT-SG who participated in the development of this document:

Graeme Proudler (Editor)	Independent
Ira McDonald (Editor)	High North
Steve Luther	United States Government

Additional thanks to those who provided comments on this document during review:

Steve Hanna (Infineon), Sung Lee (Intel), Alan Tatourian (Intel)

Table of Contents

List of Tables

1. Scope and Audience

1.1 Scope

This reference document provides implementation guidance for trusted platforms built with resource-constrained devices.

This reference document is not a TCG Specification and therefore is not normative.

1.2 Audience

The intended audience for this reference document is designers, developers and manufacturers of resource-constrained devices, software, and services. This reference document is intended to assist in the determination of whether an embedded device could be a trusted platform and (if so) what resources the device will need to be a trusted platform. Typically those resources are those found in Trusted Platform Modules (TPMs).

1.3 References

The date upon which a URL was last verified by TCG is the date inside the brackets following the URL. For example, a URL verified during November 2016 is followed by [November 2016].

[1] NIST, Special Publication (SP) 800-90A Revision 1, Recommendation for Random Number Generation Using Deterministic Random Bit Generators, http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf [November 2016]

[2] NIST, Randomness Beacon, www.nist.gov/itl/csd/ct/nist_beacon.cfm [November 2016]

[3] Trusted Computing Group, Trusted Platform Module Library 2.0, listed at www.trustedcomputinggroup.org/tpm-library-specification/ [November 2016]

[4] Trusted Computing Group, TPM 2.0 Mobile Reference Architecture Family, listed at www.trustedcomputinggroup.org/work-groups/mobile/ [November 2016]

[5] Trusted Computing Group, Multiple Stakeholder Model, listed at www.trustedcomputinggroup.org/work-groups/mobile/ [November 2016]

[6] Trusted Computing Group, TPM Software Stack specifications, listed at www.trustedcomputinggroup.org/work-groups/software-stack/ [November 2016]

[7] Trusted Computing Group, Enterprise and Opal Secure Encrypting Drives, listed at www.trustedcomputinggroup.org/work-groups/storage/ [November 2016]

[8] Trusted Computing Group, TCG Storage Architecture Core Specification, listed at www.trustedcomputinggroup.org/work-groups/storage/ [November 2016]

[9] Trusted Computing Group, "TCG Storage Interface Interactions Specification (SIIS)", listed at www.trustedcomputinggroup.org/work-groups/storage/ [November 2016]

[10] Storage Security Industry Forum (SSIF), Guide to Data-At-Rest Solutions, listed at www.trustedcomputinggroup.org/solutions-guide-data-rest/ [November 2016]

[11] Trusted Computing Group , "TCG EK Credential Profile for TPM Family 2.0", listed at www.trustedcomputinggroup.org/work-groups/infrastructure/ [November 2016]

[12] Trusted Computing Group, "TCG Credential Profiles For TPM Family 1.2" listed at www.trustedcomputinggroup.org/infrastructure-work-group-tcg-credential-profiles-specification/ [November 2016]

[13] Trusted Computing Group "TPM Keys for Platform Identity for TPM 1.2", listed at www.trustedcomputinggroup.org/work-groups/infrastructure/ [November 2016]

[14] Trusted Computing Group, "A CMC Profile for AIK Certificate Enrollment", listed at www.trustedcomputinggroup.org/work-groups/infrastructure/ [November 2016]

[15] Trusted Computing Group, Trusted Network Communication, www.trustedcomputinggroup.org/work-groups/trusted-network-communications/ points to "TNC Resources" at www.trustedcomputinggroup.org/work-groups/trusted-network-communications/tnc-resources/ [November 2016]

[16] Open Source Trusted Network Communications software, "StrongSwan", https://strongswan.org/ [November 2016]

[17] Trusted Computing Group, "TPM 1.2 Protection Profile", listed at www.trustedcomputinggroup.org/work-groups/pc-client/ [November 2016]

[18] Trusted Computing Group, "TCG EFI Protocol Specification", listed at www.trustedcomputinggroup.org/work-groups/pc-client/ [November 2016]

[19] Trusted Computing Group, IF-MAP Metadata for ICS Security, listed at www.trustedcomputinggroup.org/work-groups/trusted-network-communications/tnc-resources/ [November 2016]

[20] International Society of Automation, ISA-100, https://www.isa.org/isa100/ [November 2016]

[21] AllSeen Alliance, The AllJoyn Framework, https://allseenalliance.org/framework [November 2016]

[22] Intel, TPM Software Stack software, https://github.com/01org/TPM2.0-TSS , [November 2016]

[23] IBM, TPM Software Stack software, https://sourceforge.net/projects/ibmtpm20tss , [November 2016]

[24] Microsoft, TPM Software Stack software, https://github.com/Microsoft/TSS.MSR [November 2016]

2. Preface

This reference document provides (section 3) guidance for countering threats using trusted platform services, (section 4) guidance for providing trusted platform services, and indicates (section 5) how to calculate the code sizes and working memory resources needed to implement trusted platform services and use cases.

An ideal trusted platform has a stack of security services, each layer either relying upon services or protections provided by previous layers, or enhancing the services provided by previous layers. The bottom-most service (that underpins all security and privacy) is the isolation of processes. Arguably the next-most critical layer is a service that provides random numbers. Then most platforms have a service that protects secrets, and services that use secrets for identity and confidentiality. More advanced services release secrets to specific processes, enable reasoning about the trustworthiness of a device, and enable privacy. This stack of trusted platform security services supports operating systems and applications, which can continue to use conventional security protocols (to communicate over the Internet, for example).

3. Implementation Guidance for Countering Threats

This section briefly describes the use of Trusted Platforms to address the threats indicated in the titles.

3.1 Tampering with Hardware

The amount of hardware protection (especially tamper resistance) required by a device depends on the degree of access by a rogue to the device, the effect of loss of access to the information the device provides, and the effect of misinformation. For example, if a device's information is low value or low importance, the device probably needs little hardware protection. If a device is in a secure environment, it probably needs little hardware protection. On the other hand, a device in an insecure environment might benefit from a limited amount of protection if the device cannot easily be removed for detailed inspection, or might need a sophisticated level of protection if the device contains valuable data and can be removed to an environment with extensive inspection facilities.

This document does not include a substantive description of methods for the protection of devices from hardware tampering. The intricacies of hardware protection mechanisms are rarely revealed because that would assist attackers. For the same reason, manufacturers may advertise just the well-known threats that are addressed by their products, not all threats known to the manufacturer.

The TCG has published a rigorous Common Criteria Protection Profile [17] for TPMs. However, the TCG doesn't currently provide guidance on methods for the hardware protection of devices, or hardware protection of computation within devices.

