TRUSTED® COMPUTING GROUP

GUIDANCE

Overview of TCG Technologies for Device Identification and Attestation

Version 1.0
Revision 1.39
April 29, 2024

Contact: admin@trustedcomputinggroup.org

PUBLISHED

DISCLAIMERS, NOTICES, AND LICENSE TERMS

ACKNOWLEDGEMENTS

The TCG would like to gratefully acknowledge the contributions of the following individuals and companies who volunteered their time and efforts for the development of this specification.

Tom Brostrom, Cyber Pack Ventures, Inc.

Guy Fedorkow (Editor), Juniper Networks

Chris Fenner, Google, Inc.

Jessica Fitzgerald-McKay, United States Government

Ludovic Jacquin, NVIDIA

Darren Krahn, Google, Inc.

Tom Laffey, Hewlett Packard Enterprise

Amy Nelson, Dell

Dennis Mattoon, Microsoft

Graeme Proudler, Invited Expert

Joshua Schiffman, HP Inc

Ned Smith, Intel

Silviu Vlasceanu, Huawei Technologies Co., Ltd.

Monty Wiseman, Beyond Identity

CONTENTS

1 Introduction

As the world becomes more digitized and an increasing number of human-oriented tasks are performed using devices, there is a pressing need for those devices to be provably trustworthy for a wide range of users. To reduce the burden of bringing all these devices online, there's a concurrent evolution to low-touch remote configuration and monitoring, often using cloud-based resources, making it even more important than ever to know which device is which and what they're up to. And when security factors come into play, as might be the case in IoT devices that control physical things, or devices that handle financial transactions, simply asking the device for its name, and whether it's in good shape, without requesting proof, just isn't good enough.

Increasingly, designers are being encouraged to turn to cryptographic device identity and attestation, techniques by which a unique identity is programmed into the device and used along with other mechanisms to report on the health of the device, in a way that's difficult to either change or spoof.

1.1 Scope and Audience

The Trusted Computing Group has developed several technologies that can be used to implement cryptographic device identity and attestation, providing different approaches to the problem for different use-cases and situations, ranging from the smallest IoT devices to the largest computers. This guidance document outlines three of those technologies, and provides some comparison points to give architects and designers of trusted computing components, platforms, and systems a starting point in evaluating implementation directions and security characteristics.

This document covers two interrelated technologies, covered in separate sections below:

- At an externally-visible level, IEEE DevID and the TCG Platform Certificate are standardized mechanisms to expose identity and provenance information about a device,
- At a level normally hidden inside a device, TPM, MARS and DICE can all provide Roots of Trust to validate software integrity and anchor the device identity certificate(s).

1.2 Executive Summary

As an ever-widening segment of society comes to depend on computing devices, technology to improve trust in these devices becomes paramount. This document outlines three such technologies developed by the Trusted Computing Group, which provide a foundation for trustworthy computing, based on Identity and Attestation. The document provides guidelines that device designers, architects and product evaluators can use to determine which technology fits their application.

In this context, Attestation is a process by which a computing device can produce difficult-to-forge cryptographic evidence showing the state of its operational software. This evidence can then be used by an external entity to verify that the device's software is known and expected on this device, comes from a trusted source, and has not been tampered with on the device. Attestation always depends on a "chain of trust" model, in which an implicitly-trusted root can be used to validate the lowest-level software, which in turn can be trusted to validate the next, and so on until the device is fully operational. This document focuses on the Roots of Trust that measure, i.e., create evidence of, that first layer, and that securely report the results. Of course, a critical precept in a chain of trust model is that the first link in the chain, called the Root of Trust for Measurement (RTM), must be trustworthy. Without that first link, subsequent evidence is automatically suspect, and trust cannot be established.

Device identity is an often-overlooked part of the trust model for attestation of the integrity of software on the device. Networked devices are notoriously hard to identify reliably, so if a compromised device can simply take on the identity and evidence-of-integrity of an untampered unit, the proof is of little value.

Both Identity and Attestation always depend on some kind of Root of Trust, and Roots of Trust in computing devices always rely on some kind of hardware to protect critical information, often identified as Shielded Locations and

Protected Capabilities[1]. This document describes three TCG Root of Trust technologies that offer different capabilities and tradeoffs, allowing the device designer to balance a more or less expansive set of Shielded Locations and Protected Capabilities against differing requirements for the hardware required to make the Root of Trust.

Three TCG Root of Trust technologies are described and compared below in this document:

- The Trusted Platform Module (TPM), which provides the most expansive set of Shielded Locations and Protected Capabilities,
- Measurement and Attestation RootS (MARS) provides functionality analogous to a subset of TPM capabilities, considerably slimmed-down to reduce the hardware requirements,
- Device Identity and Composition Engine (DICE) offers just root capabilities for attestation and identity, with significantly reduced hardware requirements. A variant of DICE called DICE Protection Environment (DPE) requires more hardware, but provides better protection against leakage of sensitive information.

A functional summary of the three Root of Trust technologies with variants is given here in Table 1:[2]

Pattern	Summary
TPM	Shielded storage, signing, measurement and attestation primitives as a self-contained crypto-processor
MARS	Measurement and Attestation in a microcontroller IP block
DICE	DICE-Enabled Root of Trust for Measurement (RTM) and Identity
DICE and DPE	DICE-Enabled RTM plus shielded storage

Table 1: TCG Roots of Trust

In addition to requiring hardware, all three technologies assume different levels of software support to provide trustworthy Attestation and Identity as part of a complete system.

Considerable background material on use and function of the Trusted Platform Module can be found in *Trusted Computing Platforms: TPM2.0 in Context* [30].

1.3 Terminology

There are many components to consider in fielding a trusted computing device, from boot loaders to operating systems to applications. Another component of trust is a trusted supply chain, where manufacturers can certify that the product they intended to build is actually the one that was installed at a customer's site. A rudimentary requirement for trust to be established in a given device is that the device must behave *as designed* (no more, no less[3]), that it can fulfill this task without interference, and finally that the device can be identifiable from among a multitude of seemingly identical devices. Thus, to be trustworthy, a device must have a unique identity – the so-called *cryptographic identity* – that permits the device to be reliably identified. As a companion to identity, a trustworthy device should be able to demonstrate that it hasn't suffered from tampering or unexpected software changes.

Elements in the supply chain itself have many tasks, from validating suppliers of electronic components, to ensuring that shipping procedures protect against tampering through many stages of distribution and warehousing, where

[1] A Shielded Location is where sensitive information can be stored safely; a Protected Capability enforces rules on how that information can be accessed or modified.

[2] And repeated in Section 5.6

[3] This doesn't say you shouldn't be astonished by all the awesome & cool features and functionality the designers wedged into your new gadget, but it does say you should be aware of the 'extra' behavior added by that hacker keystroke logger someone slipped in.

each step might be recorded in a cryptographic record. After installation, integrity of the software supply chain can be established through Attestation.

Two types of information go together to establish trust in a device.

