R
E
F
E
R
E
N
C
E

Guidance for Securing IoT Using TCG Technology

Version 1.0
Revision 21
September 14, 2015
Published

Contact: admin@trustedcomputinggroup.org

TCG Published

TCG

Disclaimers, Notices, and License Terms

THIS DOCUMENT IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, DOCUMENT OR SAMPLE.

Without limitation, TCG disclaims all liability, including liability for infringement of any proprietary rights, relating to use of information in this document and to the implementation of this document, and TCG disclaims all liability for cost of procurement of substitute goods or services, lost profits, loss of use, loss of data or any incidental, consequential, direct, indirect, or special damages, whether under contract, tort, warranty or otherwise, arising in any way out of use or reliance upon this document or any information herein.

This document is copyrighted by Trusted Computing Group (TCG), and no license, express or implied, is granted herein other than as follows: You may not copy or reproduce the document or distribute it to others without written permission from TCG, except that you may freely do so for the purposes of (a) examining or implementing TCG documents or (b) developing, testing, or promoting information technology standards and best practices, so long as you distribute the document with these disclaimers, notices, and license terms.

Contact the Trusted Computing Group at www.trustedcomputinggroup.org for information on document licensing through membership agreements.

Any marks and brands contained herein are the property of their respective owners.

Acknowledgements

The TCG wishes to thank all those who contributed to this specification. 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:

Tom Moulton	Atmel
Stacy Cannady (Editor, IoT-SG Co-Chair)	Cisco Systems
Max Pritikin	Cisco Systems
Andreas Fuchs	Fraunhofer Institute for Secure Information Technology (SIT)
Lawrence Case	Freescale Semiconductor
Yoshitaka Hiyama	Fujitsu Limited
Seigo Kotani	Fujitsu Limited
Darren Krahn	Google
Tom Laffey	Hewlett-Packard
Jim Mann	Hewlett-Packard
Ira McDonald (Editor)	High North
Nicolai Kuntze	Huawei
Guerney Hunt	IBM
Sung Lee	Intel Corporation
Alan Tatourian	Intel Corporation
Steve Hanna (Editor, IoT-SG Co-Chair)	Infineon Technologies
Paul England (Editor)	Microsoft
Merzin Kapadia	Microsoft
David Wooten	Microsoft
Charles Schmidt	The MITRE Corporation
Hidekazu Segawa	Ricoh Company LTD
Graeme Proudler	Self
Tom Brostrom	United States Government
Jonathan Hersack	United States Government
Andrew Cathrow	Verisign
Andrew Tarbox	Wave Systems

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

Primrose Mbanefo, Accenture
Dr. Dietmar Wippig, BSI
Jesus Molina, Fujitsu
Gerald Maunier, Gemalto
Jack Ring, ontopilot.com
Brian Witten, Symantec
Maarten Bron, UL
Arjan Geluk, UL
Andrew Jamieson, UL

Table of Contents

1. Scope, Audience and Purpose

1.1 Scope

This document describes typical IoT security use cases and provides guidance for applying TCG technology to those use cases.

Because IoT devices vary widely in their cost, usage, and capabilities, there is no one-size-fits-all solution to IoT security. The practical security requirements for different devices and systems will vary. Therefore, this list of solutions should be regarded as a menu from which the implementer can pick the options most suitable for their product or service.

This document is not a TCG Specification and therefore is not normative. Further, this document does not provide enough detail for a product or solution to be directly implemented by reviewing this document alone. Many other aspects and design issues must be weighed and requirements resolved to create a product or solution.

1.2 Audience and Purpose

The intended audience for this document is providers of IoT devices, software, and services. The document is a high-level introduction to how TCG technology can be applied to solve security problems in the Internet of Things market space. As a high level document, it is suitable for both business and technical readers as an initial starting point for an investigation of whether TCG technology is suitable as a solution for the reader's security requirements.

2. Preface

Most computer security is implemented at high levels in the software stack: for example, operating systems use cryptography to secure data at rest and in motion, and operating systems and applications are crafted and configured to protect user-privacy and be robust to malicious inputs. Although much progress has been made in the science and practice of building secure systems, it remains true that most non-trivial software systems will have exploitable bugs. Traditional recovery of infected and exploited systems has been time consuming and expensive: for instance operating systems and applications need to be re-installed, and passwords and machine credentials need to be changed. This has usually meant physical access (e.g. to install from a DVD) and access to important credentials (for example to enroll a device with a corporation.)

The next wave of IoT will bring orders of magnitude more devices: some with UI, some without; some physically accessible, and others not. The scale and diversity of this new world of computing demands a radical re-think of how we identify and manage devices remotely and at-scale.

Once more, most of the next wave of IoT software and service machinery will be implemented high in the software stack, but in the face of software bugs, some things will simply not be possible without some hardware support. For instance, with software-only solutions attackers will probably be able to irrevocably brick devices. Other attacks will steal device secrets that can never be securely re-provisioned, forever allowing attackers to impersonate a device or eavesdrop on its communications. These problems are not new or unique to IoT systems but they are more troubling with IoT systems because IoT systems are numerous, minimal in their security features, impractical to administer manually, and sometimes dangerous when compromised. In short - software-only solutions are fragile, and prone to irrevocable damage.