Some products include tamper resistant computing environments. Many Hardware Security Modules provide a sophisticated hardware-protected computing environment. Other devices such as ordinary Personal Computers don't provide hardware protection when sensitive data is processed, but may have hardware-protected TPMs that protect small amounts of data-at-rest. Some secure microprocessors have hardware-protected processing environments. A TCG-certified TPM is known to provide a good level of protection from hardware tampering.

3.2 Subversion of Algorithms

Replay attacks on protocols are hindered if nonces (numbers that are used only once) are included in protocols. Brute-force attacks on algorithms that use nonces and cryptographic keys are hindered if nonces and keys are long random numbers.

Nonces tend to be used in large quantities and hence almost certainly require a device to use a random number generator. Cryptographic keys may be provisioned during device manufacture but generating keys after deployment requires a device to use a random number generator.

Random numbers may be generated by initializing a state machine with high entropy data and using a hash algorithm to whiten the output of the state machine.

Devices may be provisioned with high entropy data during manufacture. The device itself may obtain limited amounts of additional high entropy data by under-sampling a signal obtained by measuring the device's environment or the actions of a human user. The device itself may obtain high entropy data from a reliable external source, albeit this requires a communication channel with confidentiality and integrity. Preferably the device itself obtains additional high entropy data by measuring random physical processes within the device.

One instance of entropy data cannot initialize more than one individual instance of a state machine (because the act of initializing an individual state machine consumes all the entropy). In other words, different individual devices must be initialized with different entropy data. Once a state machine has been initialized with entropy data, neither the entropy data nor the state machine's state must be revealed (because that could enable prediction of the random numbers produced by the state machine). The state machine must not be reset (because that would discard any entropy that was provided and could cause predictable random numbers).

Devices should derive nonces and cryptographic keys from a random number generator.

Devices should contain a generator that derives random numbers from high entropy data.

The random number generator in each device should be initialized with a fresh instance of high entropy data.

Devices should contain a source of high entropy data.

The NIST define random number generators in Special Publication SP800-90A "Recommendation for Random Number Generation Using Deterministic Random Bit Generators" [1]. The TCG defines a random number generator for TPMs in the "Random Number Generator (RNG) Module" section of Part-1 of the TPM2.0 specification [3]. A true hardware random number generator is an ideal way of generating high quality random numbers. Some microprocessors have internal random number generators. A TCG-certified TPM is known to output high quality random numbers via the command TPM2_GetRandom(). The NIST's Randomness Beacon [2] is a source of good quality random data.

3.3 Access to Concealed Data

Preventing information discovery and information tampering requires isolation of the data representing the information, isolation of the engine processing the data, and authorization controls that are enforced by the engine when data is accessed via the engine.

Devices should isolate secret data.

Devices should isolate the engines that process isolated data.

Devices may isolate data sent to or from engines that process isolated data, depending upon the data and the device's environment.

Device isolation mechanisms may be physical or logical, albeit the isolation of a mechanism providing logical isolation ultimately depends on physical isolation. Simple physical isolation of data is simpler and more nuanced than cryptographic isolation of data, but more expensive. The TCG's documents "Multiple Stakeholder Model" [5] and "TPM Mobile Reference Architecture" [4] discuss some techniques for the isolation of engines.

Generic communication security mechanisms can be used to isolate data when data is sent to or from engines that process isolated data. Generic communication security mechanisms for communication confidentiality and integrity are common knowledge, and are not discussed here apart from the communication of passwords, which is discussed in section 3.3.3 "Access Controls" and in section 4.4.4 "Authorization Methods".

3.3.1 Physical Isolation of data

Data can be isolated using a device comprising memory (with a storage capacity as large as the size of data) and a processing engine (that controls access to the stored data).

The memory simply stores plain-text data and the processing engine implements an interface that prevents arbitrary access to the plain-text data. The engine prevents arbitrary inspection of stored data and prevents tampering with the stored data. The combination of memory and engine ensures data persistence, data confidentiality, data integrity, and guarantees erasure when unique data is deleted.

Devices should be capable of storing at least small amounts of plain-text secret data and should implement an interface that prevents arbitrary access to that plain-text secret data.

This type of device could comprise semiconductor memory with an interface controlled by processor, or might be a spinning magnetic platter with an interface controlled by a processor (a conventional Hard Disk Drive, in other words).

A chip TPM's NV Storage usually comprises semiconductor memory with an interface controlled by a processing engine. It can store a limited amount of data. Space for data is allocated using the command TPM2_NV_DefineSpace(). Ordinary data is written into an allocated space via the command TPM2_NV_Write(), and data is read from that space via the command TPM2_NV_Read(). Other NV Storage commands are intended to enable an operating system to use NV Storage for monotonic counters, sticky-bit fields, and hashing registers.

Enterprise and Opal Secure Encrypting Drives (SEDs) [7] are mass-storage devices that automatically encrypt data written to storage and automatically decrypt data read from storage. SEDs are capable of storing large amounts of data and are accessed via ordinary read/write/modify commands.

3.3.2 Cryptographic Isolation of Data

Data can be isolated using a device comprising memory (with a storage capacity smaller than the size of data) and a processing engine (that controls access to the small memory), plus additional memory with a capacity larger than the size of data.

The device stores encrypted integrity-protected data in the additional memory. The device prevents the arbitrary inspection of its internal plain-text data and prevents tampering with its plain-text data. The combination of the device and additional memory ensures data confidentiality and data integrity, but does not guarantee data persistence in the additional memory or erasure of data from the additional memory. Even so, data in additional memory can reliably "be put beyond use" by erasing those cryptographic keys in the device that are necessary to decrypt the data in the additional memory.

Devices that provide cryptographic isolation of data should:

- be capable of storing small amounts of plain-text secret data;
- implement an interface that prevents arbitrary access to small amounts of plain-text secret data; and
- implement an interface to store an encrypted integrity-protected version of plain-text secret data in unprotected memory, and to retrieve that encrypted integrity-protected data from the unprotected memory.

This type of device could comprise semiconductor memory with an interface controlled by processor, plus additional memory of any sort.

The chip version of a TPM's Protected Storage Hierarchy usually comprises semiconductor memory with an interface controlled by a processing engine. It can store an unrestricted amount of data in additional memory but requires a non-trivial amount of management. Management software must create the root of an encrypted integrity-protected hierarchy in the TPM via the command TPM2_CreatePrimary() and then either create (via TPM2_Create()) or import (via TPM2_Import()) a tree of cryptographic decryption keys that is wide enough to accommodate all users and deep enough to provide the required control resolution. Only then can user data (passwords and keys) be attached to the hierarchy, using TPM2_Create() or TPM2_Import(). Keys and user data are retrieved from the encrypted integrity-protected hierarchy via the command TPM2_Load(). Once loaded, keys and user data can be used in signing commands such as TPM2_Sign() or returned to the caller via the command TPM2_Unseal(). Once loaded, keys and user data can be duplicated to other TPMs or to arbitrary software via the command TPM2_Duplicate().