Static assertions about the trustworthiness of a device can be made by the manufacturer or other entities in the supply chain. These statements, under the general category of provenance[4], include various assertions about who made the product and what its characteristics are, including:

- A TCG Root of Trust[5],
- Actors in the overall supply chain,
- Device Identity,
- Characteristics of Shielded Locations[6] and Protected Capabilities[7],
- Hardware configuration,
- Platform composition, Compliance to test suites,
- Functional and assurance evaluations,
- … etc…

A second category of assertions cover the current operating status of the device, focusing on mutable components such as software installations that may change during the device's operating life. Reporting this information typically goes under the name Remote Attestation, defined here[8] as the process of creating, sending and checking assertions about mutable platform trustworthiness characteristics, including:

- Software configuration[9]
- Configuration of security-critical controls
- Proof of software integrity as of boot time

Documents on attestation often assume that the goal is to attest the integrity of "a platform", or "a device", without saying exactly what that is, while implying that it's a complete computing device such as a laptop or server. In fact, products are often quite complex, containing many independent computing devices (e.g. disk drive controllers, Baseboard Management Controllers (BMCs), graphics accelerators, etc.), often sourced from different suppliers, each containing its own Root of Trust, and its own verification infrastructure. General discourses on attestation should not be taken to imply that a platform or device contains just one Root of Trust, or that all the components in a computing platform even contain the same kinds of Roots of Trust: each should be tuned to its specific threat model and capabilities.[10]

Within the Trusted Computing Group context, Remote Attestation is the process by which a verifier can obtain cryptographic proof as to the identity of the device in question, evidence of the integrity of software loaded on that device when it started up, and then verify that what's there is what's supposed to be there. For networking equipment, a verifier capability might be embedded in a Network Management Station (NMS), a *posture collection*

[4] i.e., Who all had a hand in making the product and what components were used. See https://csrc.nist.gov/glossary/term/provenance

[5] A Root of Trust is the primal element from which the trustworthiness of the rest of the device can be derived. See the *TCG Glossary* [1] for a formal definition. More information on Roots of Trust can be found in the TCG *TPM 2.0 Mobile Reference Architecture Specification* [7].

[6] i.e., Safe places to store security-critical information such as keys or measurements.

[7] i.e., Code or mechanism used to control access to Shielded Locations.

[8] See TCG Glossary [1] for the formal definition.

[9] i.e., firmware and software. Note that "software configuration" covers both the version of software installed by the system administrator, plus configuration and customization that may have been applied by the system administrator after a software package has been installed.

[10] For example, a complex server might use a TPM to manage operating system secrets, but have DICE and/or MARS to validate firmware executing in peripheral devices such as disk or network controllers.

server[11], or other network analytics tool (such as a software asset management solution, or a threat detection and mitigation tool, etc.).

The IETF Remote Attestation ProcedureS (RATS) Architecture [15] outlines such an attestation environment, and provides a set of definitions for commonly-used terms.

2 Motivation for Roots of Trust for Identity and Attestation

2.1 Application Areas

Trusted Computing Group specifications have long been used to secure conventional computing resources such as personal computers and servers, and have recently been extended to include networking equipment [29] and IoT [28]. However, there are an increasing number of application areas where trusted computing is essential. As examples, here are two emerging areas where cyber risks present threats that go beyond the ordinary risks of identity theft and PC ransomware:

2.1.1 Cybersecurity of Digital Asset Networks

With the global economy becoming increasingly interconnected, there is today a need for the digitization of processes within global trade, supporting a growing number of asset networks. The vision is one where next-generation technological platforms can be developed to incorporate the various technical functions, data-flows and the legal framework pertaining to trade and the financing of trade. Efforts are ongoing to create a digital equivalent of the various forms of documentation for goods that represent the pillars of shipping and trade finance (e.g. Bill of Lading; Letters of Credit)[12].

As digital asset networks increase in economic importance due to the economic value of the instruments traded on these networks, there is a growing need to provide better resiliency of the nodes (e.g. computer systems) of the networks against targeted cyberattacks[13].

Nodes with special capabilities or specific functions – such as those for cross-border or cross-network trade – become increasingly vulnerable as they become valuable “bridges” which interconnect global trade [33].

2.1.2 Internet of Things

The Internet of Things (IoT) has received much attention in recent years, as a wide range of Internet-connected consumer devices like thermostats, cameras and light bulbs are rolled out. Individually, these devices are frequently tightly constrained with little capability for malfeasance that goes beyond annoyance, but taken as a fleet, it becomes clear why even small IoT devices should be protected and attested[14].

A less well-known category of IoT devices is called Industrial IoT (IIoT), including network-connected sensors and controllers that have direct access to industrial processes, such as activating valves in a chemical plant, managing an HVAC system, or controlling motors on an assembly line. IIoT is not always small, low-cost gear – depending on the design, a jet engine or a hospital MRI machine could count as IIoT. In these cases, compromise of IIoT can have direct impact in the physical world[15], making reliable identity and attestation critical.

---

[11] See *Transitioning to the Secure Content Automation Protocol (SCAP)* Version 2 [https://csrc.nist.gov/publications/detail/white-paper/2018/09/10/transitioning-to-scap-version-2/final].

[12] For example, see https://www.tradelens.com/, https://fiata.org/n/digitalizing-the-bill-of-lading/

[13] Cf, W. Sutton, https://www.fbi.gov/history/famous-cases/willie-sutton, “…because that’s where the money is.”

[14] For an example of a hack that organizes a fleet of cheap IP cameras into a botnet, see https://www.trendmicro.com/en_us/research/17/e/persirai-new-internet-things-iot-botnet-targets-ip-cameras.html. That attack depends on easy weaknesses in the target devices, but for a more-sophisticated attack on a well-known IoT device, see https://blog.talosintelligence.com/vuln-spotlight-nest-camera-openweave-aug-2019/.

[15] See *Countdown to Zero Day: Stuxnet and the Launch of the World's First Digital Weapon*, Kim Zetter, Crown; Reprint edition (September 1, 2015).

TCG provides guidance for securing IIoT in *TCG Guidance for Securing Industrial Control Systems Using TCG Technology* [28].

2.1.3 Networking Equipment

Society depends on the Internet, and the Internet depends on the correct functioning of countless numbers of switches, routers and firewalls, each of which relies on an embedded computing environment. Attacks on network infrastructure can cause loss of service, misdirection of traffic, or compromise of secret data.

TCG provides guidance for securing Networking Equipment in *TCG Guidance for Securing Network Equipment Using TCG Technology* [29].

2.2 Deployment Use Cases

Networked devices are often placed in remote and inaccessible locations, where physical inspection is rare and/or difficult. In those situations, a cryptographic device identity may be used for ongoing management and operation of the device, and may be the only way to determine that the correct device is still in place.

Validating identity may be as simple as checking an x.509 certificate, installed by the manufacturer and linked to a key stored in a Root of Trust, to authenticate a TLS connection from a management station to the device.

A standards-based mechanism for doing this is described in IEEE 802.1AR *Secure Device Identity* [37] , usually referenced as DevID.

2.2.1 Secure Zero Touch Provisioning

Device identity is particularly important for devices that are expected to configure themselves automatically when first connected to a network, a process known as Zero Touch Provisioning. ZTP allows device owners to specify in advance how they want each device configured, potentially including the installation of confidential keying material, without relying on a skilled installation specialist to apply the configuration when the device is installed.

There are many variations of ZTP, offering radically different levels of protection for the various kinds of configuration information, some of which may be viewed as confidential. One standardized version of Secure ZTP, which uses cryptographic device identity to ensure that configurations are installed only on the correct device, can be found in IETF RFC-8572 [34].

2.2.2 Monitoring Ongoing Adherence to Security Policies

Remote Attestation provides a mechanism to enable validation of the core security mechanisms (i.e., the Trusted Computing Base), but it can also be extended to report more generally on “code, configuration and credentials”, to provide evidence that the device loaded authorized and unmodified software at boot time, and that the device was configured the way the owner expects.

The *TCG PC Client Platform Firmware Profile Specification* [5], and the IETF Remote Integrity Verification document [31] provide outlines of the information that can be collected from a device with remote attestation, including measurements taken by the Linux file system integrity package Integrity Measurement Architecture (IMA) [38].

2.2.3 Verifiable Provenance of Devices

Determination as to whether a device is trustworthy depends not only on which one is which, but also, how the device made its way from the initial manufacturer to the end user.

