TRUSTED® COMPUTING GROUP

REFERENCE

TCG Guidance for Securing Industrial Control Systems Using TCG Technology

Version 1.0
Revision 109
January 10, 2022

Contact: admin@trustedcomputinggroup.org

PUBLISHED

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 Industrial Work Group who participated in the development of this document:

NAME	AFFILIATION
Wael Ibrahim	American Express
Reed Hinkel	ARM Ltd.
Monty Wiseman	General Electric / Beyond Identity
Graeme Proudler	Graeme Proudler
Steve Hanna	Infineon
Matthew Areno	Intel
Sung Lee	Intel
David Challener	Johns Hopkins University, Applied Physics Lab
Dennis Mattoon	Microsoft
Raghupathy Krishnamurthy	Mocana
Wesly Alig	TechSolve
Joe Anderson	TechSolve
Radu Pavel	TechSolve
Jérôme Quévremont	Thales
Charles Omstead	U.S. Government
Trevor Hardcastle	Xilinx
Paul Levy	Xilinx

CONTENTS

1 SCOPE, AUDIENCE, AND PURPOSE

This document, the ***TCG Guidance for Securing Industrial Control Systems***, is part of the Trusted Computing Group's collection of Reference Documents, which are informative (non-normative) documents that provide advice and guidance. This document provides advice to architects and designers on where and how TCG technology and standards can best be used to secure industrial control systems.

1.1 Scope

The ***TCG Guidance for Securing Industrial Control Systems*** provides recommendations, detailed advice, and procedures relating to the use of TCG specifications to improve the security of industrial control systems. For a broader discussion of ICS security beyond just TCG standards, IEC 62443 [39] is a definitive source. This document is not normative, so the guidance contained here is not binding but rather advisory.

For the purposes of this document, the terms "industrial control systems" (ICS) and Operational Technology (OT) are considered to be synonymous with the term "Industrial Automation and Control Systems (IACS)" as defined in IEC 62443:

> *The term "Industrial Automation and Control Systems" (IACS), includes control systems used in manufacturing and processing plants and facilities, building environmental control systems, geographically dispersed operations such as utilities (i.e., electricity, gas, and water), pipelines and petroleum production and distribution facilities, and other industries and applications such as transportation networks, that use automated or remotely controlled or monitored assets.*[1]

The term ICS is used throughout this document instead of IACS because ICS is much more commonly used in industrial settings. Also, the commonly-used term "device" is employed here to indicate an ICS device such as a sensor or actuator or PLC (Programmable Logic Controller) instead of the term "component", which is used by IEC 62443. For the purposes of this document, ICS is synonymous with IACS and "device" is synonymous with "component."

Information Technology (IT) systems such as laptops, tablets, and servers may be used in an industrial environment and may even be considered part of an industrial control system or ICS. TCG has other documents [43], [44] that describe how to secure such IT systems. This document focuses instead on systems and issues that are specific to industrial environments. However, this document describes certain use cases that involve laptops, tablets, and servers when they are used in industrial applications.

ICS are increasingly connected to each other and even to the outside world, sometimes with TCP/IP and sometimes with legacy protocols such as MODBUS. While connectivity increases the attack surface of ICS, even completely disconnected ICS and those connected to "air-gapped" networks are vulnerable to attacks, as evidenced by Stuxnet and many other attacks on disconnected systems. For this reason, the scope of this document includes disconnected systems, as well as air-gapped systems, intermittently connected systems, and fully networked systems. IEC 62443 also includes all of these systems within its scope. Guidance on handling intermittent or non-existent connectivity is included in sections 5.21 and 6.21 below.

Physical security is largely a separate domain from cybersecurity. Therefore, the topic of physical security is generally beyond the scope of this document. However, sections 5.6 and 6.6 provide some information on potential physical attacks about industrial systems and their mitigations. Virtualization is also considered out of scope for this version of this document, although it may be considered in a future version.

1.2 Audience

Industrial security affects many stakeholders: Device Manufacturers, System Integrators, System Owners, Operators, Maintainers, Regulators, Auditors, and the public at large. Because all of these stakeholders are affected, they may all find some value in this document.

[1] IEC TS 62443-1-1 ed.1.0 "Copyright © 2009 IEC Geneva, Switzerland. www.iec.ch"

Security experts who are unfamiliar with industrial control systems can provide valuable assistance in securing industrial control systems so long as they understand that there are important differences between IT and industrial control systems architecture.

Within these stakeholders, the intended audience is architects and engineers responsible for or concerned with industrial security.

This document may not be comprehensible to all. Some technical knowledge is required to understand this document. Elementary security concepts are not explained here. Trusted computing concepts are explained briefly and references to more in-depth documents are provided. A fundamental understanding of the Trusted Platform Module Library Part 1 [3] and DICE [23] is assumed. Concepts relating to industrial automation and industrial devices are explained briefly and references to more in-depth documents are provided.

1.3 Purpose

Maintaining the security, safety, and proper functioning of industrial control systems is essential. Failure of these systems can result in property damage, injury, or even death. While industrial control systems are designed with fail-safe measures to ensure that the failure of one or two devices will not result in unsafe conditions, cyber attacks may be purposely designed to overcome the fail-safes. Functional safety cannot be ensured unless the system is secure.

Trusted computing can play a critical role in ensuring that industrial control systems include protections against exploitation of software vulnerabilities and even against physical attacks. By using TCG specifications, product implementers and customers can employ sophisticated protections without requiring custom hardware. Open standards from TCG ease adoption and interoperability because they are widely supported and compatible across products from many vendors.

Cyber attacks on industrial systems are not a theoretical problem. Recent years have seen a growing trend of such attacks, such as the 2014 attack on a German steel mill [47] which resulted in “massive damage” to equipment and the 2015 attack on the Ukrainian power grid [45] which resulted in a power outage of several hours for more than 200,000 people. The need to improve cyber protection is exemplified by a US Presidential Executive Order on “Improving the Nation's Cybersecurity” [50].

The purpose of this document is to describe how trusted computing can be used to prevent cyber attacks on industrial control systems from succeeding. With this purpose in mind, this document describes the security challenges faced by industrial control systems, lays out typical use cases for securing those systems, and describes how trusted computing can be used to implement these use cases.

A secondary purpose of this document is to help trusted computing experts understand and better address industrial uses of trusted computing technology. When TCG becomes aware of a new domain that could benefit from trusted computing technology, TCG writes a guidance document (like this one) to record issues arising from that domain.

2 INTRODUCTION TO INDUSTRIAL CONTROL SYSTEMS SECURITY

2.1 Industrial Control Systems Reference Model

Figure 1 shows a simplified reference model often used for Industrial Control Systems.

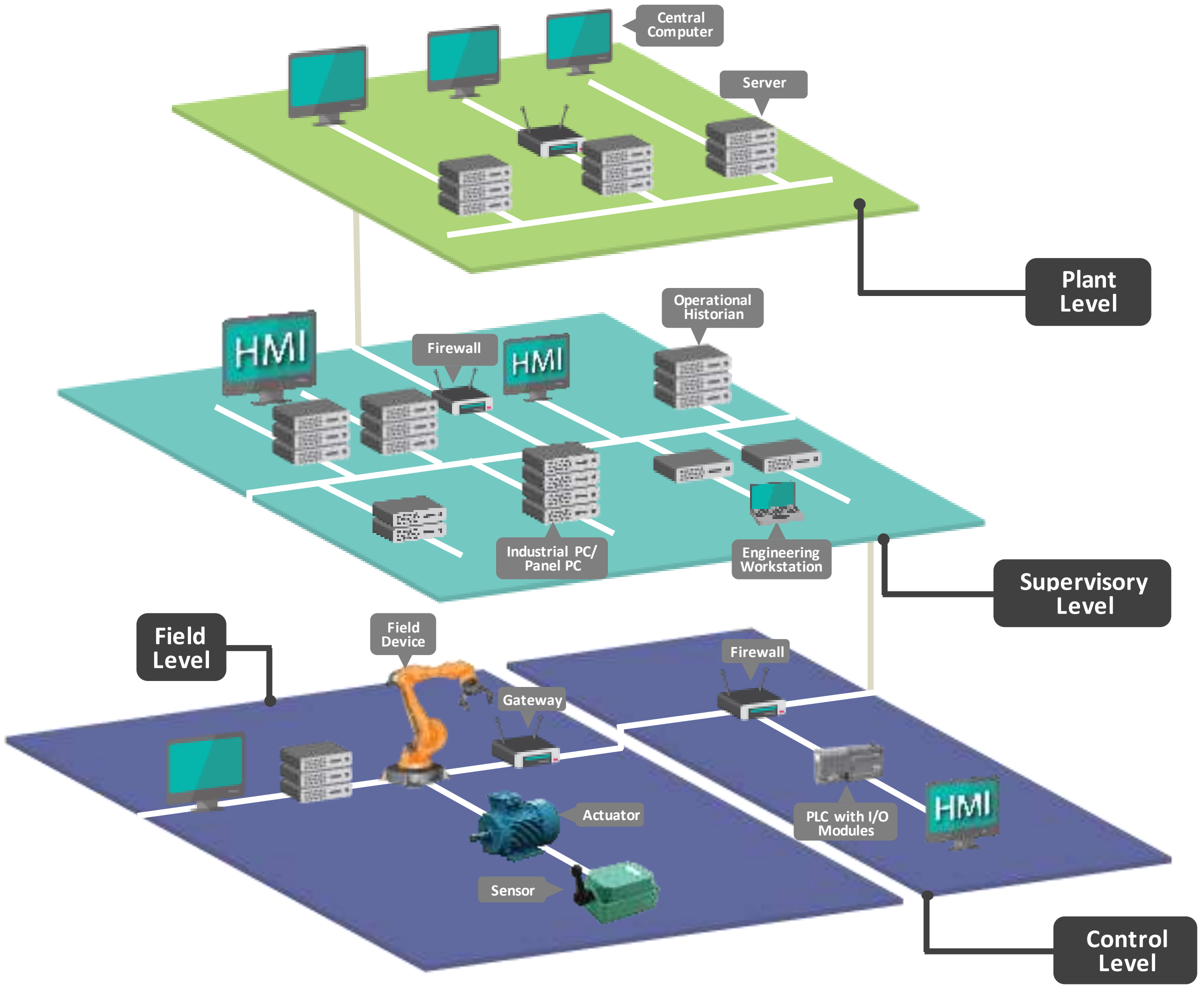

Figure 1 - Simplified Industrial Control Systems Reference Model

2.1.1 Overview

2.1.1.1 Field Level

The lowest level in this model is the Field level, which is composed of sensors, actuators, local controllers, and network elements such as switches. Composite devices such as a robot arm may combine several of these functions into one system within the Field level, providing an efficiently integrated function. Industrial Control Systems are often actually systems of systems, several layers deep.

2.1.1.2 Control Level

The next level up in the model is the Control level, where Field level devices and systems are directly controlled. The Control level includes devices such as PLCs (Programmable Logic Controllers), Human Machine Interface (HMI) displays and inputs such as keyboards, and more network elements such as switches. PLCs are programmable systems designed to respond in real time to conditions at the Field level. For example, a sensor in a vessel in the Field Level may send a low temperature message to a PLC that may cause the logic within the PLC to turn on a heater.

2.1.1.3 Supervisory Level

Above the Control level is the Supervisory level. At this level, operators use SCADA (Supervisory Control And Data Acquisition) systems such as Engineering Workstations to set and adjust the system's logic. The SCADA systems create the process control logic (the custom code written to handle the particular inputs and outputs connected to the PLC) and perform debugging and maintenance. HMIs may also be used at this level to monitor and adjust systems at levels within the Control and Field level. SCADA servers at this level store relevant data and Historians maintain time sequences of sensor data. Systems at the Supervisory level are not generally expected to respond in real time.

2.1.1.4 Plant Level

At the top of the model is the Plant level, where business functions such as customer service reside. At this level, conventional IT equipment is used. In today's world of Industrial IoT, additional levels such as a Cloud level may also be incorporated, either atop or alongside the Supervisory or Plant level.

2.1.2 Referenced List of ICS Devices

2.1.2.1 Within Scope

The following is a list of ICS devices within scope of this document:

1. Sensors and actuators
    - These directly sense and control physical processes or equipment. These are typically connected directly to the PLCs.
2. PLC (Programmable Logic Controller)
    - The PLCs implement the closed-loop logic based on sensor information, following ladder logic and commands from the HMI to alter actuators.
3. RTU (Remote Terminal Unit)
    - The RTUs provide remote functions such as sensing, and actuation controlled by a central server.
4. HMI (Human Machine Interface)
    - Provides human operators and automated processes the means to change the settings within a PLC based on a set of requirements. These controls and settings must be within the limits set by the PLC's process control ladder logic (i.e., within the set points).
5. Engineering Workstation
    - Used by technicians to program and adjust the system's control logic. The technician creates the ladder logic which is converted to object code and sent to the PLCs. Engineering Workstations may also communicate directly to PLCs for monitoring, debugging, and maintenance. They may also be used to update PLC firmware.

    NOTE: In practice, HMI and Engineering Workstation functions are often combined within a single device with different functions enabled for different users. This document treats the HMI and the Engineering Workstation as distinct devices.
6. Network Equipment
    - Network Equipment in industrial systems routes the network traffic within and between the Field, Control, and Supervisory Level. In addition to routing, switching, and other basic connectivity functions, network equipment such as ICS gateways are commonly used to implement the Zones and Conduits described in IEC 62443. Protocol conversion (e.g., serial-to-Ethernet, MODBUS-to-IP,

etc.) is an essential function but can lead to security issues as identity is masked or single points of failure are created. Other network security functions such as firewalls, VPN (Virtual Private Network), and IDS (Intrusion Detection System) are also considered to be Network Equipment.

7. Industrial PC
   - Industrial PCs may be used to implement a variety of other diagnostic or management functions.
8. Handheld devices
   - Handheld devices may be used to directly provision, change logic, or diagnose devices typically within the Field Level.

2.1.2.2 Out of Scope

Devices at the Plant Level are out of scope for this document as they are IT systems that already have other TCG guidance and specifications which govern them. Typical devices found at this level include:

1. PCs and general-purpose Workstations
2. Networking Equipment (Often called IT Equipment)
3. IT Servers

2.2 What's Different About Industrial Control Systems?

Industrial control systems and the devices that make them up are generally fixed-function devices, designed to perform a specific task. This contrasts with IT systems, which are general-purpose data transformation devices designed to perform many different tasks over time, depending on the software that is loaded onto them.

Another substantial difference between Industrial Control Systems and IT systems is the emphasis on safety in ICS. Because ICS control complex and often dangerous physical systems, safety is always the highest priority in designing ICS.

Reliability and uptime are also high priorities for ICS, as the cost of downtime may be quite high. Because of this need for reliability and also because of the high capital costs for acquiring and installing ICS, these systems are designed to operate reliably for decades or longer. This presents unique challenges for cybersecurity where new attacks are discovered every day. Cybersecurity defenses depend on regular updates to keep up with new attacks. Even fundamentals such as cryptographic algorithms generally need to be upgraded every few years, at least to increase key length. ICS operators are reluctant to shut down systems for upgrades due to the downtime incurred. And the different lifecycles for IT and ICS can lead to outrageously obsolete IT systems being used in ICS environments. This problem of legacy systems must be considered and planned for.

Finally, many ICS must provide real-time response to events, whereas IT systems rarely face hard real-time constraints. For example, an ICS may need to stop a feed pump when a pressure sensor reaches a certain reading. Failure to respond within a certain time period (e.g., 10 milliseconds) could result in a chemical spill or even an explosion. Due to these real-time concerns and to potential safety issues, any update to the ICS generally must be tested and certified by the manufacturer to ensure that real-time constraints will still be met.

2.3 Realities of Industrial Control Systems

Real life rarely matches the theoretical purity of the ICS Reference Model in Figure 1. Here are some examples where reality often diverges from theory.

The separate levels shown in the ICS Reference Model are often intermixed. For example, devices at the Control level are often intermixed with those at the Field level. That's not a bad thing. Placing a PLC physically close to the devices it controls and on the same network can improve performance and also allow physical proximity between the PLC and the device being controlled. This can ease the job of an operator sent to debug a problem.

ICS are often used at large companies with complex organizational structures and outsourced functions. A single ICS may span multiple organizations, administrative domains, and physical locations. For example, large enterprises may have an IT department that acts as an internal service provider, offering network connectivity to multiple departments and groups. Alternatively, networking functions and IT administration may have been fully or partially outsourced to another company. In this context, several administrative domains or even several companies may share use of and even control of a single device.

Due to the widely distributed nature of many ICS, the Maintainer of a given device often will not have direct physical connectivity to the device, and will require authenticated remote access to carry out management functions on the device.

2.4 Why Secure Industrial Control Systems?

The performance and safety of an industrial control system and its environment depend on accurate information flow. This flow, which may use a variety of control and feedback means, requires protection from corruption (intentional or unintentional interference or alterations) to the extent specified in the manufacturing scope.

Protection from interference and alteration by external sources, whether nefarious and accidental, needs to be explicitly considered and implemented in system designs. Designers should not make optimistic assumptions but must plan for a worst case scenario involving skilled, malicious attacks.

2.5 Stakeholders

Many stakeholders are involved in the use and secure deployment of Industrial Control Systems, forming a complex ecosystem:

- Device Manufacturer
    - Device Manufacturers make and sell (or lease) devices that are assembled into Industrial Control Systems.
    - The Device Manufacturer is usually responsible for supplying a complete device, including the hardware, software, and supporting services such as software updates.
    - Device Manufacturers usually depend on multiple suppliers of hardware, software, and services. These suppliers may create critical modules such as a power supply or operating system or provide critical services such as final assembly of the device.
- TPM (Trusted Platform Module) Manufacturer
    - The TPM Manufacturer creates and initializes the TPM and supplies the TPM to the Device Manufacturer.
    - TPM Manufacturers usually equip each TPM with a unique Endorsement Key (EK) and EK Certificate.
- System Integrator
    - The System Integrator integrates multiple Devices into an Industrial Control System in order to solve a problem for the System Owner.
    - The System Integrator may have experts in areas such as security, enabling it to offer specialized services.

  - The System Integrator may provide to the System Owner a set of instructions for the operation of the Industrial Control Systems, including policies and procedures.

- System Owner
  - The System Owner owns an Industrial Control System and uses it for some purpose, such as running a manufacturing plant.
  - A single System Owner may own many Industrial Control Systems and use these systems in one or more facilities, either in isolation or in coordination.
  - In some cases, the System Owner may lease or rent the Industrial Control System instead of owning it. For the purposes of this document, this distinction has little impact.
- Operator
  - The Operator is a person who operates the Industrial Control System or some aspect of it. Generally, the Operator is employed by the System Owner or a subcontractor of the System Owner.
  - While Operators should be trained to comply with policies and procedures, some Operators may not do so. Whatever the cause of this non-compliance (ignorance, laziness, forgetfulness, sabotage, etc.), safety must be protected.
- Maintainer
  - The Maintainer is a person who maintains the Industrial Control System or some aspect of it. The Maintainer is generally employed by the System Owner or a subcontractor of the System Owner.
  - Maintainers are generally more trained and trusted than Operators. Because of their responsibilities, they must have greater privilege or access. For example, they might have the ability to update installed software. This greater access means that the consequences of error or malfeasance by a Maintainer is potentially greater than for an Operator. Still, safety systems should be resilient so that safety is ensured even if a Maintainer makes an error or performs a deliberately malicious action.
- Auditor
  - An Auditor is an organization hired to review operation of an Industrial Control System, especially for compliance with regulations.
- Regulator
  - A Regulator is an organization that establishes regulations and requirements regarding operation of an Industrial Control System and oversees compliance.

Other stakeholders (citizens, neighbors, shareholders, etc.) may have an interest in preserving the safety and security of the Industrial Control System but they have no direct impact on it so they are not listed in this section.

2.6 Summary of Recommendations

While section 6 of this document gives specific recommendations for the use of TCG technology in enhancing the security of industrial equipment, the guidance can be summarized as follows:

- Devices should use a TPM or DICE as a hardware-based root of trust.
- Cryptographic Device Identity based on IEEE 802.1AR, using the TPM or DICE to protect keys, can provide a reliable way to identify remote devices for purposes such as counterfeit detection, device management, configuration and authentication.