Fragile software-only solutions represent risks to consumers and to device and service providers. Device providers risk warranty returns for systems that cannot be repaired in the field. Customers risk their data, their privacy, and their time. In the very worst of cases, customer health and wealth may be put at risk.

TCG technologies do not provide an immediate solution to all IoT device and service security needs, but they enable existing and new IoT solutions *to be fundamentally far more robust than today's state-of the art*. This document defines a set of security-related use cases, and describes how TCG technologies can be applied to the problems.

3. Use Cases

In this section we describe a set of fundamental security capabilities that will be required of many IoT devices. In the IoT Framework section (section 4), we describe how TCG technologies can solve these problems.

The fundamental security capabilities are:

- **Establishing and Protecting Device Identity**

  IoT devices should have the ability to perform mutual authentication with IoT services or with other IoT devices. All parties can then use the results of this authentication to determine authorization and/or to log the identity of other parties. This prevents unauthorized IoT devices from gaining access to IoT services and prevents unauthorized parties from masquerading as IoT services. Further, it promotes accountability and enables forensic analysis.

- **Protection Against Malware Infection**

  IoT devices should be able to resist malware infections, both volatile and persistent. If a malware infection takes place, these devices should minimize the impact and enable recovery.

  - **Protecting Device Health**

    IoT devices should include a mechanism for securely determining software/firmware versions and a secure software/firmware update mechanism. This helps devices stay one step ahead of malware by rapidly and securely installing updates to known vulnerabilities.

  - **Detecting Malware Infections**

    Malware detection enables a variety of responses such as mitigation and remediation. However, malware is often stealthy, employing a variety of ruses to avoid detection. Therefore, malware detection must be equally clever.

  - **Recovering from Infections**

    Inevitably, some IoT devices will become infected with malware. When this happens, safe recovery should be feasible. This includes the ability to detect an infected device, restore it to a healthy state, and resume proper functioning. This process should not require physical access to the device. Instead, the recovery process should take place over the network.

  - **Maintaining Secrets while Infected**

    If an IoT device is infected with malware, important secrets such as user data and long-term keys should be protected so that the malware cannot access them.

- **Protecting Against Hardware Tampering**

  Some kinds of IoT devices need to protect themselves against hardware tampering. For example, electric meters typically give consumers unlimited physical access along with an incentive to hack the device and steal service. In such circumstances, complete protection against tampering is often not possible. However, it is possible to raise the cost of tampering so that it requires specialized equipment or to limit the scope of the damage caused by such tampering.

  - **Protecting the Confidentiality of Data**

    Some data must be protected against disclosure. For example, an attacker that can copy secret cryptographic keys from an IoT device may be able to impersonate that device or obtain confidential user data.

  - **Protecting the Integrity of Data**

    Some data must be protected against unauthorized and undetected modification. For example, an attacker that can modify the readings on an electric meter may be able to steal power.

  - **Protecting Computation from Tampering**

    If computation can be interfered with, security checks can be skipped and the reliability of the IoT device can be compromised.

- **Confidentiality, Integrity, and Availability of Data at Rest**

  Confidential data stored on an IoT device should be protected.

- **Reselling or Decommissioning a Device**

  Resale and decommissioning are inevitable phases in the device lifecycle, especially for expensive devices which are likely to have a significant resale value. Before a device is resold or decommissioned, any sensitive data belonging to the previous owner should be securely erased. Then the device can be securely transferred to a new owner or prepared for disassembly and recycling.

- **Meeting Cryptographic Protocol Requirements**

  All IoT devices are in some way connected to a network that may not be trustworthy. Cryptographic protocols ensure the security of communications over that network and should be supported. Good sources of entropy, secure key storage, and cryptographic acceleration may be needed. Because cryptographic algorithms eventually become weakened and then obsolete, cryptographic agility may also be needed, especially for long-lived IoT devices.

- **Supporting Multiple Models of Provisioning**

  IoT technologies must support practical, common methods of provisioning credentials, policies, and anything else needed to make an IoT device functional for the customer. Some IoT devices will be provisioned during manufacture, others at first use. Some devices will be provisioned under conditions of physical security, and others by end users. In some cases, customers may wish to use anonymous remote attestation and other techniques to protect their privacy.

- **Maintaining Audit Logs**

  Secure logging is essential to maintaining accountability and enabling forensic analysis.

- **Remote Manageability**

  Most IoT devices need secure remote management capabilities. Requiring physical access to manage an IoT device won’t scale to a large number of devices.

- **Securing Legacy Hardware**

  The world is currently full of legacy devices that do not support these use cases. Fortunately, the security of these devices can be improved using gateway devices that handle the security for them.

The contents of this document are intended to span these use cases but are not intended to be limited to these use cases.

4. IoT Framework

This section provides general guidance but not implementation details on how to use the Trusted Computing Group's technologies and standards to address the use cases defined in section 3.

Because IoT devices vary widely in their cost, usage, and capabilities, there is no one-size-fits-all solution to IoT security. The practical security requirements for different devices and systems will vary. Therefore, this list of solutions should be regarded as a menu from which the implementer can pick the options most suitable for their device or service.

4.1 Establishing and Protecting Device Identity

Almost all IoT scenarios require reliable authentication of the devices in use, but unfortunately the Internet does not provide reliable endpoint authentication so devices must identify themselves instead. There are many types of device identifiers in common use: simplest, and probably least secure, is a public name or globally unique identifier (GUID). However, a public name or GUID by itself does not provide authenticated identity for an IoT device because adversaries that obtain the name or GUID can impersonate the device.