NIST Special Publication 1800-34B, *Validating the Integrity of Computing Devices*[16] describes the problem this way:

[16] https://www.nccoe.nist.gov/sites/default/files/legacy-files/tpm-sca-nist-sp1800-34b-preliminary-draft.pdf, Section 1.1

Technologies today rely on complex, globally distributed and interconnected supply chain ecosystems to provide highly refined, cost-effective, and reusable solutions. Most organizations' security processes consider only the visible state of computing devices. The provenance and integrity of a delivered device and its components are typically accepted without validating through technology that there have been no unexpected modifications. Provenance is the comprehensive history of a device throughout the entire life cycle from creation to ownership, including changes made within the device or its components. Assuming that all acquired computing devices are genuine and unmodified increases the risk of a compromise affecting products in an organization's supply chain, which in turn increases risks to customers and end users, as illustrated in Figure 1*. Mitigating this risk is not addressed at all in many cases.*

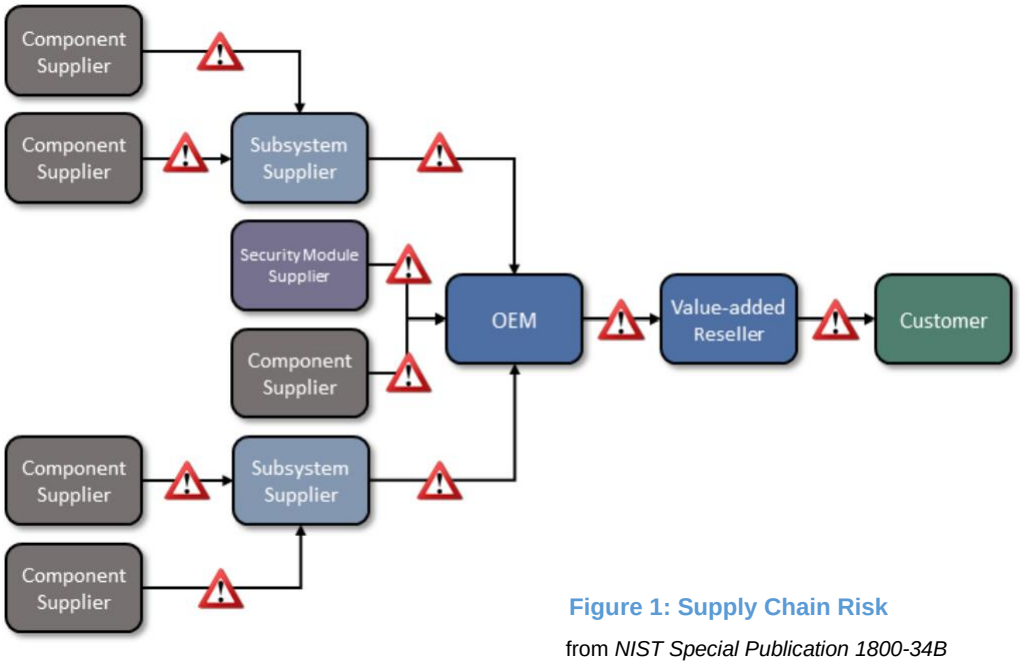

Figure 1: Supply Chain Risk

from *NIST Special Publication 1800-34B*

3 Overview of Identity and Attestation

3.1 Cryptographic Identity

Devices attached to a network always need some kind of identifier by which each device can be addressed uniquely. Often the devices are in remote locations where it's not even possible to physically identify which one is which. Devices often start out being totally identical during manufacturing, but during that process, usually receive some unique identifier (e.g., a MAC address and/or a serial number).

For many devices, that's just a number programmed into flash memory, leaving it subject to being changed or copied, possibly by the user, or probably by a hacker. Even if the identifier is not changed, if it's just a number in non-volatile storage, it's subject to *spoofing*, where an attacker can take over the intended device's identifier, claiming to *be* that device. As long as the attacker produces a faithful copy, the user may not have any way to verify that the device they think they are controlling is actually the one they installed, and not an intruder's device masking as legitimate.

This problem can be solved with cryptographic device identity, a mechanism where each device is shipped with a unique private key that cannot be copied or changed, and a corresponding public key certificate, signed by the manufacturer identifying the device and certifying its authenticity. A device owner is provided with a matching public key[17], which can be used to challenge the device to prove possession of the secret key. From there, the owner can analyze the signature chain to determine whether the device was manufactured by a trusted supplier.

TCG Technology for cryptographic device identity is described in Section 4.

3.2 Remote Attestation

Remote Attestation, proving the state of a remote device, also depends on reliable device identity to block spoofing or adversary-in-the-middle attacks.

Attestation assumes a number of interlocking roles, three of which are highlighted in Figure 2:

- The device that generates evidence (the "Attester" in Figure 2),
- Endorsers, generating Reference Integrity Measurements[18], that are used to determine whether the evidence generated by the Attester shows that the device's configuration lines up with the Owner's expectation,
- And the Verifier, a trusted management agent that analyzes the evidence.

Other roles to complete the attestation picture include policy owners, plus the Relying Party that may take action on an attestation result.

The roles shown in Figure 2 are defined in detail in the IETF RATS Architecture [15].

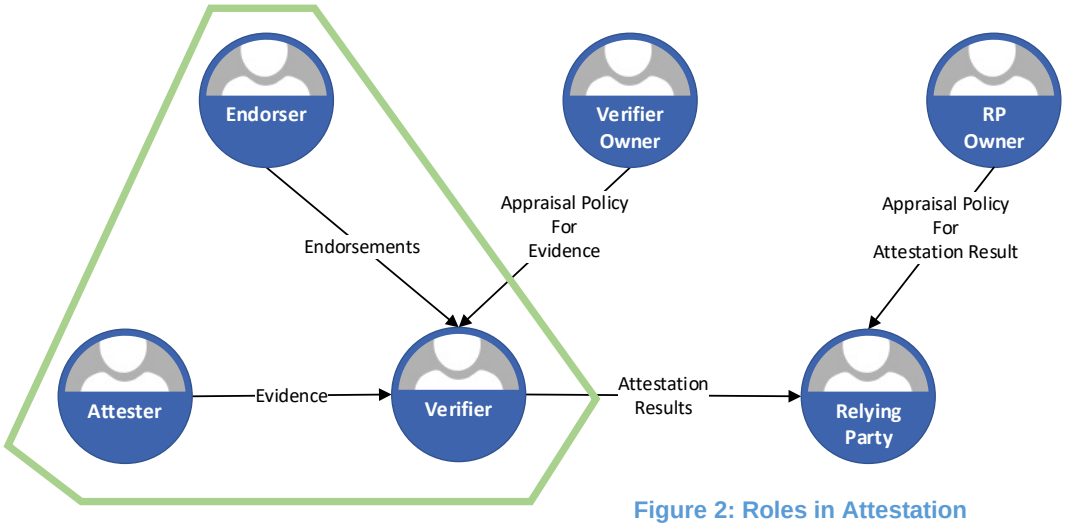

Figure 2: Roles in Attestation

Highlighting indicates roles discussed in this document

Remote Attestation works by establishing a chain of trust through stages from a known Root of Trust to the operational device software. During startup, each stage 'measures' (i.e., computes a cryptographic hash of) the next stage(s) before it's run, storing the results in a Shielded Location, where results from a previous stage can't be changed by a subsequent stage. At a later point, the measurements and associated logs can be sent to a remote

[17] Keying is often done with asymmetric cryptography, but symmetric cryptography may be more attractive in resource-constrained environments.

[18] A Reference Integrity Manifest (RIM) is a file that gives the expected measurements for a given release or version of software. RIMs must be signed by a trusted authority to assure validity. The TCG RIM information model can be found at [13].