- The TPM or DICE can be used to protect confidential data such as VPN keys in industrial equipment.
- Secure software update is a necessity in today's world of rapidly changing threats. Attestation using a TPM or DICE can offer assurance as to the integrity of software running on industrial equipment.
- Use of the TPM's random number generator can enhance the strength of cryptographic protocols by providing additional entropy for cryptographic keys. Without strong entropy (highly unpredictable randomness), even the strongest cryptographic protocol can be weakened by low entropy keys.
- Novel approaches such as Secure Zero Touch Provisioning and Integrity-Protected History should be considered.
- Risks due to legacy systems and disconnected operation should be addressed to preserve security and uptime.

3 PRINCIPLES

The principles listed in this section are considered best practices in the world of industrial cybersecurity. In this section, the term Data refers to 1) programmatic content used to operate/transform a second stream of content and to 2) the second stream of content itself.

- All Data Transportation will be authenticated by secure means. Authenticated data streams are explicitly verified as complete in the context prior to consumption.
- All Data Transportation will be acknowledged in a secure manner if acknowledgement is required.
- All Data Transformation processes are verified by independent secure means verified by an authorized chain of trust. Code used for operating on data will be verified by independent means prior to execution.
- Operators who can perform Data Transportation or Data Transformation will be authenticated prior to accessing the system using state of the art security methods.
- The strength of security means and methods used will be explicitly specified.

4 BUILDING BLOCKS

NOTE: This section was copied from the TCG Guidance for Securing Network Equipment [51].

4.1 Chain of Trust

The security of the devices that make up Industrial Control Systems is critically dependent on the integrity of the software running on the device. Software integrity is usually assured with a Chain-of-Trust model, with each stage during startup checking the next stage before execution, as shown in Figure 2.

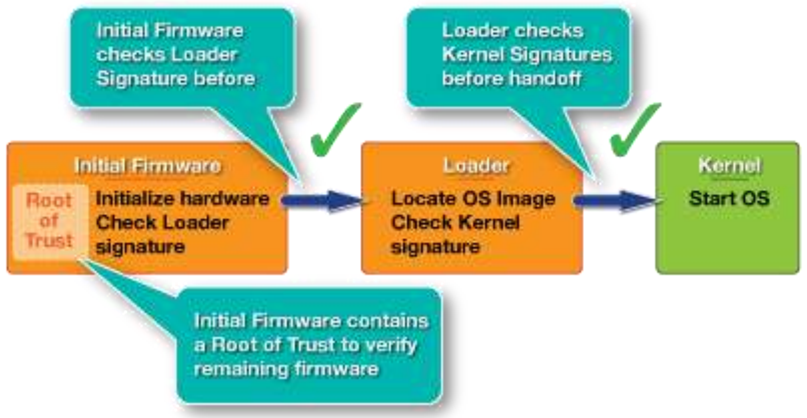
Initial Firmware checks Loader Signature before
Loader checks Kernel Signatures before handoff
Initial Firmware
Root of Trust
Initialize hardware Check Loader signature
Loader
Locate OS Image Check Kernel signature
Kernel
Start OS
Initial Firmware contains a Root of Trust to verify remaining firmware

Figure 2 - Chain of Trust

In the Chain-of-Trust model, shown in Figure 2, the Root of Trust is established first. In this example, a Root of Trust for Verification (RTV) is illustrated but a Root of Trust for Measurement (RTM) could be employed in addition to or instead of the RTV. Since nothing is executed before the RTV, it must be trusted implicitly. The RTV validates the remaining Initial Firmware ("Firmware" for short) before handoff and execution. Then the Initial Firmware executes and validates the next stage, in this example, the Loader. Then the Loader executes and validates the Kernel. In this example the Kernel is the final stage executed. Note that the Chain-of-Trust mode is extensible to as many stages as needed for the system, not just those listed in this example.

With a TPM in the system, measurements can be made of each stage in the Chain-of-Trust model and extended into PCRs (Platform Configuration Register) within the TPM. Beyond providing secure storage for the measurements used in the Chain-of-Trust, the PCRs allow for periodic run-time integrity checks of the system. Often, different PCRs are used for different stages in the boot sequence. Mapping of the PCRs to each of the stages is out of scope for this document. Such details are provided in TCG Platform Specifications such as the PC Client Platform Firmware Profile [48].

Underpinning essentially all of the security mechanisms outlined in this document is a reliable Root of Trust. The Root of Trust is typically implemented so it's available to deliver the very first instructions to the device for execution after reset. The Root of Trust may have several functions, among which are performing the first measurement (RTM) and ensuring that the Initial Firmware has not changed without authorization (RTV).

Although TCG technology is not needed to secure the root of trust, NIST SP 800-155 [24] does specify the use of a TPM as part of a comprehensive software integrity strategy, linking to the Health Check features covered in Section 6.12 of this document.

4.2 Secure Boot vs. Measured Boot

There are generally two approaches to ensuring that code running on a device is authorized, and has not been subject to illicit modification.

- *Secure Boot* (also known as *Verified Boot*) is a process by which each stage of the boot process checks a cryptographic signature on the next stage before executing it. In a typical system, there might be a BIOS that does a check of an OS Loader's signature, which in turn might do a check of the OS kernel before launching it.
- *Measured Boot* (also known as *Attested Boot* or *Authenticated Boot*) is a process by which each stage of the boot "measures" or computes and stores the hash of software or firmware in the next stage prior to launching it.

Secure and Measured Boot can be extended to run-time software through mechanisms such as the *Linux Integrity Measurement Architecture* [16]. See Section 6.12.1 of this document for more on the use of IMA.

Secure Boot is a subset of an overall Secure Computing architecture, which dictates a platform's behavior, as opposed to Trusted Computing, which enables the Maintainer to deduce a platform's behavior. While implementations of secure boot are relatively common, complete Secure Computing or Trusted Computing environments are still difficult to design and implement.

Secure Boot and Measured Boot have some commonality and some differences:

- While Measured Boot depends critically on the TPM to safely store measurements[2] (See Section 6.12), Secure Boot can be implemented without the use of TPM technology.
- Both techniques rely on a Root of Trust in the first code executed, although Secure boot could be used to reliably instantiate the Roots of Trust required by Trusted Computing.
- While measured boot won't stop a corrupted system from starting, it is able to vouch for the state of a broader range of items in the boot path (e.g. configuration files that might impact system security, in addition to executables).
- Secure Boot and Measured Boot have different risk profiles. Secure Boot can convert an innocent mistake such as a key mismatch into an industrial system outage by refusing to start, while Measured Boot can allow a corrupted device to continue running at the risk of system damage.
- "Sealing", or encrypting data such that it can only be decrypted when a platform is in a known measured state, is a cost-effective way to protect data-at-rest, even with secure computing.
- A single system could reasonably execute both Secure Boot and Measured Boot at the same time. Measured Boot and attestation enhance confidence in a device's behavior even when it's not practical to cryptographically sign each and every object accessed as the system starts up.[3]

4.3 802.1AR Device Identification

[This Informative summary of 802.1AR appeared first in *TPM Keys for Platform Identity for TPM 1.2* [5]; it has been edited in this section for wording and for applicability to TPM 2.0]

The IEEE 802.1 working group defined IEEE 802.1AR, which was approved on 22 December 2009 [1] with an updated version [2] approved on 14 June 2018. This standard defines a per-device unique identity installed at manufacturing time and used subsequently in device-to-device authentication exchanges. This standards-based device identity can also be coupled in multiple ways with user identification.

[2] A *measurement* is the cryptographic hash of an object such as an executable file.
[3] For example, a Device Manufacturer can't sign a customer-mutable configuration file, but the Administrator may know exactly what should be in the file on a fielded machine. Measurement can report a fingerprint of the contents without requiring a signature on each file.

The 802.1AR standard defines a secure device identifier (DevID) as "a cryptographic identity that is bound to a device and used to assert the device's identity". It further specifies:

- the DevID is an X.509 credential
- globally unique per-device identifiers and the cryptographic binding of a device to its identifiers
- the relationship between an initial identity installed during manufacturing and subsequent locally-significant identities installed by the Administrator
- interfaces and methods for use of DevIDs with existing, and new, provisioning and authentication protocols

The initially installed identity is defined as an IDevID ("I" for initial) and subsequently locally defined identities are LDevIDs ("L" for local). An IDevID will be created at manufacturing time and is proof that the device holding the corresponding private key has been manufactured by a certain manufacturer. An LDevID is created on the Administrator's premises and is proof that the device holding the corresponding private key is owned by a certain enterprise (or individual).

There is another dimension to the IDevID/ LDevID discussion with TPMs prior to and including version 1.2[4]. In order to create an IDevID, the Device Manufacturer needs to take ownership of the TPM. However, commands that change ownership make all previously created keys unusable. Section 6.1.2.4 of this document outlines precautions that must be taken by the Device Manufacturer to ensure sure that the IDevID cannot easily be made unusable by the TPM. More on this topic can be found in *TPM Keys for Platform Identity for TPM 1.2* [5], Section 6. As mentioned earlier, this is not an issue for TPM 2.0.

The 802.1AR specification DevID module (covering IDevID and LDevID) contains a service interface, storage holding a DevID secret and certificate, secure hashing functions, a random number generator (RNG) and asymmetric cryptography functions. These functions exist in the TPM and calls by middleware (such as the TPM Software Stack (TSS) or the TPM API) can be used to satisfy interface requirements outlined in 802.1AR.

The IEEE Standard 802.1AR can be used together with TPM-based keys and certificates. The TPM acts partly as the Secure Device Identifier Module (DevID Module) which the standard defines as "a logical security component that will secure, store and operate on one or more DevID Secret(s) [(private key)] and associated DevID credentials". The document *TPM Keys for Platform Identity for TPM 1.2* [5] addresses usage of the TPM after provisioning has occurred and leverages TCG-specified X.509 certificate extensions to prove TPM residency of the keys, using the Subject Key Attestation Evidence (SKAE) extension [9]. Future TCG specifications may detail how TPM 2.0 key attributes and residency are proven for use as Device Identifiers.

4.4 Hierarchies

TPM 2.0 provides three independent key hierarchies: a "platform hierarchy" for platform protection, independent of specific users; an "endorsement hierarchy" for exposing information; related to identity and a "storage hierarchy" for cryptographic usage related to specific users or applications.

[4] This aspect does not arise with TPM 2.0.

5 USE CASES

5.1 Device Identity

Providing reliable remotely-verifiable device identity for each industrial device is a prerequisite for most use cases related to securing industrial devices. This section describes a few use cases that demonstrate the value of device identity for industrial devices and the variety of ways that device identity can be implemented and used.

Before describing these use cases, it’s helpful to distinguish two kinds of device identity: Device Manufacturer identity and System Owner identity. The Manufacturer identity for a particular device is established, configured and managed by the Device Manufacturer, although it can also be used (e.g., verified) by the device owner. The Owner identity for a device is established by the device Maintainer and is generally used only by the Maintainer. Manufacturer identity is generally[5] unique across all products from that manufacturer (e.g., a model number and serial number) while Owner identity may be unique only within the Maintainer’s domain. Manufacturer identity is defined as either TCG Platform Certificates or Initial Device ID in IEEE 802.1AR [2] (See Section 6.1) and TPM AKs or Local Device ID, respectively.

For the purposes of this section, device identity is implemented using a private key kept secret by the device and a certificate associated with that key, issued to the device by a Certification Authority (CA). The use of asymmetric cryptography and X.509 certificates enables easy integration with existing network protocols such as OPC-UA and TLS.

5.1.1 Manufacturer Device Identity and Counterfeit Protection

Both Device Manufacturers and System Owners need to verify the authenticity of their devices, determining whether they are “counterfeit” (made by an unauthorized party or in an unauthorized manner) or “authentic” (made by authorized parties in an authorized manner).

Business Drivers: To achieve reliable operation for end-user customers, System Owners need to be assured that they are using authentic devices from the expected manufacturers. At the same time, Device Manufacturers want to reduce lost sales, protect their brand, prevent returns and warranty repairs of counterfeit devices. Distributors want to detect counterfeit devices before they are resold. Auditors want to verify that industrial operators meet compliance regulations.

5.1.2 Identity for Network Access

Business Drivers: ICS Operators often wish to ensure that only authorized devices are connected to their networks. This is true for ICS but also for some IT systems, as industrial operations extend to the cloud and external data centers. Further, remote identification of industrial devices allows for remote deployment and provisioning. Today, deployed devices are often manually provisioned by trusted and skilled personnel. Using a Manufacturer identity, a Maintainer can remotely provision communication and other device-specific keys into industrial devices.

5.2 Access Control

Business Drivers: During the life cycle of industrial control systems (procurement, installation, test, etc.), one of the most important phases is operation. The operational phase of a product’s lifecycle is the primary motivation for its creation. That said, the value of a system is directly related to its ability to operate to its design specifications.

Access control is essential to maintain the integrity of the device and the system. Only authorized Operators and devices should be permitted to use or connect to industrial devices (“Device Access Control”). Further, only authorized devices should connect to networks (“Network Access Control”) and only authorized parties should have

[5] The Manufacturer and Maintainer are responsible for managing uniqueness in their respective name spaces, but typically these identifiers will not be ‘cryptographically strong’; the public key associated with the device identity can also be used as a unique identifier when there is a strong requirement for uniqueness.

access to sensitive data ("Data Access Control"). Thus, several kinds of access control are commonly used in industrial systems.

5.3 Zero Touch Provisioning

Business Drivers: When an industrial device fails, operators must replace that device quickly without compromising safety, security, or reliability. For safety-critical devices, redundancy or fail-safe mechanisms may be employed to ensure safety. However, timely and efficient replacement of failed devices is still important to minimize disruption and cost.

In these cases, Zero Touch Provisioning (ZTP) may be desirable. ZTP is a mechanism that automatically configures a newly installed device when it is connected and powered on. To ensure safe and secure operation, this Zero Touch Provisioning process must be conducted securely.

Because the result of the Zero Touch Provisioning process may be the installation of credentials onto a device, thus permitting the device to gain access to an operational network, careful design is required to ensure that only authorized devices are able to complete the ZTP process.

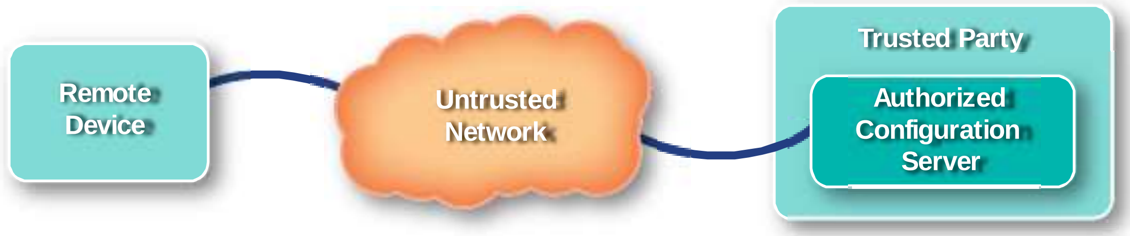

Figure 3 - Zero Touch Provisioning Architecture

Part of this use case may require proof that the device has been installed by an authorized operator, perhaps through the use of out-of-band authorization information.

5.4 Securing Secrets

Business Drivers: Industrial Control Systems often contain sensitive information such as intellectual property (software, design files, etc.), operational historian logs, and cryptographic keys (e.g., shared secrets, passwords, VPN keys, SSL/TLS keys, OPC UA keys, and stored data encryption keys). Disclosure of these secrets could result in disclosure of confidential intellectual property (such as manufacturing processes or design elements) and privacy-sensitive information or even enable malicious tampering with the system. Industrial control operators must protect these secrets against disclosure to keep their systems secure and reliable and also to meet regulatory or customer requirements for confidentiality and privacy. This protection must include protection for data at rest, data in motion, and sometimes data in use. Physical attacks are sometimes of concern, as described in section 5.6.

5.5 Protection of Software and Configuration Data

Business Drivers: Industrial Control Systems usually require configuration, often involving many parameters stored in a variety of files. The System Owner may wish to retain control over changes to the configuration of a device, with the goal of ensuring that the System Owner's processes are protected from unauthorized configuration changes. Further, the System Owner may wish to keep the configuration confidential so that neither the Device Manufacturer nor the Operator can see the configuration. This is especially important when the configuration includes confidential formulations and manufacturing details. The protections just described may apply to software equally.

Device Manufacturers or other parties may have some software and/or configuration files on the device that they need to keep confidential and/or integrity protected. For example, the Manufacturer may need to track the Periodic

Maintenance Interval (PMI), which tracks when certain maintenance tasks have been performed and how they were performed. Another example is the software used to perform certain optimized tasks, which the Manufacturer would not like to have disclosed to their competitors. A third example is that unauthorized modifications to critical software or configuration could take the system outside its design parameters and lead to safety issues.

5.6 Physical Attacks

Business Drivers: The varied and often unusual physical locations of Industrial Control Systems make them especially susceptible to a number of potential physical attacks. Traditional computer networks consist of clusters of computers and electronics interconnected via wired or wireless communication mediums. The clusters are often physically co-located and managed by on-site personnel. In contrast, Industrial Control Systems may or may not maintain constant communications with other systems, may be completely isolated geographically, and may not be protected physically from physical access by either malicious or non-malicious individuals.

In this context, it is critical for both Device Manufacturers and System Owners to recognize and understand the potential for physical attacks against their systems. For instance, isolated systems with little or no restrictions of physical access (i.e., pipelines, oil derrick, etc.) may be maliciously altered or replaced in order to create an attack on more physically secured systems. Individuals within an organization may be manipulated into utilizing their physical access to systems in order to compromise or impede the performance of critical systems. It therefore becomes necessary for security designers to consider physical attacks as a potential pathway into their most critical systems and design countermeasures that will try to prevent as many attacks as possible.

5.7 Licensing for Feature Authorization and Product Differentiation

Business Drivers: One of the attractive features of Industrial IoT systems is the ability to add features to industrial devices over time. In some cases, these extra features may be supplied free of charge to all. In others, they may be supplied only to customers who subscribe to software updates or only to customers who have purchased a particular new feature. The revenue from subscriptions and feature sales can fund long-term support and updates for device software, a feature highly desirable from a security standpoint. This use case is not intended to cover pay-per-use or *-as-a-service use cases, which are covered in section 5.8.

5.8 Equipment-As-A-Service

Business Drivers: Industrial IoT enables new business models such as Equipment-as-a-Service (EaaS). With an EaaS model, industrial devices or systems move from a purchase or lease business model to a pay-per-use model. Users pay according to the amount that they have used the device or system. For example, a company that is using a drill press or a laser cutter may receive a monthly bill based on the number of drilling or cutting operations done in the last month. A company using an air compressor may pay based on the number of cubic meters of compressed air used.

An EaaS business model permits the device or system user to pay for only the operations that they need. As production grows or shrinks, the associated costs grow or shrink as well. Contrast this to an equipment purchase business model where the user incurs a large one-time cost or an equipment lease business model where the user pays a fixed monthly fee.

Clearly, an EaaS user has an incentive to turn back the usage counter so that they are charged for fewer operations than they actually performed. The user may also have unrestricted and unmonitored physical access to the device that enables such tampering. However, successful rollback of the usage counter undermines the entire EaaS business model and must be prevented or made uneconomical. Otherwise, the model cannot succeed.

5.9 Counterfeit Detection

Business Drivers: Counterfeit products are a widespread danger to industrial customers. As procurement has moved to the Internet and new online shops have arisen, customers find it hard to determine whether they are buying authentic PLCs, sensors, batteries, bearings, seals, and other products. Buying a counterfeit part or supply can result in leaking seals, battery fires, damaged devices, and other risks. The risks of buying a counterfeit PLC or other control element can be even greater as counterfeit devices may include design elements (e.g., software or hardware) that are malicious and virtually impossible to detect.

Customers want to know when they're getting brand-name products or counterfeits so that they can manage their costs and risks. Manufacturers want to protect their brand and ensure customer safety and satisfaction.

5.10 Remote Device Management

Business Drivers: System Owners with a large number of devices often want to manage those devices remotely and rapidly, including the ability to monitor devices and reconfigure them dynamically. Remote management and reconfiguration are especially important in modern manufacturing systems that aim to respond quickly to customer demand such as those implementing "Lot Size 1" (the ability to manufacture goods to custom specifications in a fully automated manner).

Remote device management encompasses both normal operator adjustments such as changing parameters within defined setpoints and more fundamental changes such as changing ladder logic. Stronger security protections are needed for more fundamental changes as they are riskier.

5.11 Software Inventory

Business Drivers: Industrial systems include a wide variety of software. Operating systems vary from general-purpose operating systems on Industrial PCs to embedded systems with a real-time OS. Some small embedded systems have no OS at all. Application software varies from GUI (Graphical User Interface) applications to ladder-logic interpreters to hard-coded routines. Ladder logic and similar user-defined programming are another important layer of software.