3.3.3 Access controls

Isolated data is useless unless it can be accessed. Therefore devices should provide an interface for callers to prove they have sufficient privilege to use or read isolated data.

The best method of proving sufficient privilege depends on device architecture and network architecture. If nothing can observe or tamper with the path between a caller and the engine controlling access to isolated data, a simple password (passed as plain text) is sufficient. Otherwise it is prudent to send nonces along with data, sign the combination of nonce and data with a secret, and pass the HMAC signature but not the secret. If a caller cannot be online, it may be necessary to use asymmetric digital signatures.

TPMs provide a plethora of access control mechanisms including passwords, HMAC, asymmetric digital signatures, hardware signals (*locality*) that indicate a level of privilege in a software stack, a logical or hardware signal that indicates the physical presence of a person, measurements (in *Platform Configuration Registers*) of the software currently executing on a device, Boolean comparison with isolated data, and combinations of these mechanisms. Authorization sessions are described in TPM2 specification [3] Part-1 section "Authorizations and Acknowledgments". All types of authorization session are started with the TPM command TPM2_StartAuthSession(). Temporary session secrets can be created from a secret value (a *salt*) already loaded into the TPM or by using the authorization of a key or data already loaded into the TPM.

TCG's "TCG Storage Architecture Core Specification" [8] describes the intended security architecture of an SED. TCG's "Storage Interface Interactions Specification" [9] describes how to manage the security properties of SEDs.

3.4 Device Impersonation

The behavior of a device is unpredictable unless the device can be identified. Remote identification of a device requires devices to use secrets to uniquely distinguish between devices. Hence a device's identification secret should be concealed from any entity that might pretend to be the device. This normally requires a device's secret to be concealed both when it is stored and when it is used.

Secrets inside a component fixed to a device can be used as that device's secrets.

Often the most difficult aspect of device identification is the initialization of an identification secret. Once one secret has been initialized, that secret can be used to initialize another secret. The initialization of all identification secrets should be done in isolated environments that vouch for the properties of the device containing the secret.

Often using an identification secret is the easiest aspect of device identification. The type of identification secret that is used depends on the trustworthiness of the channel over which the device connects and the trustworthiness of the destination to which the device connects.

Table 1: Attributes of Identification Secrets

Type of Secret	**Channel**	**Destination**	**Channel Data**	**Identification Complexity**
Plain-text password	trusted	trusted	Data accompanied by password	low
Symmetric key	untrusted	trusted	Data (HMAC) signed by symmetric key	medium
Asymmetric key	untrusted	untrusted	Data signed by asymmetric private key	high

Privacy during identification is impossible if the device must be unambiguously identified. Privacy during recognition is possible if different identification secrets are used for different destinations, or if the same identification secrets are used in anonymous or pseudonymous signing schemes.

TCG-certified TPMs are known to be suitable for storing device identification secrets for a device. TPMs are typically initialized with a secret called an Endorsement Key and a certificate that says (words to the effect that) “the device containing this Endorsement Key is a genuine TPM”. Once initialized, TPMs can initialize other secrets by (1) importing secrets from trusted entities via the command TPM2_Import(), or (2) by creating secrets inside the TPM via the command TPM2_Create() and then obtaining credentials for the new secret from a trusted entity via the command TPM2_ActivateCredential(). The TPM’s authorization mechanisms use passwords, or symmetric secrets, or asymmetric secrets, and enable secrets inside a TPM to be used as proxy secrets for the device containing the TPM. TPMs can perform both ordinary signing schemes and an anonymous or pseudonymous asymmetric signing scheme called Direct Anonymous Attestation.

3.5 Subversion by Malware

Certain types of malware infection can be prevented by the method called “verified boot” or “secure boot”: when a platform boots, the platform compares a measurement of installed software against an expected value; if the measured value is different from the expected value, the platform replaces and reinstalls the software before executing it; if the measured value is the same as the expected value, the platform just executes the installed software.

Trusted platforms provide a more flexible boot strategy: “measured boot” assumes that it doesn’t matter what software executes on a platform as long as software can’t pretend to be other software, and software can't access secrets belonging to other software. Measured boot requires both a Root-of-Trust-for-Measurement and Platform Configuration Registers that are protected from rogues.

The first software to execute on a trusted platform is called a Root-of-Trust-for-Measurement, which must be trustworthy and trusted because its behavior cannot be dynamically verified. An RTM measures the second software (whatever it may be) that will execute on the platform, records the result in a Platform Configuration Register, and executes the second software. Then the second software measures the third software (whatever it may be) that will execute on the platform, records the result in a PCR, and executes the third software. And so on until either a Trusted OS or Trusted Computing Base should have been instantiated, but may not have been.

Complex devices typically cease recording measurements in PCRs at this level in the software stack. The reason is that it is difficult to deduce the trustworthiness of a device after multiple applications have executed, unless a Trusted OS or Trusted Computing Base can isolate applications. If a Trusted OS or Trusted Computing Base has been measured and applications are isolated, it is sufficient for the Trusted OS or Trusted Computing Base itself to provide applications' keys, plus report on the applications that are currently executing.

Typically only simpler devices, such as those with a simple OS and just one application that executes until the device reboots, would record a measurement of that application in a PCR.

Devices should contain a trusted measurement process called a Root-of-Trust-for-Measurement that is the first software to execute after a device is released from reset.

Devices should contain one or more Platform Configuration Registers (PCR) in which an RTM and other measurement agents can record measurements of software before the software is executed.

If the value in a PCR is subsequently signed by a platform's cryptographic identity, the signed PCR value constitutes evidence to a third party of whatever OS or hypervisor exists in the platform. The third party can inspect the signed PCR value and decide whether it indicates that the platform is in a trustworthy state before interacting with the platform.

Devices should contain trusted services that use the values in PCRs as evidence of the software executing on the device.

If the value in a PCR is compared by a TPM with a value stored with a secret, the TPM can ensure that only the intended software has access to that software's secrets. This is a process called "sealing", which is exclusive to trusted platforms: when a secret (a signing key or password) is given to the TPM to be protected by the TPM, the caller can state the PCR values that must exist when the secret is used; if current PCR values do not match the values stored with a secret, the TPM refuses to allow the caller to use the signing key, or refuses to reveal the password to the caller.