Verifier to determine whether the software running in the device is an authentic copy of what the manufacturer provided, and what the device owner wants to run on the device.

Figure 3 shows an abstract example of how chains of trust can be seen as forming a tree[19] rooted in the RTM of a conventional computing environment.

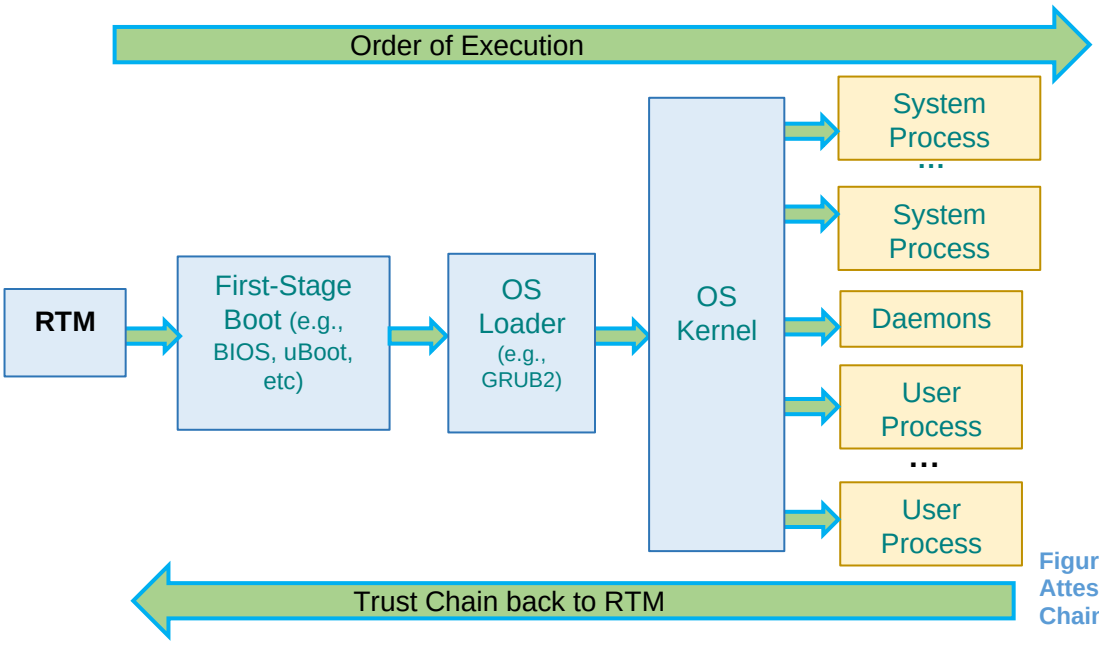

Figure 3: Sample Attestation Trust Chains

Attestation generally relies on several keys to produce a reliable result that's difficult to tamper. For conventional TPM applications, there are two keys for this purpose:

- An Identity Key to assure the verifier that it's made a connection to the intended device.
- An Attestation Key to certify that the results came from a Shielded Location[20] in the intended device[21].

An embodiment of Remote Attestation can be found in IETF *TPM-based Network Device Remote Integrity Verification* (RIV) [31]. Intended for devices that do not have a need to hide their identity (see Privacy, Section 4.2.1), RIV recommendations are based on TCG's specification for provisioning Device Identity keys ([16]). The RIV document provides procedures for manufacturers to ship devices pre-provisioned with DevID and Initial Attestation Keys signed by the manufacturer, reducing the number of steps needed to start using attestation.

4 Cryptographic Device Identity and Provenance

Device Identity and Provenance are interrelated topics that together address a number of questions:

- Which device is which?
- What organization(s) made the device?

[19] Cf. https://etherealmind.com/algorhyme-radia-perlman/, lines 1-4

[20] Shielded Locations are memory cells where secrets such as keys can be safely stored at runtime. Protected Capabilities are mechanisms that allow only authorized access to security-sensitive locations or functions. See the TCG Glossary [1] for normative definitions. Memory locations inside a TPM are generally considered Shielded, as access to these locations is controlled by TPM policies. Other roots of trust use different mechanisms, but designers should be cognizant of the mechanisms used to shield their sensitive secrets.

[21] TPM1.2, TPM2.0, DICE and MARS all have different nuances on how Attestation Keys may be used, but good crypto-hygiene usually requires separate keys for Identity and Attestation regardless of technical capabilities. TCG's specification for provisioning Device Identity keys ([16]) suggests using the same identity *information* in the Identity Certificate and the Attestation Certificate, but privacy-sensitive applications may use a different approach (e.g., a Privacy CA) to mask (while still certifying) the Attesters identity. See additional notes on privacy in Section 4.2.1.

- Have multiple organizations had roles in the configuration of the device?

In many cases, the proof of identity and provenance consists of a digital document, signed by a trustworthy authority, inextricably linked to a cryptographic key inalterably present, and uncopiable, in the device. The cryptographically-protected document should link the specified device-key with device characteristics, and should include identity-related information, such as the manufacturer identity and device serial number, and possibly include much more information about security properties and platform chain of custody. The digital document typically utilizes a standard format, such as for example the X.509v3 digital certificate.

TCG currently references or specifies two kinds of certificates, one principally for device identity and the other for provenance:

- Identity: IEEE 802.1AR DevID Device Identity Certificate [36]
- Provenance: TCG Platform Certificate [7]

Both certificate types provide identity and provenance information, but have different emphasis and use cases, described below.

4.1 Identity with 802.1AR DevID

The IEEE 802.1AR *Standard for Local and Metropolitan Area Networks — Secure Device Identity* [36] (aka "DevID") is a narrowly-focused specification that provides a simple and easy-to-use mechanism for device identity.

Essentially, a DevID provides a cryptographic version of the serial number plate that manufacturers often affix to their products. Encoded as an X.509 certificate, the DevID usually contains:

- The Model Name of the device
- The Serial Number of the device
- The public half of an Identity Key
- … all signed by a recognized authority (usually the device manufacturer).

As an avatar for the device's serial number plate, the DevID is expected to reflect the identity of the device itself, as shipped by the manufacturer, and is expected to be stable throughout the device's lifetime. As suggested by the name, DevID focuses on device identity, with enough provenance information to allow the verifier to identify the manufacturer[22].

The DevID certificate must be bound to a private key stored in a Shielded Location, where it can't be leaked or revealed. The different TCG Roots of Trust can be used to allow the private key to be used for signing to prove possession[23].

In operation, the DevID can be used as a client certificate to establish a mutually-trusted connection between a management application on a trusted server and the specific device, using a protocol like Transport Layer Security (TLS) [36]. As part of the initial TLS handshake, the device is expected to send its DevID certificate, containing identity information, and be prepared to sign a challenge from the server with the private key, proving that it has possession of the key. The management application can validate the signature on the DevID certificate to confirm the manufacturer, validate the signed challenge with the public key in the DevID certificate, and from there, deduce that the device on the other end of the TLS connection really is the one it claims to be.

[22] TPM applications usually require a separate key, parallel to the DevID, called an Attestation Key, to sign the evidence used in an attestation protocol exchange. The DevID and AK can be bound together in several ways, depending on privacy expectations, as outlined in *TPM 2.0 Keys for Device Identity and Attestation* **Error! Reference source not found.**

[23] TPM, MARS and DPE use hardware shielding to prevent the key from being revealed even to the device itself. DICE relies on device software to prevent exfiltration of the key.

For embedded gear like routers or other Industrial IoT devices, where privacy-protected personally-identifiable information is usually not a concern, initial provisioning is considerably simplified if the device is shipped by the manufacturer with the DevID already in place and ready to go. In that case, the device might prove its identity to the management application by establishing a standard TLS connection.