Software updates are rare due to concerns about reliability and downtime, the need to retest systems after software updates, and the complex and often manual task of patching industrial systems software. But unpatched systems may contain hundreds of known vulnerabilities that must be managed.

To properly secure industrial control systems, Maintainers or whoever is managing the security of the systems (integrators, manufacturers, etc.) must be able to determine with confidence which software packages and versions are running on each system. Then they can make a rational decision about how to update software or put in place countermeasures to manage known vulnerabilities. Automating this process can save money and improve security by making it easier to identify and address problems.

5.12 Attestation of Boot Integrity for Devices ("Health Check")

Business Drivers: Business operations that use industrial control systems depend on their correct operation. Some attackers have been able to compromise the operation of industrial control systems by replacing or altering their software or firmware. [45][46] Therefore, verification of the software and firmware during the boot sequence is best practice for critical systems.

5.13 Attestation of Load-Time and Run-Time Integrity of Code and Configuration on Devices

Business Drivers: Correct operation of devices that use software to implement processes is critical to successful operation of systems depending upon those devices. There are two accepted questions of compliance for correct operation of devices that use software. The first question is "was the device loaded with all of the authenticated software required for correct operation?" The second question is, "has anything or anyone corrupted this software or the configuration state during normal operation?"

5.14 Inventory of Composite Devices

Business Drivers: Many industrial control systems use modular hardware, including one or more processor modules and an extensible set of I/O modules. The behavior of the industrial control system is based upon the composite behavior of the individual modules.[6] The security of the industrial control system is therefore only accurately represented by a composite measure that includes the security of its modules. Any change to one of these modules or even an undetected exchange of modules will modify the behavior of the system and may cause failure and risk. A secure inventory allows determination of the modules used and their state.

Some industrial control systems allow modules to be replaced without triggering a complete system restart (often called ‘hot swap’); for these devices, system-level reboots may be very rare, and the system’s security posture must be re-evaluated every time an individual module is inserted or removed from the system.

5.15 Integrity-Protected History

Business Drivers: Industrial systems often include an operational historian function and sometimes audit logs as well. These operational historians and audit logs gather data in time series from a variety of sources throughout the industrial system into a central location. Once gathered and stored, the data can be used for a variety of business purposes, such as looking for trends and correlations, optimizing production, root cause failure analysis, auditing, etc.

Operational history can provide evidence of error, inefficiency, malfeasance or attack. Therefore, external attackers and insiders may all be motivated to tamper with this history. Such attempts must be thwarted to preserve efficiency, integrity, security, and safety.

5.16 Entropy Generation

Business Drivers: Businesses and governments need to ensure that data at rest (stored data) and data in motion (traffic crossing an untrusted network) are reliably protected by strong cryptographic keys.

Most secure networking protocols such as TLS, DTLS, SSH, and IPsec need cryptographic-quality random numbers to avoid the generation of predictable session keys and nonces. If attackers can predict keys and nonces, they can decrypt encrypted data, forge signatures, and defeat many other security measures.

In addition, the TCP stack for Industrial Control Systems should use high quality entropy (truly random numbers) for the TCP window starting point as well as in the selection of ephemeral ports. These help to mitigate SYN and RST attacks against the device.

Industrial Control Systems also often requires high entropy random numbers to generate Cryptographic Security Parameters (CSPs) such as asymmetric key pairs.

Entropy sources used in Industrial Control Systems may have to comply with standards such as *NIST Special Publication 800-90 A/B/C* [12] or *BSI AIS-31, A Proposal for Functionality Classes for Random Number Generators* [13] or *ISO/IEC 18031* [14].

The TPM can act as a source for cryptographic-quality entropy.

5.17 Deprovisioning

Business Drivers: Industrial Control Systems have a long lifetime so they are often sold to new owners or reused for a different purpose. In doing so, there are multiple aspects that must be considered from a security standpoint.

[6] The term “module” is used here because IEC 62443 uses the words “component” and “device” to refer to a PLC, router, or other device that may include I/O modules. IEC 62443 apparently does not have a term for replaceable elements of a component or embedded device.

Industrial Control Systems often contain information that's considered sensitive by the Owner/Operator, such as PLC programming or robotic programming. This sensitive information must be reliably and securely destroyed when a device is taken out of service, such as when it will be resold or repurposed for another application. Otherwise, unauthorized parties can scavenge sensitive information from used devices.

Deprovisioning does not typically modify or reset permanent logs and counters. Those may show whether the device has been properly used and maintained. As such, the device seller might have an incentive to roll back or change these logs and counters and should be prevented from doing so. This contrasts with some PC client use cases where logs and counters might be cleared as part of a repair, refurbishment, and resale process.

5.18 Protecting Secrets and Intellectual Property

Industrial Control Systems may contain important trade secrets, proprietary information, and/or other intellectual property assets owned by the Device Manufacturer (e.g., proprietary settings, parameters, and software) or by the System Owner/Operator (e.g., programming). The interests of both parties should be protected to ensure that their secrets are kept from each other and from other parties.

5.19 Secure Update of Software and Configuration

Industrial Control Systems need occasional updates to software (including application code, ladder logic or similar code, operating systems, firmware, etc.) and configuration. The reasons for these updates vary: fixing bugs and security vulnerabilities, adding features, or simply changing the device configuration to match operational needs. Some updates only require Owner/Maintainer approval while others (e.g., an operating system update) may also require the approval of the Device Manufacturer.

Security is paramount for industrial control system updates. Without proper protections, attackers can use the update process to install their own malicious software or configuration. Such attacks have been used repeatedly in real-world settings to infiltrate industrial systems and compromise their operation. [45][46]

Preventing malicious software updates and ensuring the application of properly authorized updates is a fundamental element of industrial security.

5.20 Mitigating the Risks of Legacy Systems

Industrial Control Systems typically include "legacy systems" – old devices or systems that do not support the latest technologies. Legacy systems often include substantial security vulnerabilities, both known and unknown. However, these legacy systems have not or cannot be upgraded or replaced for a variety of reasons. For example, they may be perfectly functional with decades left in their useful life.

Making the business case for replacing legacy systems while they are functional is difficult because industrial control systems may have large up-front acquisition and installation costs that take decades to be fully amortized. Replacing or upgrading these systems can be very expensive and disruptive.

Therefore, industrial security architectures must include provisions for mitigating the risks of legacy systems. For more details on this, see section 6.20.

5.21 Disconnected Operation

Industrial Control Systems sometimes need to operate in a disconnected mode. The reasons for disconnected operation include:

- Remote locations often experience unreliable communications. For example, inclement weather or sunspot activity can impede radio communications. And some locations simply have unreliable network connectivity due to unreliable service providers. During outages, these locations are disconnected but must continue to operate.

- Even central locations may experience network outages due to device failure, human error, natural disaster, or other exceptional events. Critical systems must continue to operate properly during such outages to maintain safety and reliability.
- Some Owners may choose for safety, security, or reliability reasons to deliberately disconnect industrial systems and/or safety systems from all connectivity or to tightly control connectivity (e.g., a “data diode” that only permits data to flow out of the system).

Even when disconnected operation is deliberately chosen, the communications break is rarely absolute. USB drives, CDs, DVDs, or other external media are often used to bring software updates into the disconnected environment and bring data and reports out. These drives can also inadvertently or deliberately transfer malware, as evidenced by Stuxnet. [46]

5.22 Usable Security

When designing security, it is important that the final system be usable by non-security experts. The system should be easier to use in a secure way than in a non-secure way. The easier way to use the system will win, most of the time. If a requirement is made that causes passwords to be too long and complicated to be remembered, they will get written down. If they are required to be changed often, then a pattern for creating them based on something that is current (like the date) will often be used, making the change meaningless. NIST guidelines [32] recommend that strong passwords be created once, committed to memory and then not changed unless circumstances warrant it. Hardware authentication tokens or biometrics may have better usability than long passwords but they may not be practical in an industrial environment where protective gear can make token usage and biometric authentication difficult.

Over the last 30 years, it has been shown that one cannot train users to act in a secure manner. Phishing attacks continue to work no matter what amount of training is used, so operators should not depend on user training to overcome deficiencies in the design.

6 TECHNOLOGY RECOMMENDATIONS

This section outlines approaches for the use of TCG technology to satisfy the Use Cases in Section 5 above. This section suggests ways that existing TCG technology can be put to use.

6.1 Device Identity

Device Identity requirements can be satisfied by the use of either TCG Platform Certificates or IEEE 802.1AR Device ID certificates (See Section 5.1 above). For more information on these certificate options, see *TCG Platform Attribute Credential Profile [20]* or Appendix B of *IEEE Std 802.1AR-2018 [2]*.

6.1.1 Methods

6.1.1.1 Using IEEE 802.1AR

Figure 4 shows the relationship of credentials installed in the TPM during its lifetime:

- The EK and its Certificate are added by the TPM Manufacturer to provide proof that the TPM is authentic
- Initial Device ID (IDevID) credentials are installed by the device manufacturer to identify the device's manufacturer and typically also the serial number.
- The device manufacturer may also install an Initial Attestation key (IAK) and certificate to allow attestation without requiring the Maintainer to set up a CA (see Section 6.1.2.3)
- The Device Manufacturer can also provide a mechanism to allow a Maintainer to configure Local Device ID's (LDevIDs) and their own Attestation keys (LAK) (Section 6.1.3)

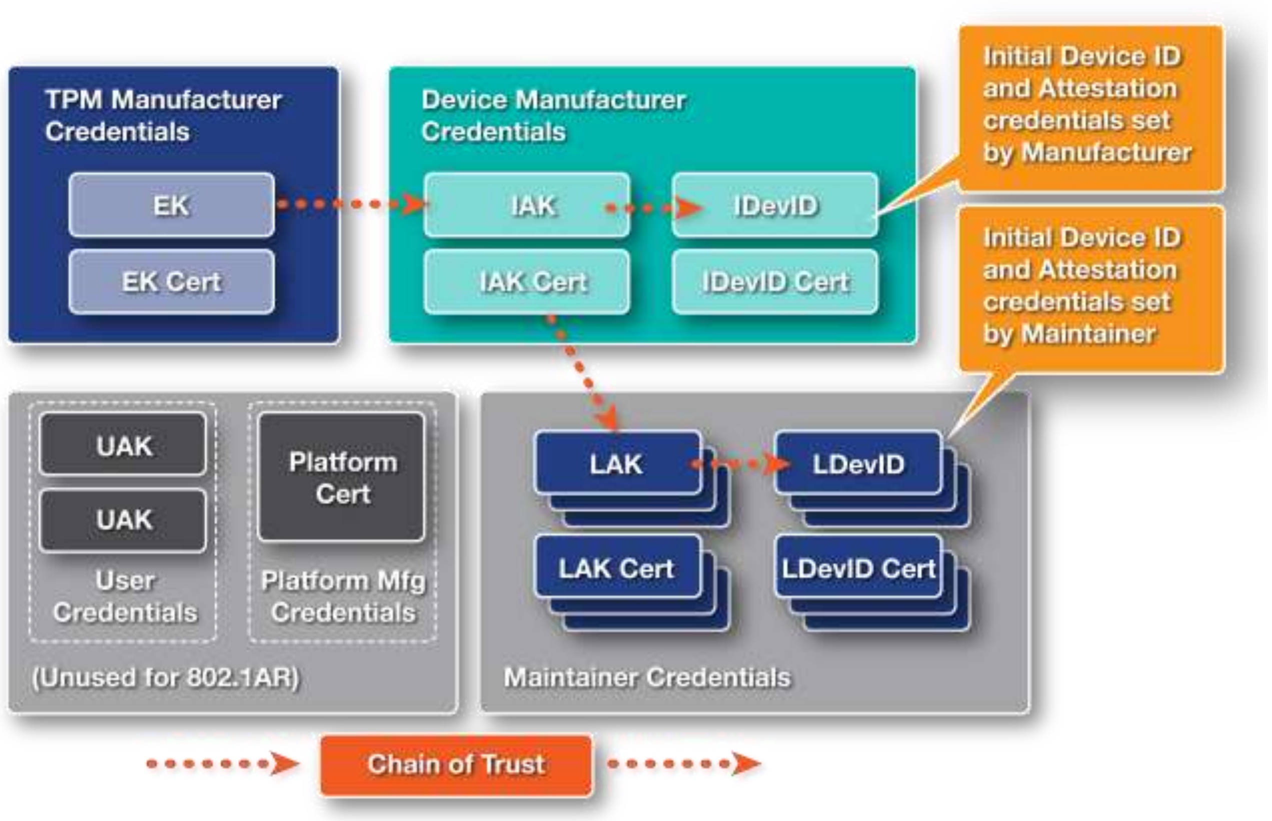
TPM Manufacturer Credentials
EK
EK Cert
Device Manufacturer Credentials
IAK
IDevID
IAK Cert
IDevID Cert
Initial Device ID and Attestation credentials set by Manufacturer
Initial Device ID and Attestation credentials set by Maintainer
UAK
UAK
User Credentials
Platform Cert
Platform Mfg Credentials
(Unused for 802.1AR)
LAK
LDevID
LAK Cert
LDevID Cert
Maintainer Credentials
Chain of Trust

Figure 4 - Credentials for Industrial Equipment

Several TCG documents are relevant to the use of TPMs to implement 802.1AR certificates (DevIDs):

- *TPM Keys for Platform Identity for TPM 1.2* [5] This document gives details useful for DevIDs in TPM 1.2, including a number of use-cases for DevID keys, and techniques for provisioning these keys.
- *TCG TPM v2.0 Provisioning Guidance* [7] This document gives guidance on TPM 2.0 hierarchies, etc., setting out expectations for the TPM Manufacturer, Device Manufacturer and Maintainer in the general case of TPMs in many kinds of devices.

To support Device Identity requirements, the Device Manufacturer must configure the TPM:

- Either the TPM Manufacturer must ensure that the TPM is configured with an EK and EK certificate when the chip is manufactured, or the Device Manufacturer must generate the EK and EK Cert.
- Although TPM 2.0 is automatically enabled, a TPM 1.2 must be configured as "always on", i.e., Enabled and Activated, when the Device Manufacturer's product is built.
- The Device Manufacturer must ensure that the TPM is configured with an IDevID key and an IDevID Certificate before the Device leaves the manufacturing facility.

6.1.1.2 Using TCG Platform Certificates

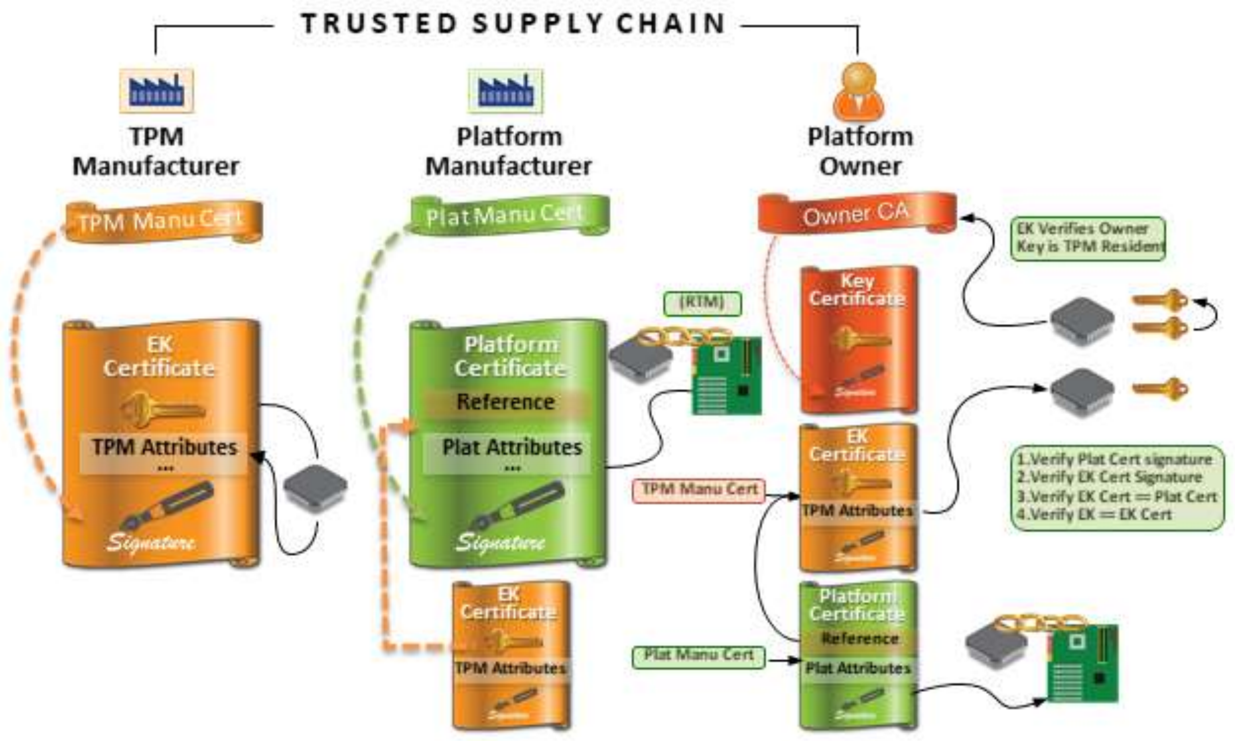

Figure 5 - Platform Certificates in Context

As illustrated in Figure 5, a TCG Platform Certificate provides a cryptographic binding between a Platform Certificate issuer (for example, the platform's manufacturer) and a specific platform. This cryptographic binding starts with the platform's TPM Endorsement Key, to its associated Endorsement Key Certificate, to the Platform Certificate, then to the Platform Certificate issuer (e.g., the platform manufacturer). This is done by including the identity of the Endorsement Certificate (specially the certificate's Holder field) into the Platform Certificate. This enables the verification of the platform's source and claimed identity. The Platform Certificate contains detailed information about the platform as defined by TCG.

TCG Platform Certificates are complementary to IEEE 802.1AR in that they both provide platform identity using platform-resident keys. IEEE 802.1AR does not require the use of a TPM and is therefore more general (in fact, 802.1AR is silent about the location and method for protecting the authenticating key). The TCG Specification mentioned in section 6.1.1.1 above specifies how to use the TPM's Endorsement Key for this purpose. While the TCG Platform Certificate provides the same binding to the platform as DevID, the Platform Certificate provides more TCG and Platform specific information about the platform. There are use cases for both the DevID certificate and the TCG Platform Certificate as a stand-alone method and as complementary methods (i.e., there is a use case for issuing both for the same platform).

6.1.1.3 Using DICE

TCG's Device Identifier Composition Engine (DICE) can be leveraged to satisfy device identity requirements as well as provide counterfeit protection and attestation. TCG DICE is also compatible with IEEE 802.1AR.

TCG has published documents [22] [23] providing information on the use of DICE for device identity and attestation. Future TCG DICE specifications may provide greater detail on DICE layering architectures, identity and attestation methods using both symmetric and asymmetric cryptography, as well as guidance on leveraging DICE to achieve IEEE 802.1AR device identification.

6.1.2 Device Manufacturer Device Identity and Counterfeit Protection

Device Identity and Counterfeit protection can be provided by an IEEE 802.1AR Initial Device Identifier (IDevID) certificate, signed by the Device Manufacturer and installed in the device while it's still under the manufacturer's control. This certificate provides cryptographic evidence that a specific physical device was manufactured by the specified manufacturer.

For this purpose, IEEE 802.1AR certificates should include the device serial number in the subject name and be signed by or traceable to the manufacturer's CA. The combination of manufacturer identity and Device serial number is expected to be unique.

An individual industrial device may comprise many field replaceable units (FRU's). In this case, the serial number in the certificate should reflect the serial number of the platform (the industrial device). This may require installing a new IDevID and/or platform certificate in the field if the FRU (Field Replaceable Unit) with the TPM needs to be replaced.

6.1.2.1 Configuring Initial Identity Credentials

To support Device Identity requirements, the Device Manufacturer must configure a certificate and keys in the TPM. There are differences between TPM 1.2 and TPM 2.0 in this process; this section outlines common elements, and subsequent sections describe the differences.

In all cases, the Device Manufacturer must ensure that the TPM is configured with an EK, EK Certificate, an IDevID key and certificate, before the industrial device leaves the manufacturing facility. The manufacturer may also want to configure an Initial Attestation Key (IAK) and certificate (see Section 6.1.2.3).