Devices should contain trusted services that use the values in PCRs to prevent secrets being used by inappropriate software, or prevent secrets being revealed to inappropriate software.

TPMs provide PCRs and trusted functions that use those PCRs, including:

- TPM2_Extend() and TPM2_Event() that record measurements in PCRs,
- TPM2_PCR_Read() and TPM2_Quote() that report the current value of PCRs,

- TPM2_Create() that associates secrets with PCR values,
- TPM2_Sign() that determines whether secrets can be used when signing data, and
- TPM2_Unseal() that determines whether secrets can be revealed outside the TPM.

4. Implementation Guidance for Trusted Platform Services

4.1 Cryptography

Many devices use cryptography to protect data that persists when the device is switched off. All devices use cryptography to protect communications over shared networks.

If devices use cryptography, devices should use standardized cryptographic algorithms. Private cryptographic algorithms may be safe but (unless one has expert advice) it is safer to use cryptography that has been studied by the cryptographic community.

Devices should use cryptographic algorithms in only the ways those algorithms are designed to be used. It may be tempting to modify cryptographic algorithms or use them in unusual ways, but one might break an assumption that the algorithms depend upon for their security. For example, one should not modify the iterative process in a block encryption algorithm, or use a mask function as an encryption function.

Devices should be cryptographically agile, meaning that devices should have the ability to use different cryptographic algorithms for each task. Without cryptographic agility, a device might be unsuitable for both mass markets and for specialist markets, or a device could be rendered obsolete overnight when a cryptographic algorithm is found to be flawed. Cryptographic agility requires processes to use data structures that name the specific algorithm which will be used with the rest of the data in that structure.

4.2 Isolation

Devices should isolate processes from each other. In particular, if some processes are not intended to access particular sensitive data, devices should isolate the processes that are intended to access those particular sensitive data from processes that have no legitimate right of access.

While isolation will in principle protect any amount of sensitive data, isolation must be physically enforced when a platform is switched off, and isolating hardware may be expensive. In practice, therefore, isolating hardware can store only a bounded amount of sensitive data. The cost of isolating hardware is minimized, and there is still (in principle) no bound on the amount of stored data, if isolating hardware protects just a single encryption key, and that key is used to encrypt other keys and data that are held in non-isolating hardware (non-protecting hardware).

Isolation prevents processes from interfering with each other, or misusing secrets, and is arguably the most substantial and onerous implementation aspect of a trusted embedded platform. Dynamic isolation mechanisms include sand boxes, visualization, and trusted execution environments. The only static isolation mechanism is physical separation. The TCG's documents "Multiple Stakeholder Model" [5] and "TPM Mobile Reference Architecture" [4] discuss isolation techniques, but do not define them.

The functionality of a single function device may be physically isolated from other functions, but processes within that device that are intended to access secrets should still be isolated from processes that are not intended to access those secrets. Unless there is some way of isolating the process that uses a secret from a process that shouldn't use that secret, the device cannot ensure that secrets are properly used. If nothing else, trusted platform primitives and facilities must be isolated from processes that are not trusted platform primitives and facilities. For example, TPMs must be isolated from the rest of a device.

Depending on the degree of security that is provided by a given method of isolation, TPMs may be physically isolated or logically isolated. The TCG's documents "Multiple Stakeholder Model" [5] and "TPM Mobile Reference Architecture" [4] discuss isolation for TPMs in mobile devices. TCG-certified TPMs are known to provide a robust degree of isolation.

4.3 Random Number Generator

If a device generates cryptographic keys or nonces, the device should have a Random Number Generator engine that produces non-deterministic numbers. This is because the security of most cryptographic algorithms is critically dependent upon numbers whose values cannot be predicted, even when other numbers supplied by the same source are known.

The TPM2_GetRandom() command of a TCG-certified TPM is known to provide good quality random numbers.

Methods of generating random numbers are described in publications of standardization organizations, such as the NIST's "Recommendation SP800-90A" [1].

4.4 Protected Storage

Trusted platforms provide three types of services to protect stored data. They differ in the amount of data that can be stored and their ability to prevent or facilitate erasure.

4.4.1 Bounded Storage

The amount of data that can be stored in Protected Bounded Storage is limited by the size of isolated memory in a device, and there may be limits on the size of individual pieces of data.

Protected Bounded Storage uses isolating hardware to guarantee data persistence, confidentiality, and integrity, with guaranteed erasure if the data has not been duplicated elsewhere.

Protected Bounded Storage should comprise isolated semiconductor memory. A Protected Bounded Storage service should ensure data persistence, confidentiality, and integrity, as well as guaranteeing erasure if the data has never been copied.

The TPM's NV (Non Volatile) Storage service stores a limited number of data objects and provides them with access controls. The service ensures persistence, data confidentiality, data integrity, and guarantees erasure when unique data is deleted.

4.4.2 Mass Storage

The amount of data that can be stored in Protected Mass Storage is limited by the size of memory in a mass storage drive.

Protected Mass Storage uses isolated hardware and cryptography to guarantee data persistence, confidentiality, and integrity with guaranteed erasure if the data has not been duplicated elsewhere.

Protected Mass Storage should comprise enhanced Hard Disk Drives, CD drives, etc. connected to a device. A Protected Mass Storage service should ensure data persistence, confidentiality, and integrity, as well as guaranteeing erasure if the data has never been copied.

One example of Protected Mass Storage is a mass-market Secure Encrypting Drive (SED). SEDs automatically encrypt data written to storage and automatically decrypt data read from storage, and enforce access controls over both drive management services and data retention services. The TCG has published SED specifications [7]. The Storage Security Industry Forum has published the white paper "SSIF Guide to Data-At-Rest Solutions" [10].

4.4.3 **Unbounded Storage**

There is no inherent limit on the amount of data that can be stored in Protected Unbounded Storage, although there may be limits on the size of individual pieces of data.

Protected Unbounded Storage uses isolating hardware and cryptography to guarantee data confidentiality and detection of data alteration, but does not guarantee data persistence, and cannot guarantee data erasure.

Protected Unbounded Storage should comprise isolated semiconductor memory for small amounts of keys and sensitive data, plus non-isolated memory for unrestricted amounts of keys and sensitive data. A Protected Unbounded Storage service should ensure data confidentiality and integrity.

4.4.3.1 Protected Storage Hierarchy

If a device stores copies of one or more cryptographic keys or sensitive data objects in a non-isolating environment, devices should provide cryptographic confidentiality and integrity protection for those keys and sensitive data. Encrypting keys should be encrypted by another key and form a branch of a tree of encrypted keys whose root key is permanently plain-text and isolated by hardware from processes that have no legitimate right to access the root key. Devices may store plain-text copies of other encrypted keys and data in isolated hardware, in order to provide faster access to those keys and data.