The 802.1AR specification also distinguishes between Initial Device Identifiers (IDevID) and Local Device Identifiers (LDevID). IDevIDs are usually installed by the device manufacturer prior to shipment, and are signed with a key traceable to the manufacturer. IDevID certificates can be used "out of the box", i.e., without any pre-configuration, to facilitate secure zero-touch configuration[24].

An LDevID certificate is optional, and can be installed by the ultimate owner of the device to attach an identity to the device that's signed by the owner and identifies the device in the owner's name space (e.g. akin to an enterprise's Asset Tag).

It should be noted that the IEEE 802.1AR specification, while suggesting the use of a TPM's Shielded Locations to store DevID private keys, does not require a TPM. Designers using other technologies will have to provide their own mechanism for shielding any identity signing keys[25].

4.2 Provenance with TCG Platform Certificate

A TCG Platform Certificate can be used to convey the manufacturer / model / serial number identifying a device. TCG Platform Certificates are defined to be Attribute Certificates[26] (see x.509 [43]) and can't be used directly with protocols like TLS, but platform certificates can express much more about the device's provenance than IDevID certificates.

Platform Certificates offer two significant features beyond simple identity:

- Platform Certificates can convey Security Assertions, statements about a device's security characteristics, as certified by the signatory. In a simple example, a Platform Certificate could assert (for example) that the specific device was equipped with a secure BIOS, or a TPM, or that it has a FIPS certification.
- Using a sequence of Delta Platform Certificates, chain-of-custody can also be documented, where a particular device might have a core component (e.g. a motherboard) that was built and certified by one manufacturer, then passed to a system builder that assembled other components and added security assertions, and then passed to a Value Added Reseller (VAR) that might have configured I/O cards or added software, plus its own security assertions.

Platform certificates can contain several categories of assertions:

- Authenticatable assertions (e.g., the signature),
- Assertions where corroborating evidence can be checked using device resources (e.g., device serial number),
- Assertions relating to the device that are unverifiable (e.g., FIPS certification), where trust in the signer of the Platform Certificate is required.

In addition, an important part of a TCG Platform Certificate includes assertions about the device's Roots of Trust, including:

[24] The Initial and optional Local DevID credentials will often be paired with Initial and optional Local Attestation Keys; see [16].

[25] Users of platforms that *should* be using a TPM to store keys can verify that they *are* in a TPM with additional steps, using, e.g. TPM2_Certify() (see TPM 2 specification [3]).

[26] There's a difference between the DevID Public Key Certificate (PKC) and the Platform Certificate's original Attribute Certificate (AC) format. PKCs are used with protocols like TLS, while ACs are signed documents that convey security assertions. See x.509 [43].

- Assertions indicating the type(s) of Root(s) of Trust, and certain security-sensitive capabilities (e.g., authentication of a physically-present user for configuration, or hardware-based memory protection for security-sensitive parts of the firmware, or the presence of tamper-protection)
- A URI to expected firmware Reference Integrity Measurements, to enable attestation
- Assertion of compliance (if any) to firmware update specifications such as SP 800-147 [40]

Platform Certificates also can be used to describe the expected configuration of a platform, including not only the base unit, but characteristics and identity of component devices, such as plug-in cards or other field-replaceable units, allowing the user to determine whether the platform as it stands still matches the platform as-shipped by the vendor(s).

Details of exactly what can be specified in a Platform Certificate are spelled out in *TCG PC Client Platform Firmware Integrity Measurement Specification* [12] and the *TCG Platform Certificate Profile* [8].

The Platform Certificate mechanism can also track authorized changes to the platform through the supply chain with the addition of cryptographically-linked Delta Certificates[27]. System Integrators and VARs can provide delta certificates to show modifications that were made during their processing, each one signed by the organization that made the change, and linked to the base platform certificate or the most recent in a series of changes.

Although an Attribute Certificate is not suitable for use with standard protocols such as TLS to prove device identity, a Platform Certificate is bound to the physical device through a TPM Endorsement Key (EK), unique to each TPM, contained in the certificate[28].

4.2.1 Keeping Identity Private

Privacy is always a fundamental concern in any security architecture, but there are important nuances.

All kinds of gear should have provisions to prevent the disclosure of information that the owner would consider private. But, beyond that, network-connected devices often fall into one of two categories:

- Equipment such as personal computers or phones typically must have strong protection to prevent the identity of the machine and the identity of its owner from being linked without the owner's permission. In this situation, it's usually necessary to avoid the disclosure of strong device identity without explicit consent from the owner. As a result, an "on-by-default" Device ID may not be a good option. To protect privacy, TCG Platform Certificates may be configured to require explicit steps to prove links to a specific device.
- Other equipment, such as routers, switches, firewalls or many kinds of Industrial IoT must be able to prove their identity routinely as part of normal operation. For example, an industrial valve controller does not have any inherent 'right to privacy', and must be able to prove its identity before it should be trusted with critical operation. In this case, a DevID, installed in the factory by the manufacturer, may be appropriate.

In considering DevID and Platform Certificate, keep in mind that:

- IDevID is inherently *not* privacy-preserving – its purpose is to provide a clear binding between a verifiable key and an externally-recognizable identifier.
- A Platform Certificate can be used in privacy-preserving applications, as it can be configured to include only manufacturing and generic model information if desired.

4.3 Comparison

While the two technologies both provide information on identity and provenance, they have different strengths:

- Platform Certificates are more valuable when:

[27] Delta Certificates are just one mechanism for this; SBOMs, Reference Integrity Manifests, etc. also can be used.

[28] Binding the Platform Certificate to the platform with DevID or EK yields different privacy characteristics; DevID is intended to identify the device as easily as possible; authenticating an EK is a more-indirect process, yielding more opportunity to control disclosure.

  - There's a complicated supply chain to be tracked, e.g., system board vendor, chassis vendor, VARs, option cards, etc.
  - Assertions about the device's security characteristics, beyond simply identity, are desirable.
  - Those assertions might be modified as the device passes through a supply chain.
  - Privacy is important, making it ill-advised to ship products with trackable identity available by default. (Be sure to use the Attribute Certificate format if privacy protection is important – see Section 4.2.1)
  - A comprehensive verification procedure is desirable[29].
  - Validation of the less-common Attribute Certificate format is acceptable[30].
    - Alternatively, the Public Key Certificate format can be used, but using this certificate to prove identity, in addition to provenance, may reveal more about the device than the owner wishes.
- DevIDs are more valuable when:
  - An immutable permanent identifier for the device is desired.
  - Out-of-the-Box, zero-touch operation with existing IETF protocols is important (e.g., TLS authentication or RFC-8572 Secure ZTP).
  - Privacy concerns don't rule out a default assumption that platform identity should be verifiable.
  - Identification of the initial OEM in the DevID yields enough provenance information, without requiring details regarding other aspects of the product's supply chain.

It should be noted that Device ID and Platform Certificate are not an "either-or" choice. Specifications neither encourage nor prohibit the use of both, and, depending on the application, there could be justification for both on the same device.

5 TCG Roots of Trust for Identity and Attestation – TPM, MARS, DICE

A critical part of any mechanism for cryptographic identity and attestation is a hardware Root of Trust, i.e., some hardware-protected identity mechanism that can provide a binding between the device and evidence about the state of the device, and a mechanism to protect keys that can be used for cryptographic proof of possession[31],[32]. With these mechanisms, cryptographically-signed evidence can be generated, showing which software was present when the device was initialized.