A second common technique is to use a cryptographic identifier (e.g., 802.1AR device IDs [802.1AR]). However, even when cryptographic device identifiers are used, many devices manage secret keys with software alone. Unfortunately, if software managing the secret key is vulnerable, then the key can leak and adversaries can impersonate the device. If this occurs the device can probably only be safely re-provisioned under conditions of physical security, and this might require physical access to the device, or even return to the manufacturer. This is costly, and may not even be possible. Therefore IoT devices should be furnished with cryptographic identities that are robust to the sorts of attack that the device is likely to suffer.

The TPM provides cryptographic device identities that are robust in the face of malware attack, and many TPMs also provide good key-protection against relatively sophisticated hardware attacks. As such, the TPM is a highly resilient foundation to use for IoT device identity. TPM capabilities that can be used to provide device identity include symmetric-key encryption, HMAC, and asymmetric cryptography (commonly RSA and ECC.) [TPM2][TPM-IDENTITY]

Device identities must be used in robust cryptographic protocols to thwart common attacks (replay, man-in-the-middle, etc.) For example, a device identity might be used in mutual authentication of a Transport Layer Security (TLS) session and to digitally sign integrity information as proof of the source of that information.

The TPM also supports a variety of provisioning flows, including provisioning of keys during chip manufacturing, device assembly, enrollment with an IoT management service, or owner-personalization. During TPM provisioning, "key attestation" can be used to allow one TPM-based key to certify that another TPM-based key is hardware-protected, thus providing more confidence in the security of the key storage. Alternatively, secure key-import can be used to install new identities over an untrusted network.

We note that there are privacy implications inherent in the use of cryptographic identities, and solution providers should carefully consider whether IoT-devices employing TCG technology are facilitating privacy hazards for their users. For example, it would generally not be considered a privacy hazard to allow unambiguous cryptographic identification of a device providing a public service (say a traffic camera.) In this case all users can rely upon the same device identity key – for example, a TPM Endorsement Key or other TPM key that is tied to the device. On the other hand, a TPM-equipped personal device that uses third-party web services (e.g. a weather feed, a traffic feed, etc.) should not reveal any long-lived keys that allow unwanted tracking. If secure pseudonymous identities are required, the TPM-based Attestation Identity Keys or Direct Anonymous Attestation can be employed. [TPM2]

Solution developers can use the TPM Software Stack (TSS) library to build libraries and tools to provision and use TPM-based IoT device identities. Vendors offer various proprietary APIs built on top of TSS or as proprietary instances of a TSS. These proprietary offerings might support features needed by the device manufacturer.[TSS]

4.2 Protection Against Malware Infection

Several TCG technologies provide protection against malware infection, as described in the subsections of this section.

4.2.1 Protecting Device Health

Many of the TCG standards provide strong building blocks that can be used to implement or supplement IoT system security.

One commonly used way of limiting how much damage malware can do is to prevent unauthorized writes to security-critical programs and data. TCG Self-Encrypting Drives, such as the commonly available “Opal” drives, include logic that firmware and operating systems can use to write-protect some or all of the IoT-device’s state. [OPAL]

The Trusted Network Communications (TNC) standards [TNC-ARCH] include a standard way to check which software or firmware is running on a particular device, including the version number. They also provide a remediation mechanism that can be used to provide instructions for obtaining and applying software and firmware updates.

To check which software or firmware is running on a particular device or perform other device health checks, use the IF-M protocol [IF-M] to query the endpoint. For IoT applications, this check will generally run over TLS using the IF-TNCCS [IF-TNCCS] and IF-T/TLS [IF-TTLS] specifications.

To gain greater confidence in the veracity of a software or firmware version check, use the TPM’s Measured Boot and Remote Attestation capabilities, as described in TCG’s white paper “Trusted Network Connect: Open Standards for Integrity-based Network Access Control” [INTEGRITY].

Traditionally, run time health verification has been handled by anti-malware products in larger systems. Whitelisting and only allowing binaries signed by the manufacturer are two good techniques for assuring only certain code is executed on the device. Use of TPM-assisted software updates, static code analysis, runtime stack protections, data execution prevention, compliance verification, and policy updates are all options that the device manufacturer can consider for assuring the integrity of the run time environment. Some of these techniques may not be practical on especially minimal devices. In that case, the only option may be to reboot periodically and use boot-time protections.

If a device requires remediation, the Remediation Instructions attribute included in IF-M [IF-M] may be employed. This attribute is generally used for manual (human-assisted) remediation today, but automated remediation can be achieved using a Remediation URI or a vendor-specific Remediation Parameters Type.

We note that practical security requires ongoing investments in software maintenance because patching is central to secure systems. If a device vendor goes out of business, or limited time-period service contracts expire and updates are no longer available, then device security will start to degrade as vulnerabilities are discovered. In light of this, some customers may wish to take full control over the IoT-client software and associated network services.

4.2.2 Detecting Malware Infections

In general the detection and remediation of malware is a hard problem because malware seeks equivalent or higher privilege than the systems that are seeking to detect and isolate it. Secure boot mitigates this problem by examining each module before it is allowed to run. However, secure-boot system policies tend to be relatively coarsely defined, potentially allowing bad or vulnerable software to load.