To do this, the Device Manufacturer will create the IDevID key-pair in the TPM, issue a signing request containing the device serial number and other information to its own CA, and sign the certificate. Per 802.1AR, the signature should use SHA256. (See Section 6.1.2.4 for how to do this with TPM 1.2). IDevID keys must be created with attributes that prevent duplication of the key from one TPM to another, to ensure that the identity of a Device can't be moved from one instance to another. (See Sections 6.1.2.4 and 6.1.2.6.)

TCG EK Credential Profile For TPM Family 2.0 [8] offers the following suggestion for EK credential lifetime, and the same advice should be considered for IDevID, or other immutable certificates. To quote section 2.2.2:

> 2.2.3 EK Credential Lifetime

An EK Credential contains fields that express the validity period of the credential. The validity period is at the discretion of the manufacturer. The credential is not expected to expire during the normal life expectancy of the platform in which it resides. The lifetime can vary widely between different types of platforms (e.g. while a typical validity period for a PC Client platform is 5-10 years, non-user device TPMs are expected to operate indefinitely into the future in which case the value 99991231235959Z should be used as expiration date). The credential lifetime can also depend on the lifetime of the TPM device and the algorithm type of the Endorsement Key. The time frame during which the security strength of the EK is acceptable SHOULD be taken into account by the manufacturer when determining the credential lifetime (e.g. see NIST SP800-57 [21]).

It is believed that there is no Personally Identifiable Information in industrial device PCRs, but that determination must be made by the Device Manufacturer.

6.1.2.2 Signing and Encryption Key Types

Two types of keys could be used for DevIDs in a TPM, as defined in *TPM Keys for Platform Identity for TPM 1.2* [5] Section 2.4.:

- Signing keys, which are restricted to signing, and allow no other operations
- General Purpose or Combined (aka Legacy) keys which can be used for signing as well as encryption and decryption

Consistent with current best practices, General Purpose keys should not be used for DevIDs. Rather, separate DevID key pairs and certificates should be used for signing and encryption.

6.1.2.3 Identity for Attestation

TPM mechanisms are arranged so that in privacy-sensitive situations, Remote Attestation can be done either without knowing the exact identity of the target system, or by trusting the correlation between platform and attestation identity to a special Privacy CA (also known as Attestation CA; see Chapter 7 of *IWG Reference Architecture for Interoperability (Part 1)* [17]).

For industrial devices, where clear identity of the devices in question is usually a critical requirement, a simpler approach can be used. In this case, the device manufacturer should provision an *Initial Attestation Key (IAK)* and certificate that parallels the IDevID, with the same device ID information as the IDevID certificate (i.e., the same Subject Name and Subject Alt Name, even though the key pairs are different). This allows a quote from the device, signed by the IAK, to be linked directly to the device that provided it, by examining the corresponding IAK certificate.[7]

Inclusion of an IAK does not preclude a mechanism whereby a Maintainer can define Local Attestation Keys (LAK's) if desired (Section6.1.3.2).

6.1.2.4 Initial Identity with TPM 1.2

TPM 1.2 has relatively limited mechanisms for managing Ownership:

- For TPM 1.2, the Device Manufacturer must configure TPM Ownership to make an SRK, so that an IDevID may be created as a child of the SRK.[8]
- To ensure that the IDevID credentials are not rendered invalid by deletion of the SRK, Ownership of the TPM must be locked.

[7] As long as the IDevID and IAK are signed by equally-reputable CAs (or more simply, the same CA) and contain the same Subject Name and Subject Alt Name, this approach eliminates the Asokan Attack, https://tools.ietf.org/html/rfc6813
[8] For TPM 1.2, this is an unavoidable overlap of administrative domains; normally, Ownership would be taken by the Administrator, not the Manufacturer, but that's incompatible with a manufacturer-supplied IDevID.

In order to prevent the SRK from being deleted, rendering any keys stored under it useless, TPM Ownership can be locked with the non-volatile TPM_DisableOwnerClear() ordinal. If the TPM supports physical presence, the volatile TPM_DisableForceClear() ordinal may be used to block ForceClear on each startup.

The detailed structure of the 802.1AR certificate for TPM 1.2 is given in *TPM Keys for Platform Identity for TPM 1.2* [5].

The 802.1AR document requires the use of SHA256 for signature generation, although TPM 1.2 can only perform SHA1 signatures using its built-in functions. However, TPM 1.2 allows the host to compute the hash, prepend the appropriate ASN.1 encoding, and to submit this to the TPM for SHA256 RSA signature generation using Distinguished Encoding Rules (DER).[9]

TCG TPM 1.2 documents do not define handles that should be used to identify DevIDs, so these must be assigned by the Device Manufacturer.

Identity keys in TPM 1.2 should be created as *non-migratable*, or a Certified Migration Key (CMK), to eliminate the chance that a device identity could be improperly copied from one device to another.[10]

6.1.2.5 Key Types in TPM 2.0

In TPM 1.2, identity keys must be descendants of the Storage Root Key, but in TPM 2.0, there are a number of options.

- Secondary, Internally-Generated keys are like DevID keys in TPM 1.2; the key pair is generated in the TPM, and can be configured so the private key is never released. A Secondary key is always the child of a parent key in the hierarchy.
- Primary keys are unique to TPM 2.0. These keys are also generated in the TPM, but they are generated using a deterministic Key Derivation Function that starts with a generic Template, and uses a primary seed in the TPM to generate the key pair. Primary keys can be deleted and regenerated from the template as long as the seed value is unchanged.
- TPM 2.0 also includes a specialized mechanism where an external entity can generate the key pair and wrap it so it can only be imported to a specific TPM. This can be used to circumvent the non-deterministic time taken to generate an RSA key inside the TPM, or it can be used to meet unique security requirements where an external key generator must be used. Imported key pairs cannot be Primary keys, and they cannot be marked as non-duplicable (i.e., they can’t be marked as fixedParent). (Listed in the table below as “Secondary Imported”)

Table 1 shows the different properties of the key generation approaches.

[9] See *TPM 1.2 Part I Design Principles* [29], Section 31.2.2 TPM_SS_RSASSAPKCS1v15_DER.
[10] In TPM 1.2, it’s possible to use a Certified Migration Key (CMK) with suitable Migration Selection Authority (MSA) and Migration Authority (MA) to allow a key to be moved under controlled conditions. The point, of course, is to make sure that an unauthorized copy of a DevID cannot be made.

Primary Key	Secondary Internally-Generated	Secondary Imported
Must be created from a template; cannot be Imported	Generated internally	Key generated externally, wrapped with the parent key
RSA key generation is non-deterministic, and can take a long time – possibly a minute or more of elapsed time. ECC has no such problem.		Importing keys is deterministic, as they're wrapped with a symmetric key
DevIDs should be created as non-Duplicable (FixedTPM, FixedParent)		Imported keys can't be marked as fixedTPM, and require a Policy to block export or migration.
Key may need to be made Persistent to avoid long regeneration each power-on	Key can be saved, wrapped by its parent, then Loaded on power-up or use.	Importing the key produces a version wrapped by the parent, which can be quickly reloaded when needed.
Certificates can be generated only with a multi-step Online process using Endorsement Credential and TPM2_ActivateCredential()[11]		Keys and certificates can be generated externally and installed off-line across an air-gap if necessary.
Key can be re-generated from a generic template after TPM2_Clear()	Keys are lost in case of a TPM2_Clear()	Wrapped key can be re-imported after a TPM2_Clear() (Assuming the parent is restored, and the Wrapped key is not lost)

Table 1 - Key Generation Options for TPM 2.0

For the remainder of this section, this document assumes the following:

- In most applications, IDevID keys will be Primary keys installed by the device manufacturer. The Clear operation may be used to remove user identity and keys from a device, but the IDevID keys can be regenerated from a generic template after a TPM2_Clear() operation.

- LDevID keys are Secondary, internally-generated keys, and are intended to be deleted by a TPM2_Clear() operation.

- Manufacturers can also support Imported DevID keys if they have high-throughput manufacturing processes where the indeterminate wait for RSA key generation would be unacceptable, or for specialized customers that want to manage their own key generation.

In TPM 1.2, special Restricted Attestation keys must be used to sign internally-generated structures such as quotes, to prevent an external actor from spoofing a quote. The TPM 2.0 retains this capability with Restricted Signing Keys, which can sign an internally-generated quote or external data, as long as the external data doesn't contain the fixed pattern that matches an internally-generated structure. (See *Trusted Platform Module Library Family 2.0 Part 1* [3], Section 25.1.2)

Thus, a Restricted Signing Key could be used for both Identity and Attestation, because the TPM will refuse to sign any value that looks like a quote.

However, for the remainder of this section, this document assumes that Device IDs (IDevID, LDevID) are separate keys from Attestation keys (IAK, LAK).

[11] Since the key-pair is generated in the TPM, it's not known in advance, implying the need for a multi-step procedure to generate the key and then incorporate the public half in a matching certificate. Conversely, the public key of an imported key pair is known in advance, so a matching certificate can be delivered along with the wrapped key for import.

6.1.2.6 Initial Identity with TPM 2.0

To support Device Identity requirements, the Device Manufacturer must configure keys in the TPM as described in the *TPM v2.0 Provisioning Guidance* [7] document.

IDevID keys must be signing keys. If they are also Restricted, then they can serve as attestation keys. However due to complexities of using restricted keys with off-the-shelf crypto software stacks such as OpenSSL[12], platform vendors may opt to provision IDevID keys as unrestricted signing keys. Doing so would necessitate provisioning a separate restricted signing key pair for attestation purposes (identified as an IAK in Figure 3).

Whether IDevID keys are externally-generated keys, as noted in Section 6.1.2.5, or Primary keys, they should be stored in the Endorsement Hierarchy.

IDevID keys are long-lived persistent keys for the device. They are analogous to the EK but without the privacy restrictions placed on the EK. Therefore, they are stored in the Endorsement hierarchy. The Platform hierarchy could be used, but generally objects in the Platform hierarchy are not available after the platform OS has launched. The Storage hierarchy is inappropriate because that is cleared when the device owner changes.

The Device Manufacturer must ensure that the TPM is configured with an EK, EK Certificate, an IDevID Key and IDevID certificate before the industrial device leaves the manufacturing facility. Several additional measures should be taken to protect the IDevID before the device ships to the customer:

- The Device Manufacturer should provision the Endorsement Hierarchy authorization
- The IDevID certificate (if it is stored in the TPM) should be write-locked to prevent it from being deleted

NOTE: The IDevID Template does not need to be stored in the TPM, as it is public information and not specific to a unique device.

In order to prevent the EPS (Endorsement Primary Seed) from being changed, rendering any keys stored under it useless, access to the TPM2_ChangeEPS() command must be restricted. This command is optional[13], and ideally it would *not* be implemented by the TPM Manufacturer. If it is implemented, then the Device Manufacturer must implement strict controls, provisioning the Platform Authentication to a random value or un-fulfillable policy upon every platform reset.

TPM2_Clear() resets the endorsement hierarchy authorization, clears out any keys under the Endorsement hierarchy and the Storage hierarchy, and performs other operations as described in the Trusted Platform Module Library Part 1, such as removing NV-defined indexes which are not protected by the TPMA_NV_PLATFORMCREATE flag. If the IDevID certificate is stored in the TPM, it should be placed in the Platform NV hierarchy, by setting its TPMA_NV_PLATFORMCREATE flag, so it's not erased by a TPM2_Clear().[14]

To prevent permanent loss of the IDevID keypair, the Device Manufacturer should block TPM2_ChangeEPS.

The exact format of an IDevID certificate for TPM 2.0 may be the subject of future TCG Specifications.

TCG TPM v2.0 Provisioning Guidance [7], Section 7.8 gives defined handles for IDevID credentials in TPM 2.0 [49].

[12] Signing external data with a Restricted key requires a Ticket, and there's some concern that managing tickets would require an involved OpenSSL rewrite. That is a non-trivial exercise.
[13] The 'optional choice' is exercised by the TPM Manufacturer; if the TPM Manufacturer implements the ordinal, the Device Manufacturer needs to block it.
[14] There is no Endorsement hierarchy for NV RAM, so the only two choices for the IDevID cert are Platform (which does survive TPM2_Clear) or Storage (which does not). The EK Certificate is stored in Platform; the IDevID certificate should be too.

It must not be possible to move identity keys from one TPM to another. In TPM 2.0, the mechanism to prevent keys from being duplicated depends on the method of creation:

- If keys are generated within the TPM, FixedParent and FixedTPM must be set (see *TPM 2.0 Trusted Platform Module Library, Part 2: Structures*, Section 8.3 [4]).
- If keys are generated outside the TPM, they can't be imported and marked as fixedTPM. Hence imported keys can't be marked as Fixed, so they must be accompanied by a Policy that prevents export.

6.1.2.7 Loss of IDevID Credentials

Device Manufacturers must be aware that loss of IDevID credentials in a fielded device could be hard to remedy.

Although the IDevID key-pair must be protected by the TPM to ensure that the private key remains secret, it's up to the Device Manufacturer to store the IDevID certificate (and possibly a wrapped IDevID key) somewhere secure, but not necessarily secret, in the platform. Device Manufacturers *could* lock the IDevID certificate and wrapped key into NVRAM in a TPM, although there's no requirement to do so, and NV space is often at a premium.

For devices that don't have space in the TPM NVRAM to store Initial identity credentials, there are some alternatives:

- If the manufacturer doesn't record the device's ID information in its own database, loss of these credentials will require return-to-factory where Initial identity keys are regenerated.
- If the manufacturer does record EK information, it is possible to re-create an IDevID in a fielded device using enrollment based on TPM Certify mechanisms, even if Ownership is lost. (See *TPM Keys for Platform Identity for TPM 1.2* [5])
- For TPM 1.2, as long as Ownership is maintained, any wrapped IDevID key and matching certificate could be replaced from a manufacturer's database. See Sections 6.1.2.4, 6.1.2.6 on locking Ownership.

6.1.2.8 Upgrading Cryptographic Algorithms

Manufacturers may want to support upgrade of cryptographic algorithms in devices with long lives. While cryptographic algorithms cannot be changed in TPM 1.2, the TPM 2.0 specification offers a mechanism to specify specific algorithms supported by the TPM. It may be possible to even change the supported algorithms of the TPM using a field upgrade. Note that changing algorithms may be vendor-specific, and might not be possible in all cases.

6.1.2.9 Compromise of Keys

Private keys associated with the EK and IDevID are immutable; once manufactured, these keys are not easily changed in a fielded device. However, compromise is unlikely, as they are protected by the TPM.

Manufacturers need to take care to protect intermediate keys stored in CAs in their manufacturing facility, since exposure of a signing key could compromise many devices, with little recourse to resecure them short of return-to-factory.

6.1.3 Local Device Identity

IEEE 802.1AR also includes provision for Local DevID's: customer-specific certificates that can be installed by the Maintainer of the industrial device, during or after installation. LDevID certificates will be signed by the Maintainer's CA.

While the LDevID can contain anything the Maintainer wants, they're probably best thought of as enterprise "asset tags", providing cryptographic evidence that the Device in question is one that was legitimately configured for use within the Maintainer's enterprise. Multiple LDevID tags might be applied to a single device, depending on its organizational role(s).

The Device Manufacturer must provide a mechanism to remove Maintainer-installed LDevID keys without deleting IDevID.

Modular industrial devices may contain multiple Field Replaceable Units: as with IDevID certificates, the basic LDevID should identify the overall Device not just the FRU. If a FRU containing a TPM is replaced, a new LDevID with the Device identity should be installed into the FRU's TPM.

6.1.3.1 How to Install a Local Device ID

A Maintainer may want to install LDevID credentials remotely, when the device is located off-site. As such, care must be taken that the LDevID is being installed in the right device.

The following steps can be used to install an LDevID into a Device that's already equipped with an IDevID and IAK. These steps assume *Enrollment over Secure Transport* (EST). [26]

- Log into the Device using a TLS session that can't be easily hijacked, authenticated by the IDevID, and use that session to complete all the steps of enrollment.
    - Use of the IDevID certificate for TLS authentication ensures that the connection terminates on the right device by showing the serial number assigned by the device manufacturer.
    - The Device's identity as expressed by the manufacturer's serial number can be trusted if:
        - The certificate can be traced to the Device Manufacturer
        - The Device can respond to a challenge, proving the Device holds the IDevID private key
    - The Manufacturer's serial number can be checked against a bill-of-sale to ensure it's the right device.

  These steps provide proof that there's a secure connection to the exact device into which the LDevID should be placed.

- The device should generate the LDevID key pair and a Certificate Signing Request (CSR) containing the public key and the asset information selected by the Maintainer.
    - The CSR may also contain TPM_CERTIFY_INFO signed by the IAK key to prove that the LDevID key is in the same TPM as the IDevID[15],[16].
- The CSR should be returned to the Enterprise CA.
    - Since an IDevID key must be configured to prevent duplication from one TPM to another, the enterprise CA can use the TPM_CERTIFY_INFO to convince itself that the IDevID and LDevID are in the same TPM, and that TPM is in the intended Device.

[15] This proof is simple if the IAK contains the same device serial number as the IDevID, and they're signed by the same issuing authority. Other mechanisms must be used if the IAK is created independently. See Section 6.1.2.3.

[16] Use of TPM Certify yields a trustworthy mechanism to get the LDevID into the same TPM as the corresponding IDevID, even if the control plane software on the device can't be trusted. If TPM Certify is not used, there's a possibility of variations of the Asokan attack, where the Device might have a valid IDevID, but do something that misuses the LDevID, e.g., not putting it a TPM at all, or passing it off to some other device. TPM Certify must be verified by a CA that puts the TCG OID indicating TPM key residency into the resulting certificate.
Note that TPM Certify info in TPM 2.0 (TPMS_CERTIFY_INFO) is quite different from TPM Certify info for TPM 1.2 (TPM_CERTIFY_INFO and TPM_CERTIFY_INFO2), but they both satisfy the requirement of proving co-residency.

- The Enterprise CA can then create the LDevID certificate with Object Identifiers (OIDs)[17] showing TPM residency, sign it and send it back to the device.

An analogous procedure can be used for cases where an externally-generated key pair is desired for LDevID (i.e., LDevID doesn't have to be generated in the TPM for those cases where an externally-generated key would be more appropriate).[18]

Chapter 21 of the book *A Practical Guide to TPM 2.0* [11] also outlines techniques that can be used to remotely provision DevID credentials.

6.1.3.1.1 Conveying TPM Certify Information

While there are a number of ways to get the TPM_CERTIFY_INFO from the TPM to the CA responsible for signing a DevID, this document recommends uniform practice between TPM 1.2 and TPM 2.0.

- For TPM 1.2, TPM_CERTIFY_INFO is added to the Certificate Signing Request as an X.509 extension. This extension is defined in Subject Key Attestation Evidence (SKAE) [9].
- For TPM 2.0, the TPM2_Certify structures are complex and specific to the TPM. Certification procedures specific to TPM2 may be the subject of future TCG specifications.

6.1.3.2 Attestation with Local Identity

Maintainers who wish to use LDevIDs for device identity, using their own credentials to identify devices, may want to provision "local" Attestation Keys (LAKs) allowing the Attestation system to reliably identify the device within the Maintainer's context. Use of a Local AK in addition to the LDevID can prove that attestation has not been compromised by an Asokan attack (as defined in IETF RFC 6813 [27]).

6.1.4 Identity for Network Access

When presenting identity for network access, the Device may use its IDevID or one of its configured LDevID certificates, at the discretion of the Maintainer. The appropriate certificate should be chosen by methods provided by the Device's operating system software.

IETF's Transport Layer Security [29] Section 4.3.2 defines the mechanism by which a server can negotiate trust anchors, by, for example, specifying a list of the distinguished names of acceptable certificate authorities.

6.1.4.1 Proving a Link to the TPM

DevIDs may be used to authenticate access to various kinds of network services, in which case the receiving service must decide how much trust can be placed in the credential.

DevID private keys that can be shown to be held by a TPM may be judged more credible, and less likely to have been compromised.

There are two techniques to demonstrate to a network service that a particular DevID key is stored in a TPM:

- Sign the DevID cert with a CA that will only be used to sign certificates that are known by the Maintainer to originate in a TPM. For TPM 1.2, this may include a specific TCG OID corresponding to the Certificate Practices Statement (CPS).
- Extend the service's login authentication to parse an additional X.509 extension field called a Subject Key Attestation Extension (SKAE, TPM 1.2 only).

[17] For TPM 1.2, the appropriate OID is defined in *TPM Keys for Platform Identity for TPM 1.2* [5], Section 3.1.1.1.
[18] Externally-generated keys don't necessarily have to risk exposure. An HSM (or another TPM) could generate an LDevID key pair, and wrap it with a key known only to the TPM destined to receive the LDevID.