In technical terms, attestation requires three Roots of Trust:

- A Root of Trust for Measurement (RTM) comprises trusted code (or equivalent) that measures the first mutable code the device will execute.
- A Root of Trust for Storage (RTS) comprises Shielded Locations to store measurements for later use in an attestation protocol exchange, as well as secrets used by the device.
- A Root of Trust for Reporting (RTR) comprises a mechanism that is used to authenticate measurements sent in the attestation protocol exchange. Typically, this would be done by using a protected key to sign a bundle of measurements.

The Trusted Computing Group has defined three hardware Root of Trust technologies for various applications and deployment infrastructures (e.g. stand-alone user devices, on-prem enterprise computers, cloud-based device

[29] The Platform Certificate may contain quite a few different assertions regarding the platform's security characteristics, but that's no help unless the Verifier is able to assess them all.

[30] Crypto libraries commonly validate Public Key Certificates, but Attribute Certificate support is less common.

[31] With asymmetric key cryptography such as RSA, proof of possession of the secret key is accomplished in a couple of steps: The challenger (the one that wants proof of possession) sends a 'nonce' (a large random number known only to the challenger). The device being challenged signs the nonce with its private key, and returns the result. The challenger can then verify the returned value using the device's expected public key. If the signed nonce matches the value the challenger sent, the device must have possession of the secret.

[32] The TCG Glossary [1] gives detailed definitions for various types of roots of trust.

farms, Industrial IoT devices, etc.), any one of which can be used as components in a system to provide platform identity and attestation. Section 5.6 below shows a comparison in terms of the required Roots of Trust.

- The Trusted Platform Module (TPM) is a self-contained security processor function that contains Shielded Locations to store keys and a rich set of Protected Capabilities to perform operations such as signing, encryption, decryption and deletion of secrets. The TPM is ideal for system vendors who use processors that include embedded TPMs (see 5.1.1)[33], or for those that must meet regulatory requirements requiring TPMs, or simply for applications that need the more-complex security features a TPM can provide.
- Measurement and Attestation RootS (MARS)[27] is a mechanism that can be built directly into processor silicon, and can be used to make a Root of Trust for identity and attestation. MARS implements a small set of TPM-like capabilities, and is intended for constrained systems where a TPM is not feasible. The MARS approach uses additional silicon to implement the Shielded Locations and Protected Capabilities, thereby isolating secrets and cryptographic functions from the targeted host. The MARS Library specification [27] allows implementations relying on symmetric or asymmetric keying.
- A Device Identifier Composition Engine (DICE) ([20], [21], [22]) is specified as a critical component of a *DICE-Enabled RTM* that also incorporates device identity, and can be built directly into processor silicon. DICE requires very little additional silicon, relying on simple hardware to provide identity and protect RTM secrets. DICE operates by bootstrapping layers of firmware, each providing its own attestation facility and its own symmetric or asymmetric cryptography attestation identity, instead of sharing a single RTS and RTR implemented as Shielded Locations and Protected Capabilities. The DICE methodology can optionally be augmented with a DICE Protection Environment [25], to further safeguard attestation information with hardware-Shielded Locations and Protected Capabilities, although at an obvious increase in firmware cost.

It should be noted that TCG specifies elements that can be used to improve system security, but it does not specify systems. As such, the designer is always left with the task of integrating the TCG components into a working system. The three methods take different approaches to equipping the designer for this integration task:

- TPM is a “module” (often literally so, with a physical implementation as a chip). The TPM itself comes with a carefully specified API, a set of standard functions, a software library yielding higher-level APIs, and numerous documents describing accepted practices for using the capabilities, all of which work well in environments like Windows and Linux.
- DICE offers a lean tool kit of simple techniques to make a Root of Trust for Measurement, to fit an environment where other, non-TCG, components may be used to provide Protected Capabilities and Shielded Locations. Designers must take a more active role in choosing the components and functions they need to meet their particular risk profiles, rather than adapting the application to the fixed feature set implied by use of the TPM.
- MARS could be considered a mid-point between DICE and TPM, providing more structured design guidance than DICE, but a lighter silicon footprint and less prescriptive environment than TPM.

5.1 TPM

The Trusted Platform Module has been developed and refined at TCG for over 20 years[34], and has been widely deployed in many applications. The first TPM2.0 specification was first published in 2014, and continues to be updated and extended with new capabilities. As of November 2023, the current version ([3]) is 1.59, published in 2019.

[33] A discrete TPM is also valuable for system designers who build devices using a variety of commercial off-the-shelf processors and want a uniform TPM environment.

[34] TCG was founded as the Trusted Computing Platform Alliance (TCPA) in 1999 to develop the TPM.

The TPM is commonly implemented as a low-cost, low-power, stand-alone security processor chip[35], and, in that form factor, is restricted to systems that can afford an extra chip.

A TPM chip typically contains a number of elements providing a rich variety of services[36]:

- Cryptographic elements for signing, encrypting and decrypting, using a variety of symmetric and asymmetric algorithms.
- Protected capabilities and shielded storage to hold private keys and other sensitive data, with a variety of policies to allow or restrict their use and/or export, possibly depending on the state of the system.
- Platform Configuration Registers (PCRs) – a special form of Shielded Locations to keep records reflecting the state of the system at each stage in system initialization, the critical requirement for attestation.
- Protected Non-volatile storage for security-sensitive information such as device identity keys, or application-specific keys such as disk-encryption keys.[37]
- Other ancillary security-related functions such as Monotonic Counters.
- A TPM also requires a cryptographic-quality random number generator. The RNG is used internally for key generation, but is also available externally for other cryptographic uses in a device.

Platform Configuration Registers (PCRs) are the critical element in TPM attestation. As a device boots, each stage of the boot computes the measurement[38] of the next stage and extends[39] the measurement into a PCR (see Figure 4). Different PCRs can be used for different environments encountered during boot, to allow layers to be compartmentalized, so a change in one layer doesn't change PCR values in other layers. Aside from reducing confusing interactions between independent asynchronous elements in the boot chain that may take measurements in an unpredictable order[40], allocating PCRs to stages of the overall boot environment allows practical 'sealing' based on TPM policies that allow the TPM to release sensitive data only if a particular layer is in a known-good state, as indicated by its PCR value.[41]

As the platform boots, it stores measurements in the event log (see Canonical Event Log [19]) and protects the integrity of the log by storing a digest of the measurement in the PCR. The event log contains the measurements themselves, and an indication of which PCR each one went to, and may contain hints as to what the measurements refer to (e.g., "GRUB version 2.06"). The event log is not encrypted, but the hashes in the event log will be cross-checked against the PCR values by a Verifier as part of later verification.

Once the device is operational, authenticated copies of the PCR values can be retrieved with TPM2_PCR_Read() and TPM2_Quote() commands. In response to a Quote, the TPM packages up the requested PCRs and provides a signature using an Attestation Key (AK), a key restricted so it won't sign forged attestation messages that originate outside the TPM. The Verifier can validate the AK to make sure the quote came from the same device that provided proof of identity and that the evidence has not been altered. The Verifier can then examine the log to make sure the hashes of all the relevant measurements line up with values retrieved from the PCRs, and then can cross-check

[35] TPMs traditionally were separate chips, but are not always! TPMs may also be implemented as part of a larger processor using a "special execution mode" to keep TPM state isolated from other tasks on the same processor. See Section 5.1.1

[36] See TPM2 Part 1 Architecture, Section 11.1 [3].

[37] In TPM2.0, NVRAM can also provide the same Protected Capabilities as PCRs and monotonic counters.

[38] That means that the current stage computes the hash of everything that's considered security-sensitive, i.e., code, credentials and configuration, contained in the next stage, before launching that next stage.

[39] Each PCR is large enough to contain a cryptographic hash (e.g., 256 bits for SHA-256). For SHA-256, the TPM Extend operation takes a new 256-bit value, hashes it with the current PCR contents, and puts the resulting value back into the PCR to form an updated value. The net result is that the PCR contains a hash of all the hashes 'extended' into that PCR.