If more fine-grained or run-time malware or security policies need to be enforced, TCG technologies offer an alternative model called attestation that is manageable even when large numbers of software modules are involved. Attestation is a platform capability that allows authoritative reporting of the software or security configuration of a platform. Attestation can provide a very detailed report of security posture, and relying parties can choose whether to communicate further, quarantine or demand remediation. Well-implemented attestation-based systems drastically increase systemic security because known-bad or known-vulnerable systems can no longer communicate.

This architecture is provided by the TPM's Measured Boot and Remote Attestation capabilities, as described in TCG's white paper “Trusted Network Connect: Open Standards for Integrity-based Network Access Control” [INTEGRITY]. This technique can even detect changes to BIOS or other firmware. Some SoC (System on Chip) vendors also offer basic hardware capabilities that have attestation functions.

4.2.3 Recovering from Infections

Once malware has been detected as described in the previous section, the IF-PEP protocol [IF-PEP] can be used to isolate the infected machine to prevent the infection from spreading.

There are a number of possibilities for remediation. Examples in use today include:

- Self validation and self remediation. In this model, the device keeps a set of golden measurements in read-only protected storage and the golden measurements are compared to current measurements made during boot. If there is a validation failure for a module, the device can delete the affected module and re-install a saved copy of that module from a local library of Last Known Good code. The system then restarts in an iterative process until all modules validate.

- Remote validation. In this model, the device measures its own integrity as part of boot, but does not validate those measurements. When the device applies to join a network, part of joining involves sending an integrity report for remote validation. If validation fails, the end point is diverted to a remediation network for action.
- Runtime integrity. Several commercial products are available that implement this model. They all perform runtime checking of code in execution. When a problem is found, the client code on the affected system handles the problem in different ways. It might replace infected code with a clean copy from storage, it might appeal to peers and request a clean copy from them, or it might announce to a remote PDP that it is now untrustworthy and wait on remediation.

Infected devices may exhibit arbitrary behavior, so in general it is the responsibility of other devices and services to quarantine or reject communications from devices that are not able to prove themselves sound. Devices that communicate with local or cloud-based hubs admit a single point of control for security assessment and quarantine. If systems employ peer-to-peer communications then this function must be distributed across all devices (which itself is may be problematic if an infection is widespread.)

In light of this complexity, system designers should consider employing a spectrum of protection and remediation technologies to increase system resilience.

Architects should also consider the wider implications of quarantining: for instance it may be better to allow an infected IoT device to function if that device provides a service critical to life.

Finally, system vendors should strive to build systems that can recover without loss of user data or important system configuration.

4.2.4 Maintaining Secrets while Infected

IoT devices often work unattended by humans and may operate unmanaged for extended periods of time. These devices may store confidential or privacy-sensitive information such as consumer habits or manufacturing parameters. This raises a concern about the ability of unattended devices to continue operating as designed, including maintaining the confidentiality of secrets used by the device, in the face of a successful infection by malware.

The ability to maintain the confidentiality of secrets as they are used in the presence of malware infection is a problem that requires a layered approach to solve. The layered approach starts with good security engineering in the software architecture of the device and in the implementation of that architecture.

This secure architecture will depend on technology artifacts to create the secure envelope within which device secrets are protected. Some modern processors include execution modes designed to protect security-critical subsystems. These subsystems permit high-speed execution of application code but may be vulnerable to bugs in supporting software. TPM functions can be implemented using these subsystems. Dedicated TPM hardware can provide more secure cryptographic operations and integrity checks. When used together with these subsystems and execution modes, a dedicated TPM can attest to the integrity of application code and supporting software while providing strong security for cryptographic keys and operations.

4.3 Protecting Against Hardware Tampering

Hardware tampering means that an attacker has physical control of the device for some period of time. Broadly speaking, hardware tampering might occur at any of three different periods in the life cycle of a device:

1. During manufacture. In this model the attacker has access to the device as it is designed or during its manufacture. The result is that the device is built to support features and capabilities that are unknown to the device manufacturer and to customers who buy the device. This should also include that possibility that an attacker will compromise components built by a supplier of the device manufacturer in order to compromise a target device.
2. Between shipping the device from thedevice manufacturer's dock to receiving the device at the customer's dock. In this model, the attack intercepts the device as it passes through distribution on its way to a customer site. The result is that the device may have new capabilities, expected capabilities may now act in an unknown way and secrets may have been added, changed or removed from the device.
3. During deployment and usage, while serving the customer's needs. In this model, the attacker gains access to the device during the productive life of the device. Once again, the result may be that the device no longer behaves as expected, and/or its secrets may be stolen or changed.

With regard to compromise during design and manufacture, the customer should conduct serious conversations with their vendors on the topic of Secure Design Lifecycle and supply chain security as practiced by the vendor (and their suppliers). With regard to compromise in transit, this is also a supply chain matter, but the customer will have to address the distribution chain between the device manufacturer and his dock. With regard to compromise of a device in deployment within a customer network, it is the responsibility of the customer to have done the risk assessment required to understand what level of security capability is required to cost-effectively protect data processed through devices used to execute the business process. Not all security measures are created equal. Low risk assessments mandate security measures that can be less robust, but also less expensive. High risk assessments mandate security measures that are more robust and therefore more expensive.