The use of SKAE to send proof to a CA that a DevID key is stored in a TPM 1.2 device is described in *TCG Infrastructure Workgroup: Subject Key Attestation Evidence Extension [9]*.

The CA can certify that the key is in a TPM using the mechanism described in *TCG Infrastructure WG TPM Keys for Platform Identity for TPM 1.2 [5].*

To quote Section 3.1.1.1:

> *In order to assert that the CA verified TPM residency for the key used in a DevID certificate, a certificate policy OID unique to the TCG (and for this purpose) MUST be included in the certificate policy extension of the DevID certificate. The CA MUST perform verification of the AIK signature over the TPM_CERTIFY_INFO or TPM_CERTIFY_INFO2 prior to including the certificate policy TCG OID.*

6.1.4.2 Why Not Just Use the EK and Platform Certificate for Identity?

While the Endorsement Key (EK) as defined by the TCG PC Client [35] may be present by default in TPMs because of the economics of TPM manufacture, the EK is not always a good candidate for device identity.

- The conventional use of the EK credential is to prove the provenance of the TPM itself, by linking to a chain of certificates rooted at the TPM manufacturer's CA.
- The EK is not used directly as a signing key to prove possession of the EK secret key: to protect privacy in privacy-sensitive applications, that proof can only be done indirectly through other keys using TPM Certify information.

As a result of these restrictions on EK use, a separate set of credentials traceable to the device manufacturer is used to identify the device.

Similarly, some Device Manufacturers may supply a Platform Certificate including a manufacturer's serial number (see *TCG Platform Attribute Credential Profile* [20]). While that's similar to a DevID, again, they're not the same:

• The Platform certificate is an *Authorization Certificate*, not a Public Key Certificate (See RFC 5755 [40]).

• An Authorization Certificate isn't bound to a key pair, so it can't be used for proof of identity. The Platform credential does identify the associated EK credential, but has no key of its own for signing.

In some cases, the platform credential might be used as a link in the chain to create and install an IDevID or LDevID after the device has left the manufacturer's premises, although that use is not described in this document.

6.2 Access Control

As noted in section 5.2, access control in an industrial setting can be broken down into three categories or aspects: Device Access Control, Network Access Control, and Data Access Control. The following subsections explain how each of these can be implemented.

6.2.1 Device Access Control

Device access control determines which users and devices can access a particular device. Several methods may be used to implement device access control, such as physical security (doors, locks, etc.), network security (via firewalls, gateways, or application code), and HMI-based authentication (e.g., a password or PIN entered via a Human Machine Interface like a keypad).

Device access control is governed by an "access control policy" or "authorization policy" that indicates which actors (users and devices) are permitted to access which devices. More sophisticated policies may permit partial access for some actors and greater access for others. Policies may consider other factors such as time-of-day and may permit emergency overrides. To ensure consistency and minimize the need to update policies, policies often employ abstractions such as user roles or device groups. A TPM or DICE can assist with device access control by providing strong device and user identity.

6.2.2 Network Access Control

Industrial networks are often highly sensitive. If an unauthorized person or device is connected to such a network, the person or device may be able to exploit vulnerabilities in connected systems. Additionally, the unauthorized party may eavesdrop on the network or try to jam it. For these reasons, access to industrial networks is generally restricted.

Network access control can be implemented using several methods. The simplest method is to physically disconnect the network from any other networks (an “air gap”). While this method seems simple, it is not as easy or effective as it appears. Business imperatives often require connectivity to the outside world, either by carrying files on USB sticks or by connecting the network through a “jump host” or a network firewall. Through these interchange mechanisms, attackers can and often have gained access to air gapped networks. [45][46]

More sophisticated ways to implement network access control include firewalls, VLANs, gateways, data diodes, and the IEEE 802.1X standard. However, 802.1X is not generally used in industrial settings due to the complexity involved. When evaluating the security of such techniques, one must always consider the security level of the isolation method. A TPM or DICE can assist with network access control by providing strong device and user identity.

6.2.3 Data Access Control

Some industrial data is highly sensitive (e.g., a secret formula for a product) while some is less sensitive (e.g., readings from a flow sensor). For this reason, access to some data must be tightly controlled while other data may be less restricted. Even for data that does not need to be held confidential, protecting the integrity and availability of the data may be important.

Data access control measures ensure that only authorized parties can access (read, modify, or delete) particular data. Examples of these measures may include encryption, authentication, integrity checks, etc. The TPM and SEDs can help with data access control, as described below.

6.2.4 TPM 2.0 Access Controls using Policy

TPM 2.0 provides a means, called policy, for describing which conditions must be satisfied to have access to TPM privileged operations. Those conditions include:

- Changing GPIO state (the state of a General Purpose I/O pin on the TPM)
- Accessing NV indexes for read or write actions
- Accessing keys for encryption or signing

Policy for these operations, which can be controlled individually for any TPM controlled entity, can encompass (amongst others) any logical AND or OR combination of:

- Proof of knowledge of a password
- Proof of ability to use an asymmetric private key
- Evidence that the machine is in a particular state (as measured by PCRs)
- Proof that a TPM GPIO is in a particular state
- Proof that a biometric associated with a particular person has been presented (using evidence from a particular biometric reader, as signed by that reader)

- Proof that the system is in a particular location (using evidence from a GPS, as signed by that GPS)

These combinations provide a building block that can be used either directly (through control of a GPIO or NV index) or indirectly (using a TPM signature to authorize another action), to provide policy based control of external systems.

6.2.5 Access Control for Opal Drives

TCG Opal drives have the ability to make sections of the drive read only, thus allowing a system provider to prevent overwrite of sensitive data configurations. Opal drives restrict control to Maintainer Passwords (or Owner passwords through delegation) that can be used to provide write access to sections of the drive.

6.3 Secure Zero Touch Provisioning

Industrial devices typically must be specially configured for the situation where they will be used. This configuration might enable access to a corporate VPN or other network, might authorize access to restricted content, or might install additional software such as management agent or firewall software and IT security policy configuration for the device.

There are many cases where an industrial device may be shipped with no customer-specific configuration, and must be configured before it can be used.

In these cases, Zero Touch Provisioning (ZTP) may be desirable, requiring a mechanism where the device communicates through the network as soon as it's activated, to obtain the configuration information that will specify policy for operational use.

Currently, the implementations and deployments of these techniques are limited, but they could be more common in the future.

Part of this use-case may require proof that the device is actually in the hands of an authorized user before it is configured, perhaps through the use of an out-of-band authorization code.

The IETF Zero Touch Provisioning efforts (an ongoing effort when this document was published) provides possible frameworks in which the configuration process can work. See IETF RFC 8572, "*Secure Zero Touch Provisioning (SZTP)*" [30].

- A networked industrial device can use a Device Manufacturer-installed credential to authenticate itself to the Zero Touch Provisioning service to ensure that this device is authorized to be connected and to identify which device is being connected.
- The industrial device can check a credential supplied by the Zero Touch Provisioning service to ensure that the service is authorized to provide configuration.
- The Zero Touch Provisioning server can use Remote Attestation (Sections 5.12 and 6.12) to verify that the device is running authorized software before completing the configuration.

FIDO [42] is also working on standards for Secure Zero Touch Provisioning.

TCG technologies can improve the security of ZTP in several ways:

- A DevID key protected by the TPM (see section 6.1) can provide cryptographic assurance of the identity of the device attempting to register for configuration. However, the owner will generally need to additionally have a way to verify that this device is authorized to connect to the owner's network. For example, the owner can maintain a database of known and expected devices with a DevID or manufacturer ID and serial number for each device. The manufacturer and serial number approach will work only if the DevID contains the serial number, which is required for the IDevID but only recommended for the LDevID. One way to do this is described in IETF RFC 7030, *Enrollment over Secure Transport*. [26]

- The Trusted Network Connect IF-M protocol suite, together with PTS, can be used to offer assurance to the central configuration system that the Device is running authorized software. This mechanism is described in IETF RFC 5792, *PA-TNC: A Posture Attribute (PA) Protocol Compatible with Trusted Network Connect (TNC)* [41]. See also section 6.12 below.
- Configuration information can be written into NV indexes controlled by the platform hierarchy, which is generally not controllable by the end user.
- Configuration information can be written (by the Maintainer) into a read only section of an opal drive (only writable by the Maintainer). Note that this requires that physical access to the drive be restricted, so that the PSID (Physical Security ID) written on the drive can't be used to return the drive to an unprovisioned state and then reinstate all the data that was on it, together with a rogue's Maintainer password.

6.4 Securing Secrets

As previously noted, Industrial Control Systems sometimes contain or process secrets: data, configuration, and code that must be kept confidential. Because Industrial Control Systems are often located in insecure locations and subject to network attacks and exploitation of software vulnerabilities, protection may need to include protection of data at rest (while stored), data in motion (while being transmitted), and data in use (while being processed). Many techniques can be used to protect keys and other secrets, depending on the operations that must be performed with them and on other requirements such as performance.

Both DICE and TPMs can be used to help solve these problems.

DICE can provide a secret readable only from ROM during boot, which in turn can be used to create a secret that is derived in part from the firmware of the system, and usable by that firmware in a number of processes that can be used to encrypt and decrypt data, cryptographically bind configuration or software to a system, or attest remotely that the system has not had its software or configuration changed.

TPMs can provide data at rest protection for secrets, either by storing those secrets directly in the TPM's NV storage or by encrypting the secrets with keys that are protected by the TPM. For example, TPM policies can be used to restrict access to stored secrets or symmetric or asymmetric keys based on a system's configuration and software through the use of PCRs and based on other conditions such as required authorization values. These techniques can also provide data in use protection by using PCR values to restrict which software can access the data or keys.

The same mechanism used to create and store LDevID private keys inside the TPM can be used for other uses of asymmetric keys (persistent or not). This can provide data in transit protection if the keys are used to establish communications sessions. However, using the TPM to perform asymmetric crypto handshakes typically works only if the rate at which challenges must be signed is modest (a few per second for many hardware TPMs).

In cases where higher throughput is needed or applications where shared-secret symmetric keys are used, it will be necessary to pass the cryptographic keys to cryptographic functions outside of the TPM. In that case, the keys can be protected by the TPM using a mechanism called *sealing*, where keys or other secrets are stored in the Device's file system in an encrypted file that can only be decrypted with keys released from the TPM when pre-defined criteria are met.

Sealing is a mechanism provided by TPMs to encrypt small data objects (typically symmetric keys) using a key that's kept in the TPM. Sealing and Unsealing offer two capabilities:

NOTE: The object to be encrypted by the TPM is unstructured; that is to say, the TPM regards the object simply as a string of bytes. In TCG documents, this data object is referred to as a 'blob'.

- The object to be encrypted is not stored in the TPM: it's sent to the TPM along with keying and access information, using a "Sealing" operation, and the TPM returns the object encrypted, where it can be stored in the host platform's file system.

- The sealed object can be unsealed when returned to the TPM, assuming a number of conditions are met:
    - The relevant asymmetric key must be resident in the TPM
    - Authorization to use the parent key must be present
    - In addition, the sealed object would usually require that one or more of the PCRs have a particular value, indicating that the platform is in a known state.[19]

If the conditions are met, the TPM will return the decrypted blob, which the system designer can use as a structure containing symmetric keys for decrypting a larger object. Alternately, the decrypted blob could contain keying material for use in high-throughput accelerators or other applications.

The system designer has complete freedom in choosing which, if any, PCRs should be consulted prior to unsealing the object. Here are two examples:

1. Some secrets need to be accessed just once, early in the boot sequence of a platform. In that case, the system designer could allocate one PCR specifically for this use, and seal the secret against this PCR with a value of zero (the value after reset). After retrieving the secret, the system designer could extend that PCR with an arbitrary value, preventing the object from being unsealed during the same boot cycle by a hacker.
2. The system designer could seal the secret against a number of PCRs that represent the exact versions and configurations of software booted on the box, preventing the secret from being revealed unless the OS is in the exact right configuration. This approach needs to be used with caution, as a software update that's not flawlessly staged could render the secret permanently unintelligible.

Note that sealing can be done not just against PCRs, but against any other policy values that a TPM can evaluate, including the state of GPIOs, values stored in NV, or signed data from sensors.

DICE engines can also provide for an encryption key that is available only when the system is in a particular state, as defined by the exact set of firmware and software that it is running.

Additionally, Opal drives can have sections created that can only be opened and read when a certain key is available. This key can be sealed to the TPM, thus allowing for sections of the drive only being readable when a system is in a defined state.

6.5 Protection of Software and Configuration Data

As described in section 5.5, Industrial Control Systems often contain sensitive data that must be protected. For example:

- Device Manufacturer software and data may need to be protected from anyone else (including the Owner) seeing or changing it
- Owner software and data may need to be protected from anyone else (including the Device Manufacturer) seeing or changing it

In this case, the Device Manufacturer or Owner may wish to sign software and configuration files and configure the device to only accept software and configuration files signed by the authorized Device Manufacturer or Owner. In some cases, it may be desirable to prevent rollback to previous versions of configuration files. Software and configuration signatures may be checked periodically or as needed and can even be attested to so that the Device

[19] Technically, an object can be sealed without requiring reference to any PCR, but that would have the same effect as a simpler encrypt/decrypt operation.

Manufacturer or Owner can verify that the software or configuration has not been modified. Attestation of software and configuration is covered in section 6.13.

To protect ICS configuration files against disclosure, these files may be encrypted in transit, at rest, and even in use. Some ICS systems could benefit from full-disk encryption as described in TCG's Opal specifications to protect data at rest, to discourage cloning. The document TCG Storage Opal Integration Guidelines [15] gives an overview of the Trusted Computing Group suite of specifications for self-encrypting storage devices. Essentially, Opal provides a standard for protecting portions of the storage device. This data is encrypted while stored, using cryptographic keys stored in the drive. The encrypted data can only be accessed if the proper authentication keys are presented. This provides strong protection against unauthorized access to data at rest.

If a TPM is on the system, an asymmetric TPM key can be generated and used to sign configurations that the owner wishes to authorize, and then check those signatures during the boot sequence. Physical hijacking of such a solution is possible, but it could be made resistant against software attacks. Encryption of data at rest can also be accomplished with a TPM by encrypting the data with a key protected by the TPM. Alternatively, data can be stored directly in the Platform hierarchy (for manufacturer configuration data) or Storage hierarchy (for owner configuration data) with appropriate policies to restrict access to the data.

If a DICE is on the system, then software could be encrypted using a key derived in part from the configuration approved by the administrator, so that the system would not boot if the configuration were changed. Note that this could lead to a denial of service attack.

Organizations wishing to enforce proper configurations may want to create configuration files centrally, and distribute them to ICS devices in a way that they can't be changed in transit or moved from one ICS device to another device that is not authorized to receive the file.

Transport-level security (e.g. a TLS session authenticated with a DevID) would normally be used with network-connected devices to ensure that the right configuration goes to the right device, and this works fine with a TPM or DICE. However, this requires a connection between the configuring system and the device being configured. An alternate approach that can work across an air-gap is to encrypt and sign configuration files so they can only be decrypted and used on the intended ICS device.

This can be accomplished if the organization managing the ICS devices creates a configuration file for each device, and encrypts each one using a symmetric key, which in turn can be encrypted using the public portion of an IDevID or LDevID key corresponding to the desired device. Authenticity can be assured by signing the package.

Once installed on the device, the configuration file can be decrypted using the TPM_UnBind() (for TPM 1.2) or TPM2_Decrypt() (for TPM 2.0) commands to decrypt and deploy the file.

Maintainers will need to build in enough agility in the provisioning process to encrypt configuration files with new keys when fielded units must be replaced.

6.6 Physical Attacks

Protection against physical attacks requires an assessment of each system in order to determine the necessary protection to obtain an acceptable level of risk. Further complicating the issue is the fact that the level of risk from the compromise of one Industrial Control System does not necessarily translate to all other such systems. Access Control, as discussed in section 6.2, in conjunction with trusted technologies, becomes the key to mitigating as many physical attacks as possible.

Full categorization of physical attacks against Industrial Control Systems is beyond the scope of this document and will likely vary based on the application, threats, and value of the assets. Determining the proper categorization system is something that must be done by System Integrators and System Owners. However, the use of TCG technologies can help to mitigate physical attacks against systems by protecting critical data.

Inclusion of a TPM inside systems provides a cryptographic element that can provide protection of critical data, as well as attestation of the identity of the system itself. Keys used by the TPM can be stored internally and never released outside the TPM, thereby preventing an attacker with physical access from exfiltrating key values. However, data that leaves the TPM may need to be protected with other measures (e.g., buried traces or encrypted sessions) if physical attack is a concern.

Using encrypted sessions between the CPU and TPM requires the CPU and TPM to share a secret, which is not easy for embedded systems. One option here is to have the CPU and TPM store or derive a long-term shared secret. Another option is to perform an encrypted Diffie-Hellman exchange using each other’s identities.

Self-Encrypting Drives (SEDs) can also be used to maintain system information and can be cryptographically tied to the TPM. The use of this approach prevents an attacker from being able to remove physical storage elements from the system and extract sensitive information. Such SEDs may also include their own physical attack mitigations to prevent unauthorized access.

6.7 Licensing for Feature Authorization and Product Differentiation

To properly implement feature licensing, it’s important to have a mechanism that can securely enable a specific feature on a specific instance of the product, in a way that the same feature can’t be enabled on similar products without authorization.

The Device Manufacturer may wish to enable installation and activation (or deactivation) of a feature in the field, given adequate authentication. An expiration date for a feature may also need to be provided, in case the feature was only purchased for a limited time.

Feature activation often must happen across an “air gap”, i.e., the device on which features are to be enabled may be housed in a facility without access to the public internet.

The actual features to be enabled within this use case are out-of-scope, but one example might be newly optimized robot programming.

TPM based policies can allow this, even allowing use of a feature for a specified amount of time, or a specified number of times. To implement this, a policy can be tied to a key or GPIO using the TPM2_PolicyAuthorize command or with NV PIN indexes to allow remote configuration / change to a policy (not available on TPM 1.2).

Industrial Device Manufacturers often supply a device with a base feature set, and an additional list of features or capabilities that may be available for a license fee, or may be subject to various legal restrictions. In these cases, it’s desirable to enable these features or capabilities on some devices, and not on others, and to ensure that the capability is only enabled by a Maintainer authorized by the Device Manufacturer.

The following is a possible scenario for this use-case, to activate a feature in a fielded device:

1. The Maintainer goes to a web portal run by the Device Manufacturer.
2. The Maintainer proves administrative rights over the device to the Device Manufacturer through the portal. This mechanism is out-of-scope, but might involve (for example) proving possession of a valid purchase order or access code for the device.
3. The Maintainer selects the desired feature from a menu of possible choices and may need to pay for the feature at this time.
4. The Device Manufacturer approves installing the desired feature on the device. For security features, this may involve checking business rules or import/export controls.
5. The Device Manufacturer provides a ‘token’ that authorizes specifically this feature on only this device. This procedure must ensure that the token is useless on any other device.

6. The Maintainer applies the token to the industrial device, perhaps after carrying it across an air-gap.
7. The Industrial Device activates the feature.

While the steps involved might be different for different classes of devices, TCG technology can be used as follows:

- Device identity can be proven to the Device Manufacturer, using an IDevID certificate.
- Information in the TPM can also be used to ensure that the Token can only enable the desired feature on the particular Device.

One notable aspect of this use case is that the Maintainer may have an interest in enabling unauthorized features on the device. In some cases, the Maintainer might just try to use the feature without paying for it. In other cases, the Maintainer might be prohibited from obtaining access to certain features due to legal restrictions such as import/export controls (e.g. strong encryption). As a result, feature authorization may need to resist determined efforts by the Maintainer to enable unauthorized features.

One further requirement may be to enable features on “air gapped” industrial devices, requiring a mechanism that does not depend on network connectivity. In the air-gapped environment, it must be possible to ship the authorization package as one or more files (for example, on a CD).

Two approaches may be taken to implementing feature authorization on an industrial device with a TPM. Approach 1 does not attempt to resist determined owner attacks. Approach 2 does. Therefore, approach 1 may be more suitable to low-cost features without legal constraints.