TPMs provide Storage Hierarchy functionality whose root key is permanently plain-text and isolated from processes that should not access the root key. The TPM's Storage Hierarchy provides confidentiality and integrity protection for encrypted keys and data held outside the TPM in a non-isolating environment. This functionality enables plain-text copies of keys and data to be temporarily loaded within the TPM's isolation boundary, and used. The TPM's Storage Hierarchy also includes means to store a small number of plain text copies of encrypted keys and data within the TPM's isolation boundary, and use them.

4.4.3.2 Multi-tasking

If a device is single tasking but it is preferable that the device appears to be multi-tasking, the device should provide replay protection plus cryptographic confidentiality and integrity protection for sensitive data-in-use stored in a non-isolating environment. The replay protection method should ensure that out-of-date copies of data-in-use are rejected. The cryptographic confidentiality method should ensure that only the device can obtain a plain-text copy of the data-in-use. The cryptographic integrity protection should ensure that only legitimate data-in-use will be interpreted by the device as data-in-use.

TPMs provide Storage Hierarchy functionality that enables plain-text copies of keys and data to be temporarily safely stored outside the TPM's isolation boundary.

4.4.3.3 Duplication of Stored Objects

If it is preferable that a device is able to export sensitive keys and data to other devices, the device should provide cryptographic confidentiality and integrity protection for that sensitive data before it leaves the device's protection.

If it is preferable that a device is able to import sensitive keys and data from other devices, the device should accept only sensitive data that has cryptographic confidentiality and integrity protection.

TPMs provide Storage Hierarchy functionality that enables plain-text copies of keys and data to be encrypted and integrity protected such that the plain-text keys and data can be recovered using a specific encryption key.

4.4.4 Authorization methods

If it is preferable for a device to restrict the usage of keys or data objects, devices should enforce access controls that apply to those keys and data.

4.4.4.1 Password

If a device can prevent a man-in-the-middle from seeing authorization information sent to a data store, the device should allow authorization information to be a plain-text password.

Passwords are useful for commands sent from a device's Trusted Computing Base, because the Trusted Computing Base is presumably able to prevent processes from inspecting data sent to the data store.

TPMs provide Storage Hierarchy functionality that enables plain-text passwords to be used for access control. Passwords are sent as plain-text to the TPM.

4.4.4.2 HMAC

If a device can't prevent a man-in-the-middle from seeing authorization information sent to a data store, the device should allow authorization information comprising HMAC signatures over data attached to nonces sent to the data store and nonces sent from the data store. A plain-text password should be the HMAC signing key.

HMAC signatures are useful for commands sent from remote entities, which must be on-line because each exchange of authorization information signs a new nonce.

TPMs provide Storage Hierarchy functionality that enables plain-text passwords to HMAC-sign requests and responses together with a nonce from the caller and a nonce from the TPM.

4.4.4.3 Enhanced

A device may provide enhanced authorization methods to enable combinations of privileges, delegation of privilege, and restricted privileges.

TPMs provide Storage Hierarchy functionality with Enhanced Authorization (EA). EA allows Boolean combinations of authorization using passwords, HMAC signatures, and asymmetric signatures, as well as authorization comparisons with counter values, timer values and data values stored on the TPM.

4.5 Device Identification

A device's attributes are its name (a label) and its characteristics (such as its purpose, manufacturer, isolation mechanisms, method of generating random numbers, storage mechanisms, and its stored keys and data).

Device identification is the process of disclosing a device's attributes. Unless a device can be completely inspected, device identification requires a trusted entity to vouch for a device's attributes by signing a credential comprising a description of some (or all) of the device's attributes.

Some trusted entity should vouch for a device by signing a credential comprising the device's attributes. Any type of cryptographic signature scheme may sign a credential comprising a description of device attributes. For unambiguous identification, nothing but the trusted entity should sign credentials with the key that signs credentials comprising a description of a device's attributes.

Often a trusted entity cannot vouch for all of a device's attributes because some attributes (such as keys and data) are generated after the trusted entity vouches for the device. Unless a trusted entity vouches for all of a device's attributes, the attributes signed by the trusted entity should include an *endorsement* key stored by the device.

If all entities other than the device are trusted not to sign data purporting to come from the device, the endorsement key may be a symmetric key. Otherwise, the endorsement key in the credential should be the public component of an asymmetric key whose private component is known only to the device. If a device does not require privacy, the endorsement key should be a signing key.

The TCG specifies [11][12] Endorsement Credentials that vouch for a TPM's attributes, albeit the TPMs in these specifications have an encrypting Endorsement Key. TCG Endorsement Credentials are signed by some trusted entity (typically the TPM's manufacturer) and include the public component of an Endorsement Key whose private component is unique to a TPM. If these Endorsement Keys were signing keys, the specified TPM could sign different types of attribute credential using the Endorsement Key via the TPM commands TPM2_Certify(), TPM2_CertifyCreation(), TPM2_GetSessionAuditDigest(), TPM2_GetTime(), and TPM2_NV_Certify().

A device may itself vouch for some or all of its attributes (a stored key or data object, for example) by signing a credential comprising those attributes, using another signing key that is itself an attribute in a credential issued by a trusted entity. Nothing but the device should use the signing key to sign credentials. The signing key should be stored in Protected Bounded Storage or Protected Mass Storage if the key cannot be replaced. Otherwise the key should be stored in Protected Unbounded Storage.

TPMs can use the commands TPM2_Certify(), TPM2_CertifyCreation(), TPM2_GetSessionAuditDigest(), TPM2_GetTime(), and TPM2_NV_Certify(), with any protected signing key.

4.5.1 Signature verification

If a device must identify itself or other entities using symmetric signatures, the device should be able to sign an HMAC signature using a password. If the signature is crucial to proper device operation, the password should be stored in Protected Bounded Storage or

Protected Mass Storage. Otherwise, the password should be stored in Protected Unbounded Storage.

If a device must identify itself or other entities using asymmetric signatures, the device should be able to verify an asymmetric signature using a public key. If the signature is crucial to proper device operation, the public key should be stored in Protected Bounded Storage or Protected Mass Storage. Otherwise the public key MAY be stored in unprotected memory.

The TPM verifies signatures using the TPM command TPM2_VerifySignature().

4.5.2 Signing

If a device must be identified using symmetric signatures, the device should be able to generate a symmetric HMAC signature using a password. If the signature is crucial to proper device operation, the password should be stored in Protected Bounded Storage or Protected Mass Storage. Otherwise, the password should be stored in Protected Unbounded Storage.