The issue of whether an appropriate risk assessment has been done is the foundation of the response for each of sections 4.3.1 through 4.3.3 below. The mission of effective data security is to make it "more trouble than it is worth" for the attacker to be successful against his chosen target.

A complicating factor to consider in this otherwise common sense approach is the lifetime of the device in deployment. Industrial control systems can remain in service for 50 years or more. Automobile manufacturers plan on 30 years for the lifetime of a car. Network infrastructure equipment can remain in service for 15 years. From a security perspective, security measures that were impossible to breach years ago may be vulnerable today. A best practice approach to lifetime security is to engineer security in a modular, upgradeable and replaceable manner. This makes it possible for the device manufacturer to replace obsolete security components as time goes on.

OEMs should also keep in mind that security engineering best practices

- Forbid the hard-coding of secrets in code or files in a device,
- Forbid the deployment of back doors or admin accounts as part of released products,
- Require removal of debug code from released products,
- Forbids a security design that calls for the use of a secret that is shared by all products.

The following general remarks apply to each of sections 4.3.1 to 4.3.3.

Since we are focused on hardware tampering, that means that the customer should consider solutions that implement the security envelope inside security hardware that includes countermeasures against tampering. Having said that, some security hardware is more robust than others.

A risk analysis should provide the information necessary to define the size and capabilities of the HSM (Hardware Security Module). It may be that the HSM is nothing more than shielded NVRAM that is used to protect one or more roots of trust for the platform. It may be that the security envelope must be substantially larger and more capable. This risk analysis costs time and resources to perform, but the payoff can be substantial in terms of not over-spending or under-spending on security while still protecting the brand from damage that comes as part of a failed security implementation.

A hardware-based security envelope might be nothing more than a general purpose microprocessor that is isolated from other processing within the device. The security envelope is created by isolation of the processing of confidential data from other processing on the device. This is a low bar for an attacker with possession of the device.

Beyond the use of a general purpose processor, there are processors that support a variety of hardware features that are designed to make it harder for an attacker who has physical possession to compromise the device. Use of hardware countermeasures as the primary tool for defending against tampering places the HSM in a middle range of resistance to physical attack. Most TPM chips fall in this category.

At the high end of resistance to physical attack are HSMs that use hardware, firmware and software security mechanisms coordinated to resist physical attack. This method of protection evolved to protect personal financial data stored and used on smart cards and to protect confidential information on set-top boxes.

4.3.1 Protecting the Confidentiality of Data

In this case, the objective of the security design is to

- Protect confidential data at rest by encrypting that data and storing the encryption key within a security envelope.
- Protect confidential data in process by decrypting and processing confidential data within a security envelope. Once processing is complete, the confidential data must be re-encrypted before being written to storage.

The TPM is an example of an HSM designed to protect specific small secrets, such as keys and to protect a specific set of crypto operations using those keys, like digital signatures. It is not designed to be for bulk data encryption. Secure processor modes can be used to protect keys and ongoing computation, although practical security will be degraded if very large subsystems are run in isolated containers because the software systems themselves may contain exploitable bugs.

For protection of data at rest, the customer should consider the use of self-encrypting storage hardware or software based encryption. Self-encrypting storage hardware features high speed bulk data encryption hardware integrated into the storage device controller. Data written to the storage media is encrypted as it passes through the hardware encryption engine. Data read from the device is decrypted as it passes through the hardware encryption engine. The encryption engine operates at bus speed (minimal performance impact) and the key used to encrypt and encrypt data (called the Media Encryption Key or MEK) is non-exportable from the storage device controller.

4.3.2 Protecting the Integrity of Data

There are a few ways to protect data against an attack intended to perform unauthorized change. One is to use a Write Once or Read Only storage protection. This approach can provide high assurance that the integrity of the data at rest can't be changed (depending on the hardware mechanisms that enforce Write Once). The TPM supports a small amount of non-volatile RAM that features a Write Once technique. The available NVRAM within a TPM can vary from one chip maker to another. It is usually small – around 10K bytes.

Another mechanism is to restrict access to keys based on policy. For example, it is possible to write policy for the protection of a secret (like an encryption key) that states that if the software on the device is not in a certain configuration or if the integrity of the software is not specifically a certain value, the TPM shall not release the secret.

For larger volumes of data (e.g. executable code or archives of documents) another protection mechanism is to use standard cryptographic hash as a mechanism for validating the ongoing integrity of data of interest. In this model, a set of files that are known to be good are hashed (it could be as a group, as sets or as single files) and hashes are protected as the golden measurements. In the future, the files can be re-hashed at any time and the current hash measurements can be compared to the originals. If they match, the integrity of the data has not changed.

This mechanism can be used as a way to identify unauthorized change to executables and configuration files. It can also be used to verify the integrity of documents and it is the basis of assuring the integrity of a digitally signed document.

4.3.3 Protecting Computation from Tampering

Malware frequently uses two techniques to insert itself into a target platform. One is to modify code in memory. This technique can only last until the system is rebooted. To install in a fashion that can survive reboot, malware must use the second technique: modifying files. As stated in section4.3.2, above, the TPM can be used to protect current hash measurements of important files and data and produce a digitally signed report (called an “integrity quote”) of those measurements at any time to any entity. The digital signature on the integrity quote uses a key that cannot be exported from the TPM, thus providing evidence of which TPM (and therefore which device) produced the report. An external entity that has access to the original measurements can compare those measurements to the provided report and determine whether code on the device in question has changed or not.