[40] Note that PCR measurements are not commutative. Measuring A, followed by a measurement of B results in a different result than measuring B then A, so the measurement log must typically be consulted to untangle any asynchronous measurements. Cryptographers typically don't care about reducing confusion, but engineers often do.

[41] See Section 5.3 of TCG TPM2 Provisioning Guidance [17]

with Reference Integrity Measurements to ensure that the measurements line up with what the administrator expects to see on the device.

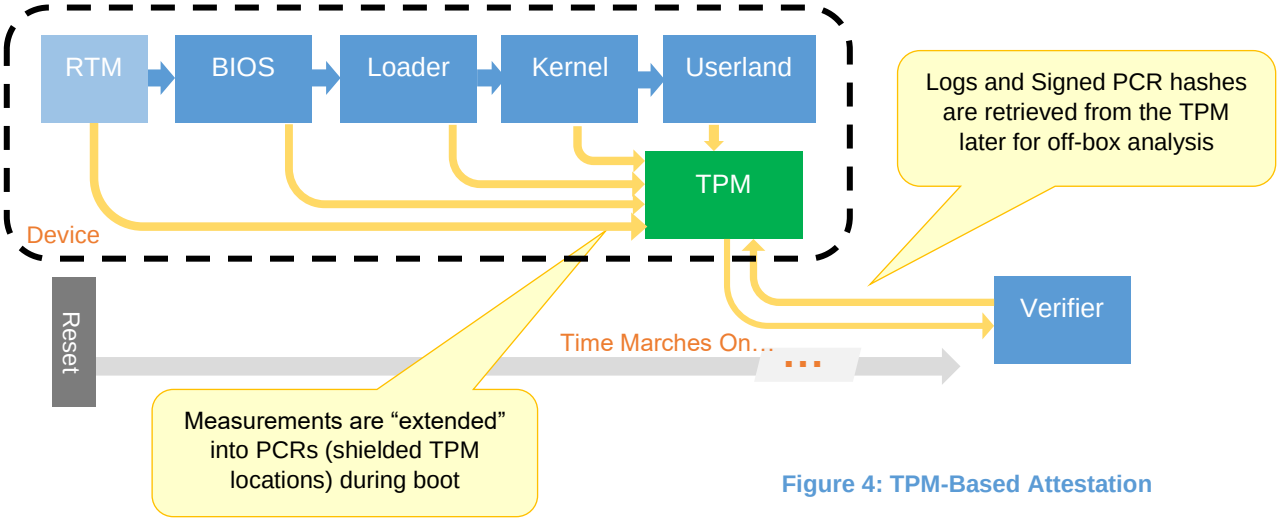

Figure 4: TPM-Based Attestation

5.1.1 TPM Implementation

The bulk of TPM2 documentation describes the API, i.e., commands and responses exchanged between the host and its TPM, although other TCG TPM2 documents also specify physical characteristics down to pinout and bus type (e.g., see [14]) to encourage multi-vendor interoperability.

In addition to discrete chips, a number of processor vendors now offer integrated TPMs, where internal SoC or processor resources are used to implement the API, providing TPM functionality without an additional discrete TPM chip.

Figure 5 illustrates some of the common approaches to TPM implementation.

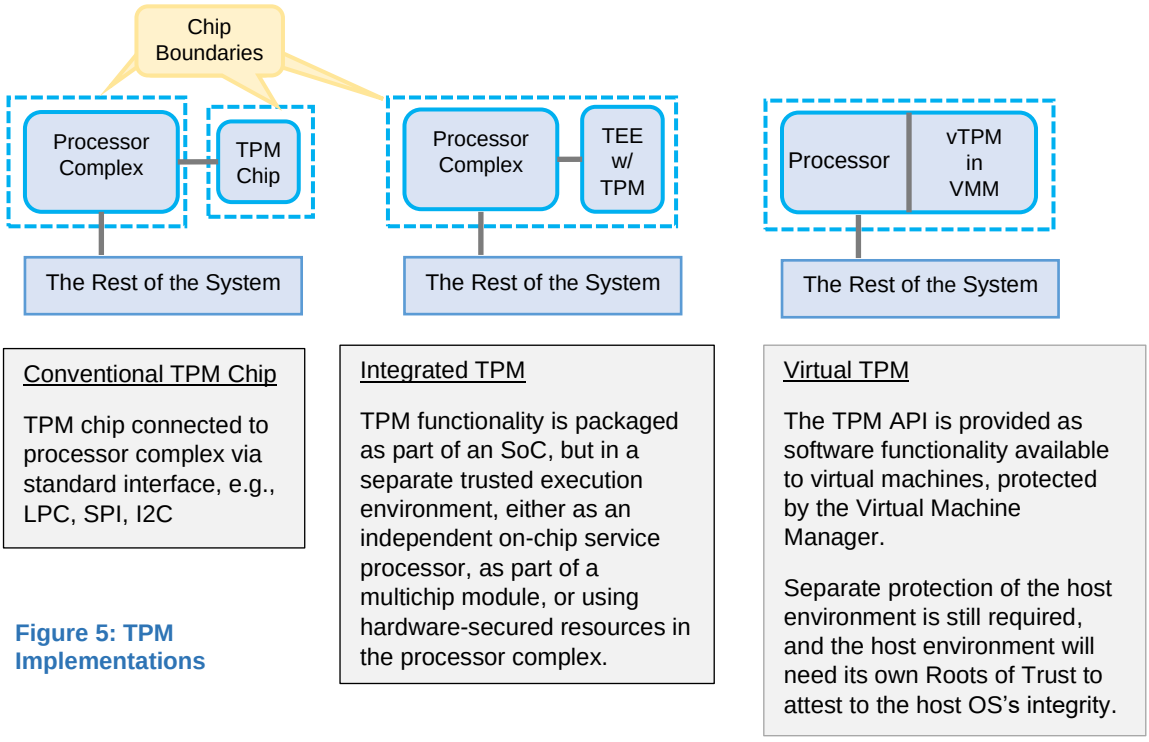

Figure 5: TPM Implementations

The various Integrated TPM implementations can provide a fully compliant API, but may depend on hardware modes in the host CPU complex to achieve adequate separation of resources,[42] to ensure that secrets don't leak from shielded TPM storage into general processor state. In choosing between a discrete TPM and an integrated TPM, users must consider:

- While the base TPM feature set is carefully specified for interoperability, different SoC vendors may offer different revision levels, compliance assertions (e.g., FIPS, CC) or extensions for their integrated TPMs. Designers that use multiple processor families may want to minimize variation by using a discrete TPM.
- An integrated TPM may reduce the risk of a bus-snooping (aka Interposer) attack (see [10], [11]).
- Discrete TPMs may also be capable of providing additional silicon process steps that stymie the physical reverse-engineering and probing techniques that might reveal secrets.[43]
- As in any security-sensitive implementation, system designers are dependent on the skill of the vendor to implement the TPM API without vulnerabilities.

A list of TPMs that have been certified under the TCG TPM Certification program can be found at: https://trustedcomputinggroup.org/membership/certification/

There are also implementations of Virtual TPMs (vTPMs), i.e., software that implements the TPM API, designed to provide TPM functionality to virtual machines. While a vTPM must implement the API, there is no assumption of a hardware Root of Trust for a vTPM. The vTPM typically relies on lower-layer mechanisms that influence security guarantees, and are typically part of the host OS. TCG has published an architectural specification for use of virtual TPMs in [18].

[42] For example, an integrated TPM could be implemented as part of a Trusted Execution Environment (TEE).

[43] See https://www.cl.cam.ac.uk/~sps32/ARO_2011.pdf for a generic analysis of physical attack and defense of silicon devices.