1. In step 5) above, the token is created by concatenating a feature identifier and IDevID public key, then signing that concatenated data structure using a Device Manufacturer Licensing private key used for feature authorization. The Token would also include the signature of the feature authorization and the Device Manufacturer licensing certificate. Optionally, the Token can also include a start date, end date, maximum number of users, or other supplementary information relevant to the feature. Figure 6 illustrates the token:

   $$\boxed{[\text{FeatureID, Limitations, IDevID Cert}]_{\text{LicensingKey}}, \text{LicensingCert}}$$

   Figure 6 - Token for Feature Authorization

   When the device receives the token, it can verify the Device Manufacturer licensing certificate and signature to make sure the token is an authorized license token from the Device Manufacturer. Then it can compare the IDevID public key against its own device identity to make sure that the license is for this device. If both of these checks pass, the device can enable the feature, subject to any limitations (e.g., maximum number of users) included in the token.

   This technique does not make much effort to resist determined owner attacks. While the owner cannot easily fabricate or modify a token, the owner has physical access to the device and therefore a sophisticated and determined owner may be able to use reverse engineering and physical attacks to fool the device into enabling a feature without having a valid token. For example, the owner could tamper with program code after boot-time code verification to skip the token verification check or cause this check to succeed when it should fail (enabling an improperly signed token to be used). Or the owner could tamper with a token after the token verification check to change the feature being authorized or to remove limitations.

2. In a more sophisticated approach, software code that’s needed to implement the feature is sent down to the device as a part of the license activation process. This ensures that unlicensed devices cannot be easily tampered with to enable unauthorized features.

There are many ways to implement this approach depending on how the Device Manufacturer has implemented software updates. For example, the token from approach 1 can be modified to add a software identifier that is then downloaded and installed through the Device Manufacturer's update process to enable the feature.

However, the software download is implemented, it should attempt to ensure that the relevant software does not fall into the hands of unauthorized parties. The software should be encrypted while transiting untrusted networks and preferably encrypted while stored in long-term storage on the device making it difficult for unauthorized parties to get a copy of the software and share it with others.[20]

The TPM may be very useful in implementing this approach. The encrypted feature software may be distributed to an industrial device encrypted with an ephemeral symmetric key. The symmetric key can be encrypted with a public key whose corresponding private key is held by the TPM. This whole bundle can be delivered to the device via any mechanism, along with the token described above. The decrypted symmetric key can be obtained from the TPM using the TPM_UnBind() command with TPM 1.2 or TPM2_Unseal() for TPM 2.0.

Subsequently, when the software is stored in local long-term storage, it may be encrypted using a key that's sealed to the TPM on this device and also to a set of PCRs, thus ensuring that the encrypted file cannot be decrypted on another system. If the chosen PCRs correspond to an early stage in the boot sequence and the unsealed key is wiped from memory immediately after use, the encrypted file cannot be decrypted at a later stage in the boot sequence (where run-time infections often take place).

Using this approach, a Maintainer who does not have a token for his own devices doesn't even have a copy of the code needed to implement the feature authorized by that token. This makes it substantially more difficult for the Maintainer to tamper with the device to enable the feature in question.

Note that TPM 2.0 also provides a feature called an NV PIN. An NV PIN can be set up to allow any other TPM entity (key, GPIO, etc.) to be used for only a set number of times. Once that number is exceeded, the entity will no longer be usable, until the NV PIN is reset (by the policy of the NV PIN). The NV PIN policy can be restricted in numerous ways, via logical combinations of TPM policies. The NV PIN can be used to restrict the number of times that a particular feature is used, which might be especially useful to enable restricted usage of a feature during a trial mode.

6.8 Equipment-As-A-Service

TPMs are ideal for implementing equipment-as-a-service, as described in section 5.8. NV counter functionality can be used to count the number of times a service is used, and then quote that value remotely (with TPM2_NV_Certify), or they can use an NV_PIN mechanism to pre-authorize usage of a TPM controlled resource a number of times, by counting the number of times that service has been successfully authorized.

In the former case, software needs to be developed that will increment the NV counter whenever a usage takes place (e.g. a page is printed). Additionally, software needs to be developed that uses TPM2_NV_Certify to remotely tell the provider of the service what the value of the counter is.

In the latter case (using the NV_PIN to pre-authorize usage of a resource), an NV_PIN is created by the manufacturer. NV_PINs have 4 important features.

- The contents of the NV_PIN are broken into two of the important features.

[20] Although not practical for most devices, the software could be encrypted whenever it is not loaded on a tamper-resistant secure processor.

    - The most significant 4 bytes contain the number of pre-authorized authorizations using the NV_PIN's password.
    - The least significant 4 bytes contain the number of successful authorizations using the NV_PIN's password.
- The password of the NV index that hosts the NV_PIN is the third important feature.
    - It can be a NULL password.
    - It cannot be used to modify the contents of the NV index that hosts the NV_PIN
    - It can be used to authorize other features of a TPM including
        - Use of an asymmetric key
        - Use of (or release of) a symmetric key
        - Changing the value of a GPIO (which can flip a switch)
- The policy of the NV index that hosts the NV_PIN is the last of the four important features
    - Set by the manufacturer in this case
    - Becomes part of the name of the index
        - This prevents the deliberate deletion and recreation of the NV index in an attack.
    - This allows changing the contents of the NV index
        - resetting the number of uses
        - resetting the limit of the number of uses

To use this for a service level agreement, if the user buys the right to use software 10 times, the service provider would set the limit in an NV_PIN_PASS to be 10. The software would then use the NV_PIN_PASS's password to decrypt a key, which the software then uses to perform the service. Once the service is completed, the decrypted data and key are destroyed from memory. Alternatively, a GPIO that is connected to an actuator would be turned on by using the NV_PIN_PASS's software, and after a set amount of time, would be turned off.

If the device is intermittently connected or not connected at all, the NV_PIN method may be more attractive because it does not require connectivity to a remote server.

6.9 Counterfeit Detection

Equipment used in Industrial Control System must leverage mechanisms that allow for easy identification and attestation of equipment. Use of an IDevID that can be created in the factory or created in the field and validated with the EK and the Platform Certificate supports this functionality. Consideration should be given to both the circumstance of the IDevID creation, as well as how the validation occurs.

For creation, it is highly recommended that whenever possible this occurs within a controlled and trusted environment. This scenario provides stronger confidence in the equipment and helps to minimize counterfeit issues by ensuring only parts provisioned within this trusted environment are allowed access to their associated networks or devices. In cases where creation of the IDevID must occur in the field, it is highly recommended that physical presence at the device be a part of the validation process. Such a requirement will help to alleviate the potential for arbitrary insertion of counterfeit devices into critical networks or systems.

In determining the correct method for validation, System Owners must determine what level of validation is acceptable. Use of the IDevID alone is simple, straightforward, and provides information on the Manufacturer identity. A Platform Certificate provides information beyond merely the Manufacturer, but rather helps to establish trust in the entire platform itself. Although both contain the device serial number, Manufacturer identity, and Manufacturer signature, the Platform Certificate provides information related to the TPM and Trusted Building Block (TBB) used in the system. This adds additional complexity, but provides a higher security solution.

Additionally, once the device identifier (IDevID or Platform Certificate) is installed on a device, it must be protected from direct software access. Should an attacker be able to gain execution on a system that has already been provisioned and authenticated, extraction of this information may allow them to repurpose a malicious device with this same information and then physically exchange them. As such, the associated information must be protected from direct access by software. The use of a TPM or other secure storage element will help to alleviate such attacks.

Including a TPM in Industrial IoT products makes it easier to detect counterfeit products. As products get smarter and are more connected, they often include a security chip such as a TPM that provides device identity along with other features. The TPM can be provisioned with a public-private key pair and signed manufacturer's platform certificate vouching that this part comes from a particular manufacturer. If the platform certificate chains up to a trusted root CA and the private key is stored in tamper-resistant storage, the customer can verify that the part is authentic (not counterfeit). For products that pass through a long supply chain or are assembled from multiple modules, customers may want the ability to verify the complete supply chain or the origin of the modules.

DICE can be used in low cost devices, in a similar way. DICE allows for a certified IDevID or LDevID to be created with a device that allows a device to identify itself. The resulting identity should always be validated to ensure the process was not hijacked at any point or spoofed by a malicious attacker. As before, it is recommended that the creation of the certified IDevID or LDevID occurs within a trusted facility or under physical supervision if occurring in the field.

6.10 Remote Device Management

A management station should be able to perform checks at any time to verify the identity, software and configuration of each managed device. Classic issues like missed updates and deviations in configuration parameters can be detected in an automated way, making established management systems more secure in the following ways:

- Trusted Computing allows monitoring and verifying the state of the devices against expected values documented in the management system. Deviations can be reported automatically to the Administrator using Remote Attestation and the PTS protocol suite (see *TCG Attestation PTS Protocol: Binding to TNC IF-M* [10]), reducing operational costs to the customer.

  Remote Attestation can also increase confidence in the supply chain, by proving that devices have the correct software loaded.

- Using a DevID to authenticate the device tightens the ability of the management station to check the inventory of devices deployed. Unauthorized or changed devices can be detected without manual checks of the physical devices. DICE provides a similar key, unique to a device, such as a sensor.

- Secure communications between the device and the management station are essential for safe remote management. A DevID or DICE key can be used to establish a secure communication channel using TLS or other similar protocols.

- Using the TPM in the ICS for signature verification on updates and commands that claim to have come from the Manufacturer and are being passed on by the Owner through their management system. The Owner can also use a TPM in an Engineering Workstation (for example) for signing or countersigning sensitive updates (i.e., control logic updates) and commands to indicate that they are coming from or authorized by the Owner itself. Using a TPM to protect the keys used for signing and verifying commands and updates can prevent attackers from stealing the signing key, modifying the verification key, or using either one without authorization.

- The PTS specification [10] allows a remote device to attest to the state of the system as recorded in the PCRs of a TPM or implicitly with DICE. This attestation is secured by Roots of Trust, and relies on digital signatures and freshness attestation. This provides feedback to the owner so he knows if a feature needs to be updated or if it has been updated. By daisy chaining from a Root of Trust to trusted measuring software, this provides a strong method for measuring the state of a system.
- The TPM2_PolicyAuthorize command together with TPM_PolicyCPHash provides a means for a remote manager to update configurations stored in TPM NV, in a way that does not provide the end user a means to change that configuration. Together with PTS or TPM2_NV_Certify, this provides a means for an end user to both provide updates and also determine if they have been applied.

6.11 Software Inventory

Mechanisms can be implemented to allow a Maintainer to query devices to discover which revision level of which software and firmware are installed on each device in their network and to be able to rely on the responses received. The TPM and TNC standards are especially well suited to this task as a TPM with a strong RTM (Root of Trust for Measurement) can provide a reliable software inventory, even if the software is old or compromised. DICE is designed to do this as well.

The International Standards Organization (ISO) has published document ISO/IEC 19770-2:2015 *Information technology -- Software asset management -- Part 2: Software identification tag* [6] describing the format for Software Identification tags (SWID tags), stored in XML files on a device. SWID tags encode version information and expected hashes for each software package, and should be updated by the device manufacturer's software update process each time a new image or patch is installed.

SWID tags can be retrieved from a device to determine which software was last installed on the device. Although techniques beyond the scope of this document must be used to determine whether the software installed is the version expected, comparison of the hashes in the SWID tags with hashes retrieved from the TPM would provide assurance that the installed software matches a valid release from the manufacturer.

While SWID tags are simply files and can be retrieved with any number of different mechanisms, the TCG Trusted Network Communications workgroup has incorporated SWID tag retrieval into the TNC protocol suite. Specified in *SWID Message and Attributes for IF-M* [31], the TNC protocol allows a management station to determine whether each network device is loaded with the expected firmware or software release.

The presence of a particular SWID tag indicates which software version is probably installed. For purposes of attestation, the SWID tags must be signed by the network device manufacturer, or potentially by the Maintainer of the device or other organizations, providing confidence that the hashes claimed by SWID values are the ones that should be present on the device.

SWID tags may contain hashes of the software modules they describe: with careful alignment by the network device manufacturer, these hashes in SWID tags can be used in some cases as "known good values" for Attestation, and can be compared to what was recorded in the TPM using a measured-boot process.

This allows a degree of automatic checking as part of the Attestation procedure described in Section 6.12:

- Use the TNC protocol described in *SWID Message and Attributes for IF-M* [31] to retrieve signed SWID values showing what should be installed on the device
- Use TNC protocol *TCG Attestation PTS Protocol: Binding to TNC IF-M* [10] to retrieve signed PCR values from the TPM, along with boot logs, to determine what was actually loaded during the boot process.

Signatures, module names and hashes can then be compared by the external Attestation Server to detect tampering.

The following documents provide details on SWID tags:

- *ISO/IEC 19770-2:2015(en) Information technology — IT asset management — Part 2: Software identification tag; [6]*
- *Guidelines for the Creation of Interoperable Software Identification (SWID) Tags*; NIST IR 8060 [32]
- *SWID Message and Attributes for IF-M* [31]

TCG, ISO, and IETF may update these documents but these versions can be used now.

6.12 Attestation of Boot Integrity for Devices ("Health Check")

The correct initialization of an industrial control system should include verification of key aspects of hardware, firmware and software whose execution sequence is required to bring the industrial control system to its operational state prior to its execution. Once verified, no unauthenticated changes to the fundamental hardware, firmware, and software are allowed.

The boot state of the platform may be verified via remote attestation or local controls.

Remote attestation is a mechanism by which measurements stored in a TPM can be retrieved by a remote attestation server to determine the configuration of the device at boot time as well as at runtime. In the case of industrial devices, remote attestation might be invoked by a management station (called the Challenger or Verifier), causing an industrial device to report its boot (and runtime) configuration. Alternatively, two peer devices might do Mutual Attestation to prove to each other that their configurations are acceptable.

Attestation of both boot time and run time configuration can include a wide range of objects that might have a bearing on the security posture of a device.

Boot time objects typically include

- The initial firmware loaded (such as UEFI code)
- Options of the initial firmware
- Boot loader
- OS kernel

When extended to run time, Attestation can include

- Executables launched by the OS,
- Shared libraries loaded into memory for execution,
- Policy or configuration files
- Scripts
- Cryptographic key material and credentials used by the OS, or other relevant sources of state information.

As a result of an attestation process, the Challenger will receive a cryptographically signed "quote" and log from the Device indicating which versions of which modules were loaded and run as the device started up.

Attestation requires that the Device implements "Measured Boot" (Section 4.2), in which, starting at a Root of Trust for Measurement (RTM), each stage computes a hash of the next stage and stores the result in a TPM Platform Configuration Register (PCR) before launching it. Each stage of the process can also measure any ancillary files that may be of interest in determining the device's security state, as listed above. Along with extending measurements into PCRs, device software is expected to keep a log file which contains the name and version of the object along with the hash. The log file does not need to be secure: the management station can confirm its contents by computing the hash of the hashes and comparing the result to the signed PCR values. While the contents of the log file are not specified, the log data structure used by PTS is defined in Section 3.25 of *TCG Attestation PTS Protocol: Binding to TNC IF-M* [10], and could serve as a template.

To obtain attestation information, the Challenger must establish the identity of the Device, and then request the signed quote plus log information. This can be done using the TCG Platform Trust Services (PTS) protocols, defined in *TCG Attestation PTS Protocol: Binding to TNC IF-M* [10].

Attestation has several preconditions:

- The Device must extend the hash of each boot object into the appropriate PCR, and keep the corresponding log entries, keeping in mind, due to boot time requirements, that some parts of the log may be generated after the extend operation. If runtime integrity must be attested as well, the Device must extend the hash of each runtime object into the appropriate PCR, and also keep the corresponding log entries.
- The Device must have an Attestation Key (either IAK or LAK) configured, to allow it to sign the quote.
- Either a Privacy CA must be used to certify an AK, or the AK should be otherwise traceable to the device's identity.[21] (See Sections 6.1.2.3 and 6.1.3.2.)
- The device must implement the PTS protocol (or equivalent) to deliver attestation information.

TCG specifications allow the Challenger to obtain attestation information from the Device, but it's the Challenger's task to determine whether the hashes are acceptable. Acceptable hashes for software modules are called Reference Integrity Metrics (RIMs) in TCG documents[22], and:

- May be obtained directly from the Device Manufacturer.
- Could be learned from 'known-good' systems.
- Could be retrieved from the device under attestation, assuming the reference measurements are signed by the entity that's authorized to provide software for the product.

Extending Measured Boot into the OS requires additional OS kernel functionality, allowing for recording of measurements before software execution or reading[23] of files identified as security-critical. Each file identified for analysis is measured and recorded in the TPM, with a corresponding a measurement log entry. Unlike earlier stages of the boot process, OS measurements may continue as long as the system runs, resulting in incremental changes to logs and PCRs as different objects are accessed through the system's life.

6.12.1 Linux Integrity Measurement Architecture (IMA)

In Linux, a part of the security sub-system called the Integrity Measurement Architecture (IMA) carries out the measurement process. IMA extends the principle of Measured Boot into the operating system by providing means

[21] Devices in which the AK and DevID are not tightly bound are subject to the "Asokan Attack" (RFC 6813) where an infected device tricks a clean device into attesting on its behalf. Mechanisms such as TPM_Certify and SKAE can be used to provide proof that the AK and DevID are resident in the same TPM.
[22] For TPM 1.2, see *Reference Manifest (RM) Schema Specification [37].*
[23] Examples of simple file reads that could have integrity implications are configuration files or interpreted-language scripts such as shell or Python. These aren't strictly executable, but they may be able to trigger unauthorized execution.

to measure selected applications started by the OS. System designers and Maintainers can benefit from the following:

- Measurement of any software executable on the system as it is loaded into memory for execution, including shared libraries.
- Measurement of all files read by a particular user, including programs running with permissions of that user.
- Measurement of all files used by specific packages, such as database systems or web servers.

The system designer must decide on the OS and application software and other objects to be measured. Executable objects would typically be measured, but measurements may also include any other kind of object such as those listed in Section 6.12 above. IMA maintains a measurement log, referred to as Integrity Measurement Log (IML) in TCG documents[24].

IMA has many configuration options:

- IMA log formats are configurable using *IMA templates*, allowing a system designer to adapt the log format to his/her needs.
- IMA is configurable using a policy, which to a certain extent allows for definition of files to be measured. IMA uses that policy to decide whether a file is to be measured or not according on each file open(). The policy can be hard-coded and compiled into the Linux kernel or it can be set dynamically on every boot by writing it to a file in the Linux *securityfs* filesystem (*/sys/kernel/security/ima/policy*). It is advisable to always set the policy as early as possible in the boot process, although Linux kernels of version 4.7 and later allow the policy to be updated after initial boot.
- IMA also supports policies that are based on Linux Security Modules (LSM) rules. LSM provides an interface for security technologies to extend the Linux kernel's capabilities with more security-related features. It is used by technologies such as SELinux, Simplified Mandatory Access Control Kernel (SMACK), TOMOYO, AppArmor, Yama, and IMA, as well as others[25]. IMA is usually used to measure immutable files, so to exclude all log files—which are, by definition, mutable—the SELinux attribute "logfile" can be used to tag these, allowing the definition of a policy rule to exclude files that are tagged with that attribute. Of course, a Maintainer must create a policy rule to set that attribute on all log files. The same applies to scripts and other interpreted executables. Using LSM rules a Maintainer can improve the measurement process and define more fine-grained IMA policies by including and excluding files depending on their expected usage.

However, system designers need to be aware of some limitations of IMA, and some impact on the overall system, including the following:

- Files that are accessed by IMA for measuring may cause measurement violations due to concurrent exclusive file accesses, called the Time of Measure Time of Use (ToMToU) problem[26]. Whenever IMA cannot gain read access to a file, a measurement violation is indicated by a special log entry, containing a file hash of all zeros, but, in turn, the PCR gets extended with a hash of all ones. System designers may wish to flag that situation when analyzing log files.
- Binding the platform state to an IMA-maintained PCR might not make a lot of sense in most cases. As program execution order, even during boot, is non-deterministic on most modern operating systems, the

[24] also known as Stored Measurement Log (SML) in some documents
[25] See https://wiki.gentoo.org/wiki/Extended_Verification_Module.
[26] If a file that is opened exclusively for use/writing (at the time of use), the measurement may fail, as the measurement process cannot also open the file for reading in order to measure it (at the time of measurement).

sequence of measurements in the log is very likely not to be the same across boot cycles. This will result in unpredictable PCR values, even if the set of executed programs has not changed.