Another option available for detecting tampering against executable code in the device is to use the TPM as a way of creating an audit log of the integrity of software. The way the log is built is that the code in question is measured or hashed on a periodic basis. Each new measurement is extended into the log. The value of this historical log can be predicted (if no changes were made or if authorized changes were made). If the current value of the log does not match the expected value, the software has been tampered with.

Finally, with regard to attack against computation done within the TPM, there are differences between TPM devices offered by different vendors. Some vendors provide protection for the TPM as a matter of differentiation against their competition. If protection against tampering with the computations done by a TPM is important, check with your TPM vendor to see what help they can provide with their product.

4.4 Confidentiality, Integrity, and Availability of Data at Rest

4.4.1 Availability

IoT-devices will employ a mix of read-only and read-write memory technologies to store their computer programs and data critical to their operation. Destructive malware will seek to corrupt or delete writable state, so protection measures must be employed. Simplistic solutions to this requirement place all IoT device code in ROM, but this will obstruct device updates, and will generally not be acceptable.

The TCG has defined a variety of technologies that seek to limit exposure to attacks on the availability of writable state. One key concept is that of a Root of Trust for Update or RTU. The RTU is the minimal functionality needed to perform a secure update of a device. Although not explicitly described in TCG specifications, having an RTU check a certificate on a software upload is a common implementation for a secure minimal-RTU. The NIST document [800-147] describes requirements for PC-platform firmware-updates that are also applicable to IoT-devices.

Platforms must also implement protections that ensure that only the RTU can perform an update. TCG has defined a family of storage controller technologies known as “Opal” that allow storage regions to be unlocked for write access by an entity that can provide proper authentication (such as a password). [OPAL]. One Opal-supported scheme permits write operations to a region early in boot but allows the RTU to write lock the storage region before passing control to (potentially) untrusted software. It is outside the scope of TCG specifications to describe how these passwords may be managed, but one technique is to use the TPM to ensure that the password is only accessible to the properly authenticated RTU.

4.4.2 Confidentiality and Integrity

Many IoT devices will store confidential data. Some of this data may be customer data, and some may be device data – for example, keys used to ensure updates are secure. This data is also under threat from two sources: one is malware that manages to subvert the device, and the other is physical attack for devices that are lost, stolen, or operate under conditions of poor physical security.

TCG describes many technologies that allow a device manufacturer to build systems that provide robust protection for confidential data. The Opal storage technologies described above allow storage regions to be not only write protected (as previously described), but also configured so that only authorized entities can unlock the storage region for read access.[OPAL] A common use case is to provide a storage area that can only be accessed prior to OS boot (because early boot code is generally smaller, simpler and less prone to bugs than the final running system).

The TPM is also a powerful device for the protection of device data.[TPM2] One capability is a non-volatile storage feature: the TPM implements a sophisticated authorization model for the entities and circumstances under which data can be read or written. Authorized entities can be identified by program hash, proof-of-knowledge of a second secret (possibly low entropy, like a PIN), time, software configuration, etc. Unfortunately the NV-storage capacity of most TPMs is modest (perhaps kilobytes), but it is usually sufficient to protect authentication credentials (for self-encrypting drives) or encryption keys (for software FDE).

4.5 Reselling or Decommissioning a Device

Because resale or decommissioning are a natural part of the device lifecycle, the device manufacturer should include support for these use cases in the design of the device. Generally, two steps are necessary: securely erasing any sensitive user-data and resetting the device back to factory settings so that it can be configured by the new owner. With a TPM, this is performed by using the TPM2_Clear command [TPM] to release ownership. If all sensitive data was encrypted with keys stored in the TPM, this data will no longer be accessible. All self-encrypting storage solutions in the market today support a command to delete the current MEK (Master Encryption Key) and generate a new one. When this command is executed, all data on the storage device is permanently lost – a process called a “crypto erase”. The new owner of the device can verify that the proper software is loaded on the device using the techniques described in section 4.2.1 and can verify that the device has been reset using commands in this software. Then the new owner will need to take ownership of the TPM and personalize the device.

In addition to sensitive user-data, many IoT-devices will be furnished with keys from the manufacturer or service provider. Depending on the behavior of the device and service, these keys may need to survive a change in owner of the IoT device. The TPM defines different families of data and associated control so that (say) a user is authorized to clear all user data, but only the device manufacturer can clear or re-provision keys representing fundamental device identity.

4.6 Meeting Cryptographic Protocol Requirements

If the device manufacturer intends to produce devices that are capable of encryption and the target market includes national governments, then it is likely that there will be a requirement from those governments to comply with guidelines about how encryption is to be done. This includes how random numbers are generated, how keys are generated, what cryptographic algorithms are used, how keys are managed and protected and many other specifics with regard to encryption. In many cases, failure to comply with these guidelines means that the device manufacturer’s product will not be purchased by national governments. The TPM 2 specification [TPM2] includes support for true random number generation, cryptographic key generation, secure key storage, cryptographic hashing, and both asymmetric and symmetric cryptography with a choice of cryptographic algorithms. Because TPM 2 is a library specification, each TPM platform profile specifies which of these features are required and optional for that platform. The TPM 2 specification also supports some flexibility in terms of which algorithms can be run within a TPM. Refer to the TPM Algorithm Registry [ALGREG] for the range of choices. Interested OEMs are once again directed to the TPM vendor community to find out what security compliance testing the TPM vendors have already undertaken. In general, the device manufacturer will still have to undertake compliance testing of the device, of which a compliant TPM is a part. The presence of a TPM in a device does not necessarily make the device secure.