If a device must be identified using asymmetric signatures, the device should be able to generate an asymmetric signature using a private key. If the signature is crucial to proper device operation, the private key should be stored in Protected Bounded Storage or Protected Mass Storage. Otherwise, the private key should be stored in Protected Unbounded Storage.

The TPM signs arbitrary data using the TPM commands TPM2_HMAC() for symmetric signatures and TPM2_Sign() for asymmetric signatures. The TPM signs credentials with the TPM commands TPM2_Certify(), TPM2_CertifyCreation(), TPM2_GetSessionAuditDigest(), TPM2_GetTime(), TPM2_NV_Certify().

4.6 Privacy Enhancements

A device may or may not need privacy when it communicates. Whether a device needs privacy depends on the purpose of the device, what information is revealed to other entities, and what other entities could do with that information.

Two aspects of device identity are privacy sensitive. The first aspect is the ability to distinguish a device from other devices: in other words, whether a device's attributes include something unique to that device. The second aspect is the ability to distinguish a signed credential from other signed credentials: in other words, whether the same cryptographic key is used to verify all identity credentials.

4.6.1 Privacy during identification

For privacy during identification, a device should not sign a credential comprising a description of attributes that uniquely distinguish the device; similarly, the credential (issued by a trusted entity) comprising the description of the verifying key should not include a description of attributes that uniquely distinguish the device.

Privacy during identification is often impossible because many device attributes are unique to a device but must be disclosed. This may not be an issue. Usually the real privacy concern is privacy during recognition.

The TCG specifies [11][12] Endorsement Credentials for TPMs with an encrypting Endorsement Key. The encrypting Endorsement Key is used in a privacy-preserving (more accurately, repudiation-preserving) protocol [13][14] with a Certification Authority to obtain a privacy-preserving credential [12] for an Attestation Key (sometimes called an Attestation Identity Key) protected by the TPM, which the TPM can use to sign [13] credentials. The privacy-preserving property of an Attestation Key credential is that it certifies that the key belongs to a genuine TPM but does not uniquely distinguish the TPM.

4.6.2 Privacy during recognition

Device recognition is the process of matching a device's identity against an existing set of identities.

The same credential signed with the same key using an ordinary cryptographic signature scheme enables a device to be recognized, because the verification key and the verification key credential are always the same. An anonymous cryptographic signature scheme prevents a device being recognized, because the verification key and its credential are always different. A pseudonymous cryptographic signature scheme enables a device to be recognized on multiple occasions by separate entities, because the verification key and its credential are always the same for the same entity but different for different entities.

To prevent recognition, a device should use a different signing key every time it signs a credential. To permit separate recognition by separate entities, a device should use the same signing key when it signs a credential for the same entity but use different signing keys when it signs credentials for different entities.

TPMs can protect an unrestricted number of signing keys whose credentials have been justified with a TPM's Endorsement key. Since an ordinary cryptographic signature with a single key does not protect privacy during recognition, the TCG's TPM credential specifications [11][12] deliberately specify an encrypting Endorsement Key instead of a signing Endorsement Key.

Alternatively, to prevent recognition a device should use the same signing key in an anonymous signing scheme; or, to permit separate recognition by separate entities, a device should use the same signing key in a pseudonymous signing scheme.

TPMs support a cryptographic signing scheme called Direct Anonymous Attestation (DAA) which can create anonymous or pseudonymous signatures. A DAA signature requires a TPM2_Commit() command followed by an ordinary ECC signature created by one of the TPM's signing commands: TPM2_Sign(), TPM2_Certify(), TPM2_CertifyCreation(), TPM2_GetSessionAuditDigest(), TPM2_GetTime(), TPM2_NV_Certify(). The disadvantages of DAA are that it is not widely implemented and it requires considerably more processing than ordinary cryptographic signing schemes.

4.7 Trust Enhancements

Trusted platforms provide services that enable device behavior to be used as authorization to access secrets, or authorization to access networks and other resources such as servers.

The fixed behavior of fixed functionality devices can be inferred from fixed identities. Other types of devices have multiple functions, or are reprogrammable, or can be upgraded. The variable behavior of these devices can be inferred from a variable identity, specifically an identity containing an indication of the software currently executing on the device.

Trusted platforms are distinguished by a permanent function called a Root of Trust for Measurement (RTM) that performs the first operation immediately after device initialization. An RTM measures the next operation that will be performed by the platform, and records the measurement in a safe place where it can be read but can't be altered. The measurement can be used as a proxy of initial device behavior. If the first operation includes a measurement agent that measures the second operation that will be performed by the platform, and records the measurement in a safe place where it can be read but can't be altered, then that second measurement can also be used as a proxy of device behavior. Obviously, the second operation can include another measurement agent, and so on.

Trusted platforms should contain a Root of Trust for Measurement and may contain measurement agents. The RTM and any measurement agents measure software before it executes and record the measurements by *extending* measurements into Platform Configuration Registers (PCRs). The values of PCRs should be used as predictors of the device's behavior.

TCG specification "TCG EFI Protocol Specification" [18] for a PC-Client platform serves to illustrate how an RTM works. TCG specification "Trusted Platform Module Library 2.0" [3] defines a TPM that contains Platform Configuration Registers and can use PCR values for *sealed storage* (using device behavior as authorization to access secrets) and for *attestation* (using device behavior as authorization to access networks and other resources such as servers).

4.7.1 Sealed Storage

Sealed storage is particularly useful for preventing secrets being revealed to the wrong software, or preventing secrets being used by the wrong software, especially when a device boots.

A service on a device might have put sensitive data in protected storage. If that service can be replaced, the sensitive data should be protected from replacement services that have no legitimate right of access. If there is more than one way that a device might provide a service, the sensitive data should be protected from the versions of the service that have no legitimate right of access.

Protected bound storage and protected unbound storage have an authorization method called *unsealing*, which precisely verifies which service requested access to sensitive data in protected storage, or requested the use of sensitive data by protected storage. When sensitive data is *sealed* to a version of a service, the effect is that version of that service must be executing on the device before the sensitive data will be revealed by protected storage or can be used by protected storage. Sealing is particularly useful for revealing sensitive data to whatever operating system or hypervisor has booted on a device, since it is normally the OS or hypervisor that protects sensitive data once it has been released from protected storage.

A TPM seals by storing sensitive data with a measurement of the software that has legitimate access to that sensitive data. When a request is made to reveal a sealed data object, a TPM compares its PCR measurements of the current software environment with the measurements stored with the data object. If the measurements match, the TPM reveals the data object. Similarly, when a request is made to use a sealed cryptographic key, a TPM compares its PCR measurements of the current software environment with the measurements stored with the key. If the measurements match, the TPM allows the key to be used.