- Verification of IMA measurements requires more complex verification at the management station. Instead of just comparing a single PCR value to a known-good value, the management station must calculate the accumulated hash over the Integrity Measurement Log and compare it to the reported hash from the TPM Quote. Furthermore, the management station needs to compare all entries of the measurement log to previously defined reference measurements, RIMs. That is, it must have access to hundreds or even tens of thousands of reference values for a particular system configuration. Typically, a search-optimized (indexed) database is used.
- As the IMA measurement process continues to operate during the retrieval of a TPM Quote, there might be some mismatch between the last recorded entry in the PCR and the last entry in the log, when retrieving it. The log might have advanced while the "quote" operation was running. Accordingly, the appropriate log entry—the last one that was incorporated in the PCR—must be identified to allow the Challenger to verify the log. The process for identifying that log entry is:
  1. Retrieve the log for the first time, L1 (with the number of entries |L1|[27]).
  2. Generate the TPM Quote.
  3. Retrieve the log again, L2 (with the number of entries |L2|).
  4. Check whether the log has advanced (|L2| > |L1|). If so, then the log entry that has been used for the TPM Quote must be identified, which can either happen on the device under attestation itself or on the management station, which, in turn, needs either both of the log files or just the second one (L2) and additionally the number of entries in L1 (|L1|). The process would look something like what follows:[28]
     a. Calculate the accumulated hash from the whole first log (L1).
     b. Compare the accumulated hash against the hash from the TPM Quote. If it matches, then return the full log up to the current entry. Otherwise, continue.
     c. Update the accumulated hash by incorporating the next log entry from list L2. Then continue with step b.
- In an OS with multiple concurrent processes, the TSS Access Broker should be used to manage concurrent access from multiple processes to a single TPM.[29]
- While the file hashes are collected, none of the common file metadata is logged by IMA, so there's no record of user ID or group ID, either of the file itself or the process accessing it, or file attributes.
- IMA's policy language cannot distinguish between text files, log files, executable scripts, and other files. IMA can just detect access to files other than executables. All kinds of interpreted languages are affected, such as Java, Python, Ruby, Lua, shell scripts, etc. Depending on the number of files involved, IMA's facilities for measuring every file can be used. The Challenger then is responsible for identifying relevant files, such as scripts. As IMA keeps an internal hash table to avoid adding duplicate entries to the log, measuring a lot of files may cause that hash table to consume a notable amount of kernel memory and may cause out-of-

[27] |Lx| indicates the number of entries in the log.
[28] While the order of entries in the log may be non-deterministic, this document assumes that the log file contains entries in exactly the same order as they were measured into the TPM.
[29] For example, see *TSS 2.0 TAB and Resource Manager Specification* [38].

memory errors. LSM rules in an IMA policy, as mentioned above, can be used to restrict measurement only to files that have security significance.

IMA is subject to continuous improvement, and may offer additional desirable features that are not described in this document.

6.12.1.1 IMA Appraisal

IMA Appraisal is the equivalent of Secure Boot extending into the operating system. In contrast to IMA measurements, IMA Appraisal requires a file to match a known-good measurement in order to allow access to that file.

Known-good measurements can be managed several ways.

One approach is to simply learn the hash of each file from a clean system. For that purpose, a particular boot argument to the Linux kernel must be passed (*ima_appraise=fix*), instructing IMA to record the known-good measurements of accessed files. On any subsequent boot, IMA appraisal can be activated to permit access to a file only if it matches the known-good measurement, by passing the corresponding kernel boot argument (*ima_appraise=enforce*).

By default, IMA allows access to files if the measurements match and will update the known-good measurement if the file is modified after having passed the check on the initial file-open.[30]

IMA also supports *immutable* files utilizing digital signatures on the known-good measurements, generated and signed when the software modules are built, prior to installation on a particular machine. Private keys stay with the build system: only the signer's public key(s) must be available in the IMA keyring (currently *.ima_root_ca*), although they must be loaded into the keyring early in the boot process, typically in an *initramfs*.

Digital signatures provide protection against online tampering with the files. Offline tampering would still be possible by generating a malicious key-pair, re-signing all the files with the malicious private key, and then replacing the public key with the malicious one. It's possible to protect against offline changes to extended file attributes in general, including the attributes used for IMA Appraisal Known-Good Measurements, with the Linux kernel Extended Verification Module (EVM)[31]. With EVM, IMA can use a cryptographic hash (HMAC) or digital signature, with a key loaded at boot time, to verify the extended file attributes on any access. EVM keys can be protected by the TPM using Sealing, or in applications where there's an interactive user, a password can be used to unlock the EVM key.

A system designer may utilize IMA Appraisal whenever a system must run only approved software that is known prior to device deployment. While systems that use signatures generated by the device manufacturer's software build system can readily be updated in the field (assuming keys don't change), the software update mechanism on systems that require re-measurements of software on the system itself is out of scope for this document and is left to be defined by the system designer.

6.12.2 PCR Definitions

The allocation of various functions to specific PCRs depends a lot on the underlying software architecture of the industrial device.

There are two definitions available at the time of this publication to serve as starting points for industrial devices:

- UEFI-based *TCG_PC Client Implementation for BIOS* Section 3.3.3 [35]
- Mobile Platforms: *TPM 2.0 Mobile Common Profile* Section 2.5 [18]

[30] https://wiki.gentoo.org/wiki/Integrity_Measurement_Architecture#Registering_the_file_hashes_for_the_system
[31] See https://wiki.gentoo.org/wiki/Extended_Verification_Module

Guidelines for information to be collected in PCRs can also be found in the following:

- *NIST SP800-155 BIOS Integrity Measurement Guidelines* [24]

6.13 Local Verification of Load-Time and Run-Time Integrity of Code and Configuration on Devices

Load-time verification of application code (e.g., Linux IMA) is essential to system security. Prior to running the application code, the application must be verified using an immutable process that cannot be interrupted and is time sensitive in case an error occurs. It is recommended that such a process is itself verified and immutable. The verification process or secure loader typically is located in ROM that is part of the device mask set. The secure loader code is called using a set of parameters that is signed, maintaining the chain of trust. The parameters include the location of the code, the signature of the code, the target run-time address to store the code, and an optional decryption key and encryption method if the code is encrypted. Optionally, these parameters can be encrypted using public key encryption.

The Secure loader reads the application code into its control memory area and verifies the signature. If the signature verification succeeds, then the application code ID is decrypted (if necessary), stored at the indicated address location, and the OS is notified that a new application is available to run.

Run-time verification is like the load-time process except for decryption. As best practice, the Run-Time verification process is set in motion by an independent timer. This process reads the Application code and calculates the signature that is then compared with the signature referenced in the parameter file. If the signature matches, a “signature good” notification is sent to the OS. If the signature fails to match, a “signature bad” notification is sent to the OS.

Remote attestation of the local verification process can prove that local verification has been done. Among the ways to achieve this is to extend the results of the local verification into a PCR and perform a remote attestation protocol that delivers a quote of this PCR to a remote verifier.

6.14 Inventory of Composite Devices

In modular industrial devices, there’s usually one or more processor cards that manage field replaceable units (FRUs) like I/O packs or modules. I/O packs can sometimes be “hot swapped”, that is, inserted and removed while the rest of the system continues to run, yielding a system where modules may come and go on a regular basis, even though the system as a whole rarely goes down or starts up all at once. This results in several complexities:

- Each time an FRU is inserted or removed, a process running on the processor card typically notes the change in configuration, initializes the new unit and may alert other modules in the system of the new configuration. These new modules should not be admitted to the system unless they are authentic (not counterfeit) and in a known-good state.

- The conventional definition of attestation may need to be augmented with the goal of reporting not just the attested state of the main processor, but including also the state of the subsidiary FRUs, each with their own processor state.

- Today, industrial FRUs do not typically contain a TPM. That limits the ability of these FRUs to reliably attest to their software inventory. However, they may still be able to report on their software inventory even without the additional assurance that can be provided by a TPM.

TCG offers a number of mechanisms that are useful to the design of such systems:

- **Integrity-Protected Logs** (described in section 6.15.2) could be used to keep a tamper-evident record of modules that were added or subtracted during the life of the system.

- **Identity -** The system designer must set the rules and procedures for determining membership in the composite device, but Device Identity (e.g., IDevID or LDevID) could provide a solid basis for determining whether specific devices are eligible to join a composite or not.
- **Internal Attestation** among elements – System designers may want elements of a system to prove not only their identity, but also that they're running authorized software.
- **External Attestation** – Complex composite systems may elect a leader, either through hardware or software means, to communicate with the outside world and represent that state of the system. One of the functions that this leader could provide is Remote Attestation, where an external management device can identify the system, its modules and the authenticity of software running on the system, including all the software running on subsidiary units which may not be visible to an external management agent.

Although these techniques are not common practice in industrial systems today, they can provide real benefit in improving the security of composite systems in the future.

6.15 Integrity-Protected History

As noted in section 5.15, industrial systems often include an operational historian function that records history of operational events. This information can be used to provide evidence of error, inefficiency, malfeasance or attack. Sophisticated attackers may attempt to cover their tracks by altering the recorded history. Therefore, the integrity of this history must be protected. Protecting the integrity requires evidence of the authenticity, order, and time of the log file entries.

6.15.1 Functional Requirements

Integrity-Protected logged data must be protected against undetected manipulation such as deletion, insertion of modification of log events, and can additionally provide evidence of the state of the system at the time of the logged event.

Each entry in the log file provides the following information:

- Type of event
- Date and time
- Order relative to other events

The TPM-issued signature covers the following information, showing it has not been modified after being signed:

- Identity of the log device, based on the DevID
- A non-volatile monotonic sequence number showing the order of events and proving that the current event immediately follows the previous event, so that causal relationships can be determined, and deletion of events can be detected
- (optional) time of signature
- (optional) state of device during signature operation

Industrial devices may reboot unexpectedly, or they may operate without a reboot for years: as such, logging mechanisms must typically meet two additional goals.

- Log files must persist through a reboot, and retain integrity through the reboot.
- But at the same time, devices may produce a lot of logging information, so mechanisms that "trim" the logs, discarding old entries, are often provided.

The need to discard old log information implies that there can be no expectation of a permanent log that goes all the way back to the first use of the Device: it's the Maintainer's responsibility to collect and archive log information periodically if a long-term record is needed. The intent of the mechanism described in this section is to allow the Maintainer to demonstrate that a set of log files provides a complete record.

6.15.2 Integrity-Protected Log Implementation

Implementation of Integrity-Protected Logs uses a number of TPM mechanisms:

- The TPM's time-keeping mechanism can be used to generate tamper-resistant time stamps on log entries.
- The TPM's Monotonic Counter mechanism can be used to sequence log entries and to detect missing or added entries, even across boot cycles
- The TPM's signing mechanism can be used to sign log entries so that any changes in an individual entry can be detected.

6.15.2.1 Timekeeping in TPM 1.2

While the TPM does not keep a real-time clock, it does provide a mechanism that keeps track of elapsed time since the last initialization. This relative time measure is called a Tickstamp (*TPM 1.2 Part 1 Design Principles* [25] Chapter 20).

The tick counter increments at a specified rate, but cannot be "set", and does not relate to any external real time, so there is usually a requirement to establish the correlation between the monotonic tick-counter of the TPM and a trustworthy external real time clock.

The correlation process works by measuring the difference between an authorized external time source and the local Tickstamp timer. A Tickstamp is taken prior to the request to an external source, and then again after the response has been processed, yielding a bounded uncertainty in time correlation. The internal tick counter of the TPM is subject to drift, so correlation may have to be repeated periodically to maintain adequate sync. (See *TPM 1.2 Part 1 Design Principles* [25] Section 20.3 for a detailed description.)

Figure 6 shows the process:

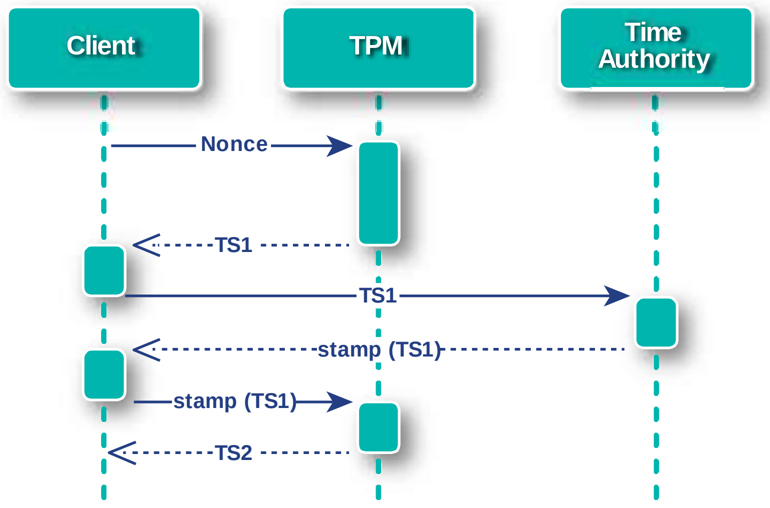

Figure 7 - Time Correlation with TPM

The TPM supports the generation of signed tick stamps that include a signature over user-supplied data associated with the current tick count value, the tick increment rate and tick session nonce. Every time the TPM is reset a new tick session nonce is generated. (See *TPM 1.2 Part 1 Design Principles* [25] Section 20.1 for a detailed description.)

6.15.2.2 Timekeeping in TPM 2.0

TPM 2.0 offers additional timekeeping facilities, described in *TPM 2.0 Part 1 Architecture* [3] Chapter 36. *Time* contains the time in milliseconds since the last startup of the TPM. *resetCount* is a counter that increments every time the TPM is successfully reset. Its purpose is, among other things, to indicate possible discontinuity in *Clock* (described in the next paragraph.) *restartCount* is used to provide an indication of discontinuities due to (1) TPM Resume, (2) TPM Restart, or (3) *_TPM_Hash_Start*; all allowing TPM *Time* to fall behind real time (*TPM 2.0 Part 1 Architecture* [3] Chapter 36.6).

Clock is a volatile value that increments in memory each millisecond. The non-volatile value *NV Clock* is updated periodically from that value. On every reboot the value of *Clock* is set to the value of *NV Clock*. However, in case of unexpected power loss the final *Clock* value might not have been written to *NV Clock*. The TPM flag *safe* of the *TPMS_CLOCK_INFO* structure is used to indicate such a situation of unexpected power loss. Mitigations are described in *TPM 2.0 Part 1 Architecture* [3] Chapter 36. In contrast, an orderly shutdown of the TPM is possible using the *TPM2_Shutdown()* command which tells the TPM to preserve an appropriate state.

Neither the TPM 1.2 nor the TPM 2.0 has a real-time clock that is able to advance time in the power-off state. The TPM 1.2 clock retains no state through a power cycle, but in TPM 2.0, *Clock* is non-volatile, and can be set forward (never backward, though, except by installing a new owner) by external software managing the TPM, allowing for correlation to an external real-time clock. As the TPM may be driven by an imprecise frequency source, the clock is subject to drifting. To compensate, the TPM 2.0 allows external software to adjust the rate of advancement by +/- 15 %.

Attestation data in TPM 2.0 includes the *resetCount* value, although it is obfuscated in some cases to preserve privacy (*TPM 2.0 Part 1 Architecture* [3] Chapter 36.7). Nevertheless, an attesting management station can detect a reboot, and hence a possible discontinuity in time, by checking whether the *resetCount* value has changed. TPM 2.0 also allows for usage of timing data in policies, e.g., to limit the usage of keys for a certain amount of time.

6.15.2.3 Assembling the Log

While this document does not specify the exact format of Integrity-Protected log files, the logs can be assembled using TPM Monotonic Counters for sequencing, TPM signing for integrity, and Tickstamps for temporality.

To detect addition or deletion of events in the log, TPM Monotonic Counters (*TPM 1.2 Design Principles* [25] Chapter 17, *TPM Rev 2.0 Part 1 – Architecture* [3] Section 37.2.4) can be used. These counters can be incremented by a TPM user, but can never be decremented or cleared. This mechanism allows log entries to be created where each entry has a sequence identifier that's exactly one larger than the previous entry, allowing deletions or additions to be easily detected. Old log entries may be deleted, but in the remaining logs, the monotonic count should increment uniformly from the oldest entry through each intermediate entry to the newest.

For high-value, low-rate[32] log information, a simple mechanism can be used:

- Each new log entry can be assembled with a Tickstamp or Timestamp to show when it happened, and the next value from a non-volatile Monotonic Counter
- The log entry can be signed with an IDevID or LDevID key, and appended to the log file.
- Reboots of the system and restarts of the TPM can be detected using the tick session nonce of a Tickstamp (TPM 1.2 and TPM 2.0) or the values *resetCount* and *restartCount* (TPM 2.0 only).

Any subsequent tampering will invalidate signatures, or show a gap in the sequence numbers.

[32] Average rate of a few log entries per hour: TPMs can't sign quickly, and the Monotonic Counter in TPM 1.2 will wrap and/or wear out if it's incremented more than once every few minutes.

It should be noted that the TPM also incorporates a mechanism that can be used to audit the command codes it has been called upon to execute: this mechanism works by keeping an internal digest of commands, which can be compared to an external log file of commands that were executed.

High-rate logs would require a more complex block-signing mechanism, something beyond the scope of this document.

6.16 Entropy Generation

Many network protocols rely on a dependable source of random numbers for correct operation. For security related protocols, random numbers may be used by a cryptographic element for functions such as key generation.

The TCG TPM definition includes a requirement for a Random Number Generator, which can be accessed after the TPM self-test sequence is complete, simply by invoking the TPM_GetRandom() or TPM2_GetRandom() ordinals.

TCG specifications allow TPM Manufacturers considerable flexibility in implementation choices. A common design approach, not mandated by TCG, would be a mechanism where a physical noise source of some sort (a True Random Number Generator, or TRNG) can be used to seed a Deterministic Random Bit Generator (DRBG), ensuing predictable statistical properties.

For TPM vendors, compliance to standards such as *NIST Special Publication 800-90A/B/C* [12] or *BSI AIS-31, A Proposal for Functionality Classes for Random Number Generators* [13], may be a requirement; in this case, the Device Manufacturer should consult the TPM Manufacturer for compliance certifications.

There are two aspects of the standards that Device Manufacturer designers should consider:

1. The standards require continuous testing of the parts of the RNG, in addition to a self-test at startup. These tests must be done inside the TPM, as they require access to internal state. If there is a failure of the self-test at any point, the TPM will report a failure in response to further requests.
2. The standards require that entropy from a physical source (TRNG) should be added periodically to the DBRG, so that repeated reads will not exhaust the entropy supply.

A compliant TPM is required to monitor and replenish the entropy level as needed, and may return fewer bytes of random numbers than requested should requests outpace generation of new entropy.

Rates at which new entropy is generated within the TPM is not specified by TCG or the standards, so it's up to the Device Manufacturer and the TPM Manufacturer to ensure that the TPM in consideration generates entropy at an adequate rate for the application.

The TPM has a mechanism with which external entropy can be added to the TPM's DRBG output (TPM_StirRandom() and TPM2_StirRandom()). As long as the TPM Manufacturer certifies that the device generates entropy on its own at an adequate rate, the Device Manufacturer is not likely to need this ordinal.[33]

It's always good practice for the Device Manufacturer's operating system to collect and merge entropy from as many sources as are available. For example, the Linux OS rng-tools can be configured to merge entropy from a number of sources such as arrival time of network traffic, or disk seek times, with the entropy obtained from the TPM to supply /dev/random for OS and application needs.

6.17 Deprovisioning

As described in section 5.17, Industrial Control Systems often contain sensitive information that an Owner would want to protect from disclosure, and which must be destroyed when the device is de-provisioned.

[33] If a platform has additional sources of entropy, StirRandom can also be used to defend against a hidden failure of the TPM's internal entropy source.

The Device Manufacturer should provide a mechanism to enable de-provisioning.

Upon de-provisioning, the device:

- Should delete any LDevID, LAK or other customer-generated keys from the TPM
- Should *not* destroy IDevID or IAK keys, since these indicate the manufacturer's device identity, not the Owner's.

Any installation-specific configuration files should also be deleted.

Deletion of files from modern flash-based memory systems is not easy: an approach to ensure that deletion is reliable would be:

- Create a local storage key as a child of the SRK
- Store all Owner configuration files in a partition using disk-encryption technology.

Upon de-provisioning, the TPM can be instructed to delete the local storage key, rendering the partition unintelligible. This is easy to do with the TPM2_Clear command, which will delete everything underneath the Owner Hierarchy. While IDevID, EK or IAK keys will no longer be available to the TPM, they can be easily regenerated, because the Endorsement hierarchy seed is not changed by this operation.

Opal drives can be completely de-provisioned by using the Revert or RevertSP commands to clear all provisioning of the drive and change the default encryption keys underlying all data stored on the drive. At this point, any data stored on the drive will be non-recoverable, and the drive will need to be re-partitioned and formatted before it can be reused. Then the operating system and software will need to be reloaded. This is an extreme option and usage should be limited.