4.7 Supporting Multiple Models of Provisioning

IoT devices can flow through a variety of provisioning steps on their way to final operation. Steps may include silicon manufacture (including TPMs), assembly by the device manufacturer, (possibly) device personalization by the vendor, and final configuration by the end customer. Some devices may also support de-provisioning for retirement or resale. Not all IoT devices will have local user interfaces, which can limit strategies for device enrollment and configuration.

In this section we confine our discussion to the provisioning and management of device keys. Generally, once one key has been provisioned, this key can be used to bootstrap arbitrarily complex configuration software and state. The TPM can be a powerful device for secure enrollment of devices, even under poorly secured conditions like an outsourced device production line or even a remote physical location.

TPMs incorporate long lived device identities called Endorsement Keys. A TPM endorsement key will typically live for the life of the TPM, and can be used as the basis of identity for an IoT-device. Endorsement Keys are usually public-private key pairs, and are usually certified by the TPM manufacturer. Once a management authority knows the public key of a device it can securely perform a wide range of software deployment and configuration steps. Association of TPM public keys to manufactured devices is typically the most challenging step, but securely managing a public key database (possibly with certificates to ensure key-veracity) is typically much easier than the secure deployment and management of secret keys.

Often, OEMs want to add a device key into each IoT device during device manufacture, enabling authorized devices to be identified in the field. Without a TPM, this can be a painful process requiring physical security on the production line for the key generation and insertion process. Using a TPM on each device, this process can be greatly simplified. Each TPM can generate the device key for its device and use the TPM's EK to vouch for the device key's security and validity. By using this mechanism in conjunction with controlled issuance of credentials and licenses to devices, overproduction and other forms of fraud on the production line can be prevented. More detailed guidance on this important but complex topic will be coming from TCG soon.

The TPM can also be used to securely establish the initial (and later) firmware/software images. If a device implements measured boot, then provisioning services can securely establish (a) the device being provisioned, and (b) the initial software load that the device will run.

Final steps of device configuration may include the establishment of user/customer-specific keys. Examples of keys that might be provisioned by the final customer might include encryption keys that are used to secure customer data, or shared keys allowing all of a customer’s IoT devices and coordination-hubs to identify each other and communicate securely. The TPM distinguishes user and platform keys by the authority that controls their lifetime. Platform key lifetime is controlled by the platform manufacturer, and the manufacturer may choose to make their keys everlasting. The TPM provides additional capabilities to create keys for the device owner that only the owner can delete. If IoT devices enable this behavior, then the TPM supports user-controlled secure de-personalization of a device so that it can be safely sold or retired. [TPM2]

4.8 Maintaining Audit Logs

IoT devices will see increasing utilization as data sensors and we will find ourselves increasingly reliant on the data that they will produce. Since IoT devices communicate over the (untrusted) Internet, cryptography must be used to protect the reports and statements made by the devices.

The TPM can be used to sign device statements or can be used to create secure channels like SSL on which a stream of statements can be made. The TPM also incorporates a variety of more sophisticated secure signature technologies that can guard against other attacks on the network or the device itself. For example, TPMs include monotonic counters. A monotonic counter – as its name implies – counts up, but not down. An IoT device can incorporate a monotonic count-value into its reports to guard against both the replay and deletion of device statements.

TPMs also include a secure-clock: While there are some common implementation limitations on the behavior of the clock (for instance, whether it is always powered), including a TPM-clock measurement in a signed data report still protects against many attacks on the device or data stream.

Finally, the TPM in a device implementing measured boot also allows the identity of the software making a report to be included in a signed report. This capability is called attestation, and can be used to guard against old and buggy software operating under the control of an adversary imitating the reports made by new and bug-free versions.

In addition to online data reporting the TPM supports secure local logging of data and information: once more, the clock/timer and monotonic counters can be used to protect these reports.

4.9 Remote Manageability

A focus of the TPM specification is to define capabilities for the protection of secrets. In principle, any small unit of data can be protected using a TPM. In practice, the secrets we are talking about are usually keys, either symmetric or asymmetric. Institutions that deal with keys already have a management infrastructure in place for the management of these keys. There are many ways to perform key management. Often, these tools are centrally based. By the time key management reaches an end point, we are usually talking about a client of some sort and that client depends on some sort of protective mechanism to ensure the confidentiality of that secret while it is stored at rest on the device.

There are a few common methods for the key management client or user to access this protective mechanism.

- RSA's PKCS #11 standard is commonly used in the Linux world as a standard method for accessing services offered by an HSM for the protection of private keys tied to digital certificates. PKCS is also supported under Windows.
- Microsoft's Cryptographic API (CAPI) and successors do the same in Windows environments.
- Java's crypto library includes support for Cryptographic Service Providers (CSPs). These CSPs can provide access to HSMs for key protection.