4.7.2 Attestation

Attestation is particularly useful for helping a third party, such as a network, determine whether a device will behave as anticipated.

A service on a device might have access to a network. If that service can be replaced, the network should be protected from replacement services that have no legitimate right of access. If there is more than one way that a device might provide a service, the network should be protected from the versions of the service that have no legitimate right of access.

Protected bound storage and protected unbound storage have a signing method called *attestation*, which reveals the software environment currently on the device. Attestation is particularly useful for revealing which operating system or hypervisor has booted on a device, since it is normally an OS or hypervisor that enforces a device's characteristics and reports what applications are executing.

A TPM attests by including a PCR measurement of the current software environment in signed data. When a request is made to access a network, a router or server (for example) compares the signed measurement with the expected measurement. If the PCR values match, the device is admitted to the network.

4.8 Secure Device Updates

In order to preserve confidence in a device, a secure update process should:

- Ensure that only genuine updates can be applied (obviously).
- Have a rollback mechanism in case the update fails (obviously).
- Verify that the device's legitimate administrator has given timely permission for an upgrade to be implemented. This minimizes loss of service while the upgrade is performed.
- Preserve existing sensitive data unless the device's legitimate administrator expressly gives permission for existing sensitive data to be erased. This is important because the loss of some sensitive data, such as cryptographic keys, may irreversibly prevent access to other important data.
- Invalidate any credentials, particularly those of cryptographic signing keys that were invalidated by the upgrade. This may be the case when an upgrade significantly changes a device's functionality or security properties.
- Preserve the manufacturer's means of issuing endorsement credentials for the device, unless the device's legitimate administrator expressly gives permission. Otherwise the device may become incapable of demonstrating that it is a genuine device.

If one successfully installs the newest update available but discovers that the resultant device is flawed, one may need to revert to an older version of the device. One need not install an old update if an old-but-secure update can be promptly reissued, so it becomes the newest update. Otherwise, an old update may be installed if the update process obtains permission from a person with physical access to the device, assuming rogues do not have physical access to the device. It is unwise to install an old update that creates devices with a publicly known vulnerability.

The section “Field Upgrade Mode” of TCG specification “Trusted Platform Module Library 2.0” [3] describes a method of securely updating a TPM.

4.9 Device Software

Devices should use standardized software interfaces.

4.9.1 Protected Storage Software

Protected storage software reduces the amount of knowledge and effort needed to access and use sensitive data.

Devices that host software applications should provide protected storage software that manages protected storage, and provides a convenient interface to protected storage.

When protected storage software is essential for correct platform behavior, protected storage software must be properly designed, implemented, and protected. However, protected storage software doesn't need to be trusted because all it does is manipulate secrets inside a TPM, which protects those secrets until/unless secrets are revealed to platform software. (Platform software that receives secrets obviously should be correctly designed, implemented, protected, and trustworthy.)

The Trusted Computing Group has published TPM Software Stack (TSS) specifications [6]. This TSS manages the TPM and provides high level TPM interfaces for applications. Examples of protected storage software have been published, by IBM [23], Intel [22], and Microsoft [24], for example. Some of these software libraries are implementations of the TCG TSS specifications and some are not.

4.9.2 Conventional Security Software

If a device has sufficient resources, the device should use conventional security software when necessary and appropriate.

Applications, operating systems and hypervisors often access secrets via conventional software interfaces such as MS-CAPI and JAVA-CSP, and use secrets in conventional internet security protocols such as PKCS. Trusted platform services augment such conventional security software, albeit the resultant increase in protection is accompanied by increased complexity. The reasons are that authorization services are required to access secrets in protected storage, trusted platform duplication protocols are required to duplicate secrets stored in protected storage, and protected storage must be managed. PC-Client and server platforms commonly use trusted platform services to improve the protection provided by conventional security software.

4.9.3 Attestation Software

If a device has sufficient resources and supports attestation, the device should use protocols that enable the device to participate in network attestation services.

The TCG's "Trusted Network Communications (TNC) Work Group" has defined standards [15] for endpoint integrity, and can use attestation provided by a TPM. Some TNC specifications have been implemented as StrongSwan [16] open-source software.

5. Cryptographic resources used by Trusted Platforms

Cryptographic primitives are needed to implement trusted platform services. Trusted platform services implemented as software require executable code, memory and a processing engine. Given a library of cryptographic primitives and their RAM and ROM requirements, one may use the tables in this section to estimate how much RAM and ROM is required to implement in software a trusted platform service or trusted computing use case.

Some constrained devices have very limited memory resources and consequently won't be able to implement trusted platform services and use cases unless the device has hardware cryptographic accelerators (for SHA, AES, RSA, ECC, etc.). Hardware accelerators have the additional advantage of stronger process isolation and tamper-resistance than software. Hardware Roots of Trust provide substantially stronger protection than software alone.

Table 2 summarizes the cryptographic primitives used by trusted platforms.

Table 2: Cryptographic Primitives

Cryptographic Primitive	
(p1) Random Number Generator (RNG)	
(p2) Protected persistent data store	
(p3) Hash	
(p4) Extend	
(p5) Encrypt/decrypt	Symmetric cryptography
(p6) HMAC	
(p7) Encrypt/decrypt	Asymmetric cryptography
(p8) Sign/verify	
(p9) Direct Anonymous Attestation	

Table 3 illustrates which cryptographic primitives are used to implement specific trusted platform services.

Table 3: Cryptographic Primitives used to implement Trusted Platform services

Trusted Platform service	Cryptographic primitives
(s1) Isolation (prevent processes from interfering with each other, or from using resources belonging to other processes)	none
(s2) Random Number Generator (a source of unpredictable numbers)	p1

Protected Unbounded Storage (Store an unrestricted number of copies of one or more keys or data objects with access controls and confidentiality and integrity protection. A limited number of stored keys and data objects can have guaranteed erasure and persistence protection)	(s3) Storage Hierarchy of keys and data (a single protected persistent plain-text key provides access to an unrestricted number of protected keys or data objects)	p1 p2 p3 p5 p6
	(s4) Temporary cache (enables multi-threading)	p1 p5 p6
	(s5) Key and data object duplication (exports and imports keys and data objects)	p5 p6 p7
Authorization methods	(s6) password (recognize a local Trusted Computing Base)	p3
	(s7) HMAC (recognize remote entities)	p1 p3 p6
	(s8) enhanced (authorization via a rich combination of methods)	p1 p2 p3 p4 p6 p8
(s9) TPM Software (software that manages trusted functions provided by a TPM, and provides a convenient interface to those trusted functions, but doesn't itself need to be trusted)		p1 p3 p4 p5 p6 p7 p8 p9
(s10) Protected Bounded Storage (Store a limited number of copies of one or more data objects with access controls, confidentiality, integrity, an erasure guarantee, and persistence protection)		p2
(s11) Protected Mass Storage (a storage device capable of protecting potentially large amounts of data-at-rest)		p1 p2 p3 p4 p5 p6 p7 p8
Basic signature services	(s12) Signature verification	p8
	(s13) Sign data	p8
	(s14) Sign credentials	p8
Privacy enhanced signing	(s15) Anonymous or pseudonymous signing	p9