Another option is to create multiple regions (LBA ranges) within the Opal drive and carefully place Owner data into one or more regions that can be erased separately using a Maintainer password. However, this requires careful effort to ensure that Owner data is not placed into non-Owner regions. One way to reduce this concern is to create a recovery partition in a region that is normally read-only. When the industrial device is de-provisioned, all other regions are wiped by using the GenKey method to change the encryption key and then the recovery partition can be used to restore the other regions to a de-provisioned state.

6.18 Protecting Secrets and Intellectual Property

The TPM can separate and protect assets with different owners using its multiple hierarchies. For example, a cryptographic key stored in the Platform hierarchy can be used to decrypt software written by the Device Manufacturer and to authenticate to the Device Manufacturer. Likewise, a cryptographic key stored in the Owner hierarchy can be used to decrypt software written by the System Owner and to authenticate to the owner's network and management system.

The TPM can also restrict access to assets based on the exact software loaded. PCRs change during the boot sequence. As a result, if a TPM object (such as a key) is locked so it can only be used with a set of PCRs that are in a prescribed state early in a boot sequence, that key (or NV RAM index, etc…) can only be used / accessed / written during that period of time when the PCRs are in that state. This can be used to restrict usage of an object to BIOS or a particular OS load.

6.18.1 Examples

Often a specified action is controlled by a particular user (or set of users), and a policy can be made that allows any of those users to perform the action via their ISC device. However, in the case of an emergency, or unavailability of an authorized user, an administrator will need to authorize more users to perform the action. Additionally, if a user leaves the company or is moved to a different job, his or her access must be removed. Both of these management functions can be performed using the TPM2_PolicyAuthorizeNV command or (with a little more difficulty) with the TPM2_PolicyAuthorize command. In the former case, an administrator uses a key to sign a new policy for the action in question, and places it in a specified non-volatile (NV) location in the TPM. Thus, when a disgruntled employee leaves the company, his access to control various machines in a manufacturing plant can be immediately removed.

Using a GPIO control, a lockout mechanism can be created that does not allow power to various functions in a device if the box is opened - unless a qualified repairman overrides this using his key. This can prevent life threatening “repairs” made by unqualified personnel.

6.19 Secure Update of Software and Configuration

Secure update requires not only technical controls but also proper processes and trained staff to ensure that only properly approved updates are applied. From a technical perspective, updates must be authenticated, integrity protected, and authorized for installation on a particular device. The entire update process from creating the software or configuration through installation and verification must be tightly controlled to ensure that one rogue person or infected machine cannot spread malware. For industrial devices, these tight controls are especially important due to safety and reliability concerns.

Hardware security can play an essential role in implementing secure updates for industrial systems. For example, keys for code signing, verification, and decryption can be stored in either the Engineering Workstation or the Device’s TPM to prevent software compromise from permitting the modification or disclosure of those keys. Access to less sensitive keys may be controlled via simple authorization such as HMAC. Access to especially sensitive keys (e.g., code signing keys) can be restricted with policies requiring multiple authorization values and PCR values to enforce requirements for multi-factor or even multi-party authorization and prevent malware from using the keys. Some software updates (e.g., firmware) may require a combination of Manufacturer and Owner authorization while other software updates (e.g., ladder logic) may only require Owner authorization. Authorization to install an update may be restricted to a particular model of device or to a single device.

PCRs in a TPM can be used with policies to create a window of time during which NV indexes can be either read or written (or both). This window in time starts when the value of PCRs matches a certain value (such as a value they attain sometime during a boot sequence) and ends when one of the PCRs to which the NV index is locked changes. Due to the nature of PCRs, they will never re-attain the initial values during the same power cycle. In this way, an update engine can be made privy to decryption keys stored in a particular NV Index or granted the ability to change values stored in particular NV Indices, while preventing any other software from having that ability.

For more comprehensive information on implementing secure updates, see the TCG document titled *Guidance for Secure Update of Software and Firmware on Embedded Systems* [19].

6.20 Mitigating the Risks of Legacy Systems

Legacy systems may be suspected or known to be vulnerable but cannot be upgraded or replaced. Therefore, these systems must be protected against attacks using other means.

One common technique for protecting legacy systems is to isolate them on a separate network. IEC 62443 recommends that security practitioners group sensitive resources into “zones” based on sensitivity and communications needs, then connect these zones with secured “conduits” that protect inter-zone communications against attacks. For an overview of this technique and how TCG standards can help to secure such a system, see the *TCG Architect’s Guide on ICS Security Using TNC Technology* [36] and the *TNC IF-MAP Metadata for ICS Security* [37].

Note that isolation is not an impenetrable defense. The Stuxnet [46] attack and the 2015 attack on the Ukrainian power grid [45] illustrate this. In both cases, attackers were able to penetrate supposedly isolated networks and infect systems on those networks, leading to failures. For this reason, network isolation should be viewed as necessary but insufficient for protecting legacy systems. Additional measures such as strict controls on and monitoring of isolated networks and connected devices should be used to detect, mitigate, and respond to attacks even on isolated networks.

6.21 Disconnected Operation

Disconnected operation presents special challenges for some of the use cases listed above. Others will be unaffected by lack of connectivity. The following are lists of use cases that fall into each of these categories.

6.21.1 Largely Unaffected by Disconnected Operation

These use cases are largely unaffected by lack of external connectivity:

- Access Control is unaffected when local means are used, such as TPM policies
- Securing Secrets, Protection of Software and Configuration Data, and Protecting Secrets and Intellectual Property typically do not require external connectivity
- Counterfeit Detection generally does not require external connectivity, although such connectivity can be useful for checking revocation status when Device Identity is used to verify authenticity
- Remote Device Management, Software Inventory, Inventory of Composite Devices, and Attestation of Boot-Time, Load-Time, or Run-Time Integrity do not require external connectivity, as long as the management station is connected to the device being managed. However, updates to the list of acceptable hashes may require some form of external communication.
- Integrity-Protected History does not require external connectivity if the logs are kept locally.
- Entropy Generation does not require external connectivity
- Deprovisioning typically does not require external connectivity, unless storage is completely wiped and must be reimaged

  NOTE: Most systems for Handling Legacy Systems are able to work without external connectivity because many legacy systems are installed in disconnected environments.

6.21.2 Affected by Disconnected Operation

These use cases are affected by lack of external connectivity:

- Device Identity may encounter difficulties with identity provisioning and revocation status checking unless some communication with the CA and revocation authority is permitted. If these features are not needed, Device Identity is not affected by disconnected operation.
- Secure Zero Touch Provisioning methods generally support disconnected operation using local servers or storage. These mechanisms should be used in disconnected situations.
- Licensing for Feature Authorization and Product Differentiation generally functions well in a disconnected mode, except that changes in licensing require external communication.
- Equipment-as-a-Service often depends on connectivity for billing. However, it can operate in a disconnected mode if a counter is used (e.g., using NV_PIN). In that case, some form of external connectivity is needed to recharge the counter.
- Secure Update of Software and Configuration requires some form of external connectivity to obtain the update images. However, this can be done with USB drives.

6.22 Usable Security

TPMs can help provide usable security in several ways. TPMs can be used to effectively convert weak passwords into strong ones. One example of this is the use of a “weak” password to authenticate to the TPM, allowing the Device or User to use a cryptographically strong key for Device or User authentication (such as FIDO) while the TPM’s dictionary attack mechanism hinders a rogue endlessly trying to guess a weak password, until they succeed. Another way is to use cryptographic key fobs such as FIDO keys for authentication. The TPM2_PolicySign command allows keys and other TPM objects to require such fobs for authentication.

7 REFERENCES

[1] IEEE Computer Society, *802.1AR-2009 – IEEE Standard for Local and Metropolitan Area Networks - Secure Device Identity*, New York, New York, 22 December 2009, https://1.ieee802.org/security/802-1ar-2009

[2] IEEE Computer Society, *802.1AR-2018 – IEEE Standard for Local and Metropolitan Area Networks - Secure Device Identity*, New York, New York, June 14, 2018, http://www.ieee802.org/1/pages/802.1ar.html

[3] Trusted Computing Group, *Trusted Platform Module Library, Part 1: Architecture*, Family "2.0", Level 00 Revision 01.59, November 2019, https://trustedcomputinggroup.org/wp-content/uploads/TCG_TPM2_r1p59_Part1_Architecture_pub.pdf

[4] Trusted Computing Group, *Trusted Platform Module Library, Part 2: Structures*, Family "2.0", Level 00 Revision 01.59, November 2019, https://trustedcomputinggroup.org/wp-content/uploads/TCG_TPM2_r1p59_Part2_Structures_pub.pdf

[5] Trusted Computing Group, *TPM Keys for Platform Identity for TPM 1.2*, Version 1.0, Revision 3, 21 August 2015. https://trustedcomputinggroup.org/wp-content/uploads/TPM_Keys_for_Platform_Identity_v1_0_r3_Final.pdf

[6] The International Organization for Standardization/International Electrotechnical Commission (ISO/IEC), *Information Technology - Software Asset Management - Part 2: Software Identification Tag*, ISO/IEC 19770-2, October 2015, https://www.iso.org/standard/65666.html

[7] Trusted Computing Group, *TCG TPM v2.0 Provisioning Guidance*, Version 1.0, Revision 1.0, March 15, 2017, https://trustedcomputinggroup.org/wp-content/uploads/TCG-TPM-v2.0-Provisioning-Guidance-Published-v1r1.pdf

[8] Trusted Computing Group, *TCG EK Credential Profile For TPM*, Family 2.0, Version 2.4, Revision 3, July 2021. https://trustedcomputinggroup.org/wp-content/uploads/TCG_IWG_EKCredentialProfile_v2p4_r3.pdf

[9] Trusted Computing Group, *TCG Infrastructure Workgroup Subject Key Attestation Evidence Extension*, Specification Version 1.0, Revision 7, 16 June 2005, https://trustedcomputinggroup.org/wp-content/uploads/IWG_SKAE_Extension_1-00.pdf

[10] Trusted Computing Group, *TCG Attestation PTS Protocol: Binding to TNC IF-M*, Specification Version 1.0, Revision 28, August 24, 2011, https://trustedcomputinggroup.org/wp-content/uploads/IFM_PTS_v1_0_r28.pdf

[11] *A Practical Guide to TPM 2.0*, W. Arthur & D. Challener, Apress; 1st Edition, January 28, 2015, https://www.apress.com/gp/book/9781430265832

[12] National Institute of Standards and Technology, *Recommendation for Random Number Generation Using Deterministic Random Bit Generators*, Revision 1, June 2015, http://dx.doi.org/10.6028/NIST.SP.800-90Ar1

[13] Bundesamt für Sicherheit in der Informationstechnik (BSI) AIS-31, *A Proposal for Functionality Classes for Random Number Generators*, Version 2.0, 18 September 2011, https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/AIS_31_Functionality_classes_for_random_number_generators_e.pdf

[14] ISO/IEC 18031:2011 *Information technology -- Security techniques -- Random bit generation*, published on: 2011-11, http://www.iso.org/iso/catalogue_detail.htm?csnumber=54945

[15] Trusted Computing Group, *TCG Storage Opal Integration Guidelines*, Version 1.00, Revision 1.00, March 16, 2016, https://trustedcomputinggroup.org/wp-content/uploads/TCG_Storage_ReferenceDocument_Opal_Integration_Guidelines_v1.00_r1.00.pdf

[16] *Linux Integrity Measurement Architecture*, http://linux-ima.sourceforge.net, https://wiki.gentoo.org/wiki/Project:Integrity

[17] Trusted Computing Group, *IWG Reference Architecture for Interoperability (Part 1)*, Version 1.0, Revision 1.0, June 16 2005. https://trustedcomputinggroup.org/wp-content/uploads/IWG_Architecture_v1_0_r1.pdf

[18] Trusted Computing Group, *TPM 2.0 Mobile Common Profile*, Family “2.0” Level 00 Revision 31, 21 December 2015, https://trustedcomputinggroup.org/wp-content/uploads/TPM_2.0_Mobile_Common_Profile_v2r31_FINAL.pdf

[19] Trusted Computing Group, *Guidance for Secure Update of Software and Firmware on Embedded Systems*, Version 1.0 Revision 72, February 2020, https://trustedcomputinggroup.org/wp-content/uploads/TCG-Secure-Update-of-SW-and-FW-on-Devices-v1r72_pub.pdf

[20] Trusted Computing Group, *TCG Platform Attribute Credential Profile*, Version 1.0 Revision 16, January 16, 2018, https://trustedcomputinggroup.org/wp-content/uploads/TCG-Platform-Attribute-Credential-Profile-Version-1.0.pdf

[21] NIST Special Publication 800-57 Part 1, *Recommendation for Key Management*, Revision 5, May 2020. https://doi.org/10.6028/NIST.SP.800-57pt1r5

[22] Trusted Computing Group, *Hardware Requirements for a Device Identifier Composition Engine*, Family “2.0”, Level 00 Revision 78, https://trustedcomputinggroup.org/wp-content/uploads/Hardware-Requirements-for-Device-Identifier-Composition-Engine-r78_For-Publication.pdf

[23] Trusted Computing Group, *Implicit Identity Based Device Attestation*, Version 1.0, Revision 0.93, https://trustedcomputinggroup.org/wp-content/uploads/TCG-DICE-Arch-Implicit-Identity-Based-Device-Attestation-v1-rev93.pdf

[24] National Institute of Standards and Technology, *BIOS Integrity Measurement Guidelines*, NIST SP 800-155 Draft, December 2011, https://csrc.nist.gov/publications/detail/sp/800-155/draft

[25] Trusted Computing Group, *TPM Main, Part 1: Design Principles*, Version 1.2 Revision 116, March 2011, https://trustedcomputinggroup.org/wp-content/uploads/TPM-Main-Part-1-Design-Principles_v1.2_rev116_01032011.pdf

[26] Internet Engineering Task Force, *Enrollment over Secure Transport*, RFC 7030, October 2013, https://tools.ietf.org/html/rfc7030

[27] Internet Engineering Task Force, *The Network Endpoint Assessment (NEA) Asokan Attack Analysis*, RFC 6813, October 2013, https://tools.ietf.org/html/rfc6813

[28] National Institute of Standards and Technology, *Digital Identity Guidelines: Authentication and Lifecycle Management*, NIST SP 800-63B, June 2017, https://doi.org/10.6028/NIST.SP.800-63b

[29] Internet Engineering Task Force, *The Transport Layer Security (TLS) Protocol Version 1.3*, RFC 8446, August 2018, https://tools.ietf.org/html/rfc8446

[30] Internet Engineering Task Force, *Secure Zero Touch Provisioning (SZTP)*, RFC 8572, April 2019, https://tools.ietf.org/html/rfc8572

[31] Trusted Computing Group, *SWID Message and Attributes for IF-M*, Version 1.0, Revision 29, 3 August 2015, https://trustedcomputinggroup.org/wp-content/uploads/SWID_Messages_For_IFM_v1r29.pdf

[32] National Institute of Standards and Technology, *Guidelines for the Creation of Interoperable Software Identification (SWID) Tags*, NIST IR 8060, April 2016, http://dx.doi.org/10.6028/NIST.IR.8060

[33] Trusted Computing Group, *Reference Manifest (RM) Schema Specification*, Version 2.0, https://www.trustedcomputinggroup.org/tcg-reference-manifest-rm-schema-specification/

[34] Trusted Computing Group, *TSS 2.0 TAB and Resource Manager Specification*, Family 2.0, Level 00, Version 1.0, Revision 18, April 2019, https://trustedcomputinggroup.org/wp-content/uploads/TSS_2p0_TAB_ResourceManager_v1p0_r18_04082019_pub.pdf

[35] Trusted Computing Group, *TCG PC Client Specific Implementation Specification for Conventional BIOS*, Specification Version 1.21 Errata Revision 1.00, February 24, 2020, https://trustedcomputinggroup.org/wp-content/uploads/TCG_PCClientImplementation_1-21_1_00.pdf

[36] Trusted Computing Group, *TCG Architect's Guide on ICS Security Using TNC Technology*, October 2013, https://trustedcomputinggroup.org/resource/architects-guide-ics-security-using-tnc-technology

[37] Trusted Computing Group, *TNC IF-MAP Metadata for ICS Security*, September 2014, https://trustedcomputinggroup.org/wp-content/uploads/IFMAP_Metadata_For_ICS_Security_v1_0r46.pdf

[38] Trusted Computing Group, *TCG Glossary*, May 2017, https://trustedcomputinggroup.org/resource/tcg-glossary

[39] International Electrotechnical Commission (IEC), *IEC 62443 - Security for industrial automation and control systems*, https://webstore.iec.ch

[40] Internet Engineering Task Force, *An Internet Attribute Certificate Profile for Authorization*, RFC 5755, January 2010, https://tools.ietf.org/html/rfc5755

[41] Internet Engineering Task Force, *PA-TNC: A Posture Attribute (PA) Protocol Compatible with Trusted Network Connect (TNC)*, RFC 5792, March 2010, https://tools.ietf.org/html/rfc5792

[42] FIDO Alliance, https://fidoalliance.org

[43] Trusted Computing Group, *Architect's Guide: Cybersecurity*, https://trustedcomputinggroup.org/resource/architects-guide-cybersecurity/

[44] Trusted Computing Group, *Architect's Guide: Data Security Using TCG Self-Encrypting Drive Technology*, https://trustedcomputinggroup.org/resource/data-security-using-tcg-self-encrypting-drive-technology/

[45] SANS ICS Report, *Analysis of the Cyber Attack on the Ukrainian Power Grid*, March 2016, https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2016/05/20081514/E-ISAC_SANS_Ukraine_DUC_5.pdf

[46] The Langner Group, *To Kill a Centrifuge: A Technical Analysis of What Stuxnet's Creators Tried to Achieve*, https://www.langner.com/to-kill-a-centrifuge/

[47] SANS ICS Defense Use Case, *ICS CP/PE case study paper - German Steel Mill Cyber Attack*, December 2014

[48] Trusted Computing Group, *TCG PC Client Platform Firmware Profile Specification*, Family 2.0, Level 00, Revision 1.05 Revision 23, https://trustedcomputinggroup.org/resource/pc-client-specific-platform-firmware-profile-specification/

[49] Trusted Computing Group, TPM 2.0 Keys for Device Identity and Attestation, Version 1.00, Revision 12, https://trustedcomputinggroup.org/wp-content/uploads/TPM-2p0-Keys-for-Device-Identity-and-Attestation_v1_r12_pub10082021.pdf

[50] US Presidential Executive Order: "Improving the Nation's Cybersecurity", https://www.federalregister.gov/documents/2021/05/17/2021-10460/improving-the-nations-cybersecurity

[51] TCG Guidance for Securing Network Equipment: Version 1.0, Revision 29, Jan 17, 2018, https://trustedcomputinggroup.org/wp-content/uploads/TCG_Guidance_for_Securing_NetEq_1_0r29.pdf

8 GLOSSARY

This document uses a few terms that may not be widely known:

- Device – a unique piece of industrial equipment such as a PLC
- From IEEE 802.1AR:
    - **Initial Secure Device Identifier (IDevID)**: The Secure Device Identifier installed on the device by the manufacturer.
    - **Locally Significant Secure Device Identifiers (LDevIDs)**: A Secure Device Identifier credential that is unique in the local administrative domain in which the device is used.
    - **Secure Device Identifier (DevID)**: A device identifier that is cryptographically bound to the device and is composed of the Secure Device Identifier Secret and the Secure Device Identifier Credential.

TCG-specific acronyms and abbreviations can be found in the *TCG Glossary* [42].

9 APPENDICES

9.1 Items For Further Study

9.1.1 Virtualization in Industrial Systems

Manufacturers are starting to incorporate Virtual Machine technology into Industrial Control Systems, either as an element of an integrated hardware/software product or as a software-only product running within a virtual machine environment. This may be especially useful for establishing Software Defined Networks (SDN) and Network Function Virtualization (NFV), which permit rapid changes to network configuration such as establishing new security protections or zones.

In either case, there's often a need to identify the hardware resource underlying the virtual machine, to ensure that the function is actually executing in the expected environment, and to determine whether the software is running in a virtual machine or directly on a physical machine.

9.2 Standardization Work For Further Study

9.2.1 Reference Manifests

The document *TCG Infrastructure Working Group Reference Manifest (RM) Schema Specification* [37] defines the XML schema with which integrity information is communicated between entities for TPM 1.2.

There currently is no equivalent document for TPM 2.0. Further work is needed in this area.