That covers the problem of key management as it comes down the stack from the application that uses keys.

Coming up from the HSM (in this case a TPM), we have the following stack:

- The TPM specification defines an API that can be used to request protective services from the TPM. An entity can use this API to define a passphrase and access control rules that restrict access to a secret the TPM protects.
- TCG defines the TPM Software Stack, a middleware that abstracts the complexity of the TPM API. In the Windows world, a number of ISVs provide proprietary implementations of TSS, including a bridge that makes the TPM accessible through Windows CAPI. In Linux, IBM open-sourced an implementation of TSS for Linux called Trousers.
- For PKCS #11 users, the Open Source community includes several modules that bridge PKCS #11 to Trousers.

Using a bridge to either CAPI or PKCS #11, it is possible for app developers who know one or both of these interfaces to begin using a TPM to protect keys without actually knowing anything about how TPMs work. There are a number of CAPI bridges available in the market either for free (from PC vendors) or for fee. They are implemented as Cryptographic Service Providers (CSPs) for use with CAPI. For the Linux PKCS #11 world, there are several Open Source PKCS #11 to TSS bridges.

If the customer already has a Key Management System (KMS) that supports use of CAPI or PKCS #11 on end points, transition to using a TPM to provide hardware-based protection can be done by

- Installing a TPM-aware extension into Windows CAPI
- Installing Trousers and an open source PKCS #11 bridge module under Linux.

4.10 Securing Legacy Hardware

The Trusted Network Connect (TNC) architecture includes a specification designed to improve the security of legacy Industrial Control Systems (ICS): IF-MAP Metadata for ICS Security [MAP-ICS]. This specification is designed to work as part of the ISA 100 architecture designed by the International Society for Automation (ISA) for ICS security.

In this architecture, legacy ICS devices are organized into local enclaves called Characterized Control Domains (CCDs). CCDs are interconnected over an untrusted Backhaul Network using security gateways known as Backhaul Interfaces (BHIs). The BHIs establish a secure (encrypted and authenticated) Overlay Network on top of the Backhaul

Network. The BHIs further restrict which ICS devices can communicate with each other, based on configured policies. And the IF-MAP Metadata for ICS Security specification describes how BHIs are provisioned with the credentials and policies needed to make this system work smoothly and easily.

Of course, this architecture is not perfect. If attackers can compromise one ICS device, they may be able to spread their control to others. But the BHIs can prevent attackers on the untrusted Backhaul Network from accessing ICS devices in a CCD and they can monitor traffic between the ICS devices for suspicious behavior.

This gateway architecture need not be restricted to only ICS devices. It can have broader applicability in environments where vulnerable devices are collected into enclaves and protected by gateways, like in automotive, home automation and healthcare applications.

5. References

[ALGREG] Trusted Computing Group, “TCG Algorithm Registry”, Specification Version 01.22, https://www.trustedcomputinggroup.org/resources/tcg_algorithm_registry.

[IF-M] Trusted Computing Group, “IF-M: TLV Binding”, Specification Version 1.0, http://www.trustedcomputinggroup.org/resources/tnc_ifm_tlv_binding_specification, May 2014.

[IF-TNCCS] Trusted Computing Group, “IF-TNCCS: TLV Binding”, Specification Version 2.0, https://www.trustedcomputinggroup.org/resources/tnc_iftnccs_specification, May 2014.

[IF-TTLS] Trusted Computing Group, “IF-T: TLV Binding”, Specification Version 2.0, https://www.trustedcomputinggroup.org/resources/tnc_ift_binding_to_tls, February 2013.

[INTEGRITY] Trusted Computing Group, “Trusted Network Connect: Open Standards for Integrity-based Network Access Control”, https://www.trustedcomputinggroup.org/resources/trusted_network_connect_open_standards_for_integritybased_network_access_control, April 2011.

[OPAL] Storage Work Group Storage Security Subsystem Class: Opal, Version 2.00 Final, Revision 1.00, February 2012.

[PC-CLIENT] TCG PC Client Specific Platform Firmware Profile for TPM 2.0 Systems. In preparation.

[TNC-ARCH] Trusted Computing Group, “TNC Architecture for Interoperability”, Specification Version 1.5, http://www.trustedcomputinggroup.org/resources/tnc_architecture_for_interoperability_specification, May 2012.

[TPM-IDENTITY] TPM Keys for Platform Identity for TPM 1.2, http://www.trustedcomputinggroup.org/resources/tpm_keys_for_platform_identity_for_tpm_12, February 2, 2015. TPM2.0 version is in preparation

[TPM2]	Trusted Computing Group, "TNC Architecture for Interoperability", Specification Version 1.5, http://www.trustedcomputinggroup.org/resources/tnc_architecture_for_interoperability_specification, May 2012.
[TSS]	TSS System Level API and TPM Command Transmission Interface Specification, http://www.trustedcomputinggroup.org/resources/tss_system_level_api_and_tpm_command_transmission_interface_specification, and associated specifications.
[802.1AR]	Secure Device Identity, Dec 2009. http://www.ieee802.org/1/pages/802.1ar.html.
[800-147]	NIST SP 800-147 Basic Input/Output System (BIOS) Protection Guidelines, April 2011. http://csrc.nist.gov/publications/nistpubs/800-147/NIST-SP800-147-April2011.pdf.