	(create a signature)	
	(s16) Obtain privacy enhanced credentials from a third party (vouch for the attributes of any key or data object)	p7
(s17) Internet security Protocols PKCS MS-CAPI JAVA-CSP (conventional security software)		p1 p3 p5 p6 p7 p8
Trust enhancements for storage and signature services on reprogrammable devices (Protect keys and data objects from unintended software. Enable remote parties to verify the software executing on a device)	(s18) Root of Trust for Measurement (An engine that measures software and records those measurements in PCRs)	p3
	(s19) PCRs and “enhanced” authorization (Confine the usage of stored keys and data objects according to measurements of software)	p2 p3 p4
Trusted signing	(s20) Endorsement Hierarchy of keys (Protect keys that vouch that the device is trustworthy)	p1 p2 p3 p5 p6
	(s21) Obtain trusted credentials from a third party (vouch for the attributes of any key or data object)	p7
Enhanced signing in reprogrammable devices (Perform attestation health checks)	(s22) sign PCRs (vouch for measurements of software)	p2
(s23) secure software/firmware update mechanism (safely modify or update a device)		p3 p5 p7 p8
(s24) TNC protocols		unknown

(network performs health checks)	

Table 4 lists some common uses of trusted platforms, the mandatory services needed to support them, and the optional services needed to support them. Use cases are collected together if they require the same services. More complex use cases rely upon simpler use cases, and hence the services for more complex use cases are supersets of the services for simpler use cases. For convenience and simplicity, Table 4 also indicates the primitives required by a given set of services.

Table 4: Common Uses for Trusted Platform services

Use case	Cumulative use cases	Cumulative mandatory services (and supporting cryptographic primitives)	Cumulative optional services (and supporting cryptographic primitives)
(u1) Can you protect yourself against hardware tampering? (u2) Can you protect computation from tampering	u1 to u2	s1	none
(u3) Can you safely engage in cryptographic protocols?	u1 to u3	s1 s2 (p1 p3 p4 p5 p6 p7 p8 p9)	none
(u4) Can you protect the confidentiality of data from tampering? (u5) Can you protect integrity of data from tampering? (u6) Can you maintain the confidentiality, integrity, and availability of data at rest? (u7) Can you prepare a device for resale or decommissioning?	u1 to u7	s1 s2 s3 s6 s8 s9 (p1 p2 p3 p4 p5 p6 p7 p8 p9)	s4 s5 s7 s10 s11 (p1 p2 p3 p4 p5 p6 p7 p8)
(u8) Who are you? (u9) Can you support common models of provisioning? (u10) Can you be managed remotely?	u1 to u10	s1 s2 s3 s6 s8 s9 s12 s13 s14 s17 s20 (p1 p2 p3 p4 p5 p6 p7 p8 p9)	s4 s5 s7 s10 s11 s15 s16 (p1 p2 p3 p4 p5 p6 p7 p8 p9)
(u11) Can I trust you? (u12) Can you protect computation from tampering (u13) Can you securely maintain evidence? (u14) Can you detect malware infections?	u1 to u15	s1 s2 s3 s6 s8 s9 s10 s12 s13 s14 s17 s18 s19 s20 s21 s22 s24 (p1 p2 p3 p4 p5 p6 p7 p8 p9)	s4 s5 s7 s10 s11 s15 s16 (p1 p2 p3 p4 p5 p6 p7 p8 p9)

(u15) Can you maintain secrets while infected?			
(u16) Can you stay healthy? (u17) Can you recover from infections?	u1 to u17	s1 s2 s3 s6 s8 s9 s10 s11 s12 s13 s14 s17 s18 s19 s20 s21 s22 s23 s24 (p1 p2 p3 p4 p5 p6 p7 p8 p9)	s4 s5 s7 s10 s11 s15 s16 (p1 p2 p3 p4 p5 p6 p7 p8 p9)
(u18) Can You Secure Legacy Hardware?	u1 to u18	s1 s2 s3 s6 s8 s9 s10 s11 s12 s13 s14 s17 s18 s19 s20 s21 s22 s23 s24 (p1 p2 p3 p4 p5 p6 p7 p8 p9)	s4 s5 s7 s10 s11 s15 s16 (p1 p2 p3 p4 p5 p6 p7 p8 p9)

6. Appendix

This appendix introduces overlay networks, which provide perimeter security for devices that are connected via that overlay network. Even so, if devices connected by an overlay network have no inherent security, a successful attack on one device may still enable attacks on other devices.

6.1 Overlay Networks

An overlay network may be able to plug gaps in the protection of devices that have an incomplete set of trusted platform services: if a device can identify itself and provide some simple attestation, a gateway in the overlay network might be able to provide additional key provisioning, secure communication, software update, and other trusted platform services.

Therefore devices should be provided with a cryptographic identity and be capable of attestation.

One definition of an overlay network is that given in the International Society of Automation's ISA-100 [20].

The TCG's "IF-MAP Metadata for ICS Security" specification [19] describes an overlay network intended to facilitate "secure deployment and management of large-scale industrial control systems by creating virtual OSI layer 2 and/or layer 3 overlay networks on top of standard shared IP network infrastructure - particularly (though not necessarily) TNC-compliant IP network infrastructure".

The AllJoyn overlay network [21] is a derivative of the Linux D-bus. AllJoyn devices can be directly plugged into a Windows™ platform or connected to the same wired or wireless network as a Windows platform. AllJoyn routers on a Windows platform use broadcasts to share information about provider devices and consumer devices. The device directory is dynamic, so devices can come and go. The router establishes secure end-to-end communications between providers and consumers, and enables reading of a value, calling a function and getting a value back, sending an asynchronous command with no response, and requesting a notification. Two devices can be configured to talk directly to each other. For example, a light switch can be configured to send an "ON" or "OFF" command to a light bulb. Windows also includes a gateway that interfaces with legacy networks like ZigBee or Bluetooth, translating the legacy protocols into AllJoyn.

TPMs currently support many cryptographic algorithms, but currently not the efficient (symmetric) GCM encryption/identification that is used by many low-power devices.