Finally, it should be noted that there are pure software implementations of the TPM API. Although these implementations are excellent for debugging problems (since the internal state can easily be examined), they aren't recommended for production security work (since the internal state, secret keys and all, can easily be examined).

5.2 MARS

As enhanced security procedures become more important on smaller and smaller devices, such as IoT controllers, the cost, space and power of a TPM has become more of a burden. That, coupled with a desire to leverage the existing ecosystem for managing security functions with TPMs, has led to the specification of a minimalist TPM-like, hardware-based intellectual property (IP) block[44] that can be integrated directly into a larger silicon development such as a single-chip controller for IoT or peripheral devices.

Measurement and Attestation RootS (MARS) provides a specification for that IP block.

MARS does not replicate the entire TPM API, but it offers an Identity keying structure analogous to the TPM, and Attestation capabilities patterned on the TPM approach, trimmed to more easily fit the silicon flow used in conventional processors.[45] The MARS API is described in [26].

MARS also offers the designer an option to reduce silicon area by omitting support for hardware-intensive asymmetric cryptography. Factors in making this choice, such as key management, silicon area and quantum resistance, are described in Section 5.6.7.

MARS depends on a device-specific secret called a Primary Seed (e.g., a 256-bit cryptographically unique random number[46]), set during provisioning[47], as the root for its identity. Keys and other secrets, symmetric or asymmetric, are all derived from this Primary Seed using a Key Derivation Function (KDF)[48], as can be done in the TPM. MARS does not specify non-volatile memory, so unlike the TPM, MARS must generate the keys it needs "on the fly", deriving them from the persistent Primary Seed when they're needed. Designers may want to cache keys to reduce execution latency, but that's out of scope for the MARS specification.

Like the TPM, MARS Attestation is based on a series of measurements stored by each stage into one or more Platform Configuration Registers. Like the TPM's PCRs, the PCR is never written directly; each measurement is *extended*[49] into the PCR. As with a TPM, in anything but very simple use cases, firmware making measurements should update a running event log as well, to be checked by the verifier against the PCR(s) during a later verification step (see Figure 11).

Like the TPM, MARS also reports the value of its PCR(s) by a 'quote', i.e., a data structure generated by the MARS block and signed by an attestation key known only to the MARS block.

Unlike the TPM, and in line with anticipated IoT SoC applications, MARS also includes a mechanism called Trusted Sensor Registers, which can be linked to various hardware sensors, and quoted along with PCRs. This enables sensor values to be signed when reported, with minimized risk of software tampering.[50]

---

[44] "IP block" is a term-of-art referring to a reusable unit of integrated circuit logic. The term does not imply any specific ownership or licensing requirement. It's just a bunch of gates configured to provide a hardware function in an ASIC or FPGA.

[45] In particular, MARS does not require a silicon process that can implement reprogrammable non-volatile memory.

[46] "Cryptographically Unique" means that the number is randomly chosen from such a huge space that the odds that an adversary can guess your secret is one in a zillion-zillion, i.e., about zero. For example, there are about 10^{77} 256-bit AES key values. As a rough estimate, our Milky Way galaxy contains about 10^{67} atoms. So guessing my AES key is about 10^{10} times harder than guessing which atom I'm thinking of in the entire galaxy.

[47] For example, the silicon manufacturer might program a unique seed into a chip containing MARS by blowing one-time fuses prior to shipment, but the method to do so is out of scope for the MARS specification.

[48] For some background on Key Derivation Functions, see https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-108r1.pdf

[49] As with the TPM, "extending" means hashing a new measurement with the previous PCR value and updating the PCR with the result. No change.

[50] MARS of course cannot ensure that the sensor is correct, it can only show that the value the sensor reported hasn't been tampered with since it was deposited in the TSR.

5.3 DICE

While TPM and MARS are focused on keeping tamper-resistant keys and attestation records in hardware, DICE takes a different, software-focused, approach, requiring minimal TCG-specified hardware, allowing use in hardware-constrained environments. DICE integrates device identity with software/firmware attestation, and provides combined evidence of identity and attestation, unique to each device.

As with most measured or secure-boot technologies, DICE is based on the idea of layering [21]: during the startup of a system, there are a series of discrete steps, or layers, each of which is responsible for authenticating and starting up the next one. In concept, these layers might be a boot block, followed by a boot loader (e.g., uboot), followed by an OS loader (e.g., GRUB2) followed by a kernel, and so forth. Each layer is responsible for taking a measurement of the next layer, before launching it, and adding to a trail of verifiable records (containing no secrets), before it starts the next layer. Attestation can be achieved by following the chain of records left by each layer.

DICE starts with a "Unique Device Secret" – a large (e.g., 256-bit random) number unique to each instance of the Root of Trust[51]. The UDS is accessible to the DICE within the RTM, but is equipped with a one-way latching hardware mechanism that renders the UDS invisible when the DICE has used the secret to compute a composite of the UDS and the first measurement of the layer after the RTM. In that way, the UDS is usable briefly at the start of the boot process, but can never be known once the machine is in operation.

Each DICE layer is viewed as creating a TCB, Trusted Computing Base,[52] defined in the DICE Layering Architecture [21] as follows:

> *A device's TCB consists of all security relevant components that have been loaded at a given point in the boot sequence. A TCB component is comprised of hardware, firmware, software and/or configuration. A measurement of a TCB component is a TCB Component Identifier (TCI). An example of a TCI value is a digest of component firmware. In some cases where a component does not consist of measurable firmware or software, a hardware product identifier (or equivalent) may be used.*

The initial Engine (the E in DICE) is used to implement the measurement function of the Root of Trust for Measurement (see Section 5.4) and has a few tasks:

- It must measure the next layer to form a TCB Component Identifier (TCI)
- It will generate a new secret (called a Compound Device Identifier) by hashing the UDS with the next layer TCI. The CDI is passed to the next stage, and contains a digest of the UDS, but due to the hashing, does not reveal the UDS.
- The DICE layer code generates an X.509 certificate that acts as an embedded CA[53], containing extensions with the Evidence of the next layer, and other non-secret information relevant to attestation (see Figure 6). Each layer's embedded CA certificate path must link to the previous layer, ending at the unique manufacturer-supplied Embedded CA, which is part of the RTM. The manufacturer-supplied Embedded CA in turn must be linked to a manufacturer's public root CA.

[51] Note that a single device or platform could have multiple components, each with its own root of trust, e.g., main processor, network controller, graphics accelerator, etc.

[52] The conventional "Orange Book" view of TCB is that there's One TCB; the boot process is responsible for making sure it's secure, and further OS security mechanisms are based on that Trusted Computing Base. In fact many systems (TPM, DICE and MARS) stretch that definition; DICE explicitly treats each layer as a TCB, which may be used to derive a new TCB present in the next layer.

[53] Note that the system designer may have to carefully consider the value of the x.509 pathLenConstraint parameter in the Manufacturer's Embedded CA to ensure that unauthorized extensions of the attestation chain can't be created.

- Finally, the Root of Trust must hide the UDS, so it cannot be revealed. Because this function[54] must be trusted to take the first measurement, and to hide the UDS, it must be carefully protected against attackers (see Section 5.4).

Subsequent layers each repeat more or less the same process:

- In place of the UDS, subsequent layers start with the CDI from the previous layer as the incoming secret.
- The same calculations to measure the next layer and generate an attestation certificate are followed, linking the new CA into the CA certificate generated by the previous layer.
- Rather than activating hardware protection of the UDS, subsequent layers simply "forget", i.e., erase the incoming CDI and any derived secret keys, while passing the new CDI, formed by hashing the incoming CDI with the hash of the next layer's code, on to the next layer.[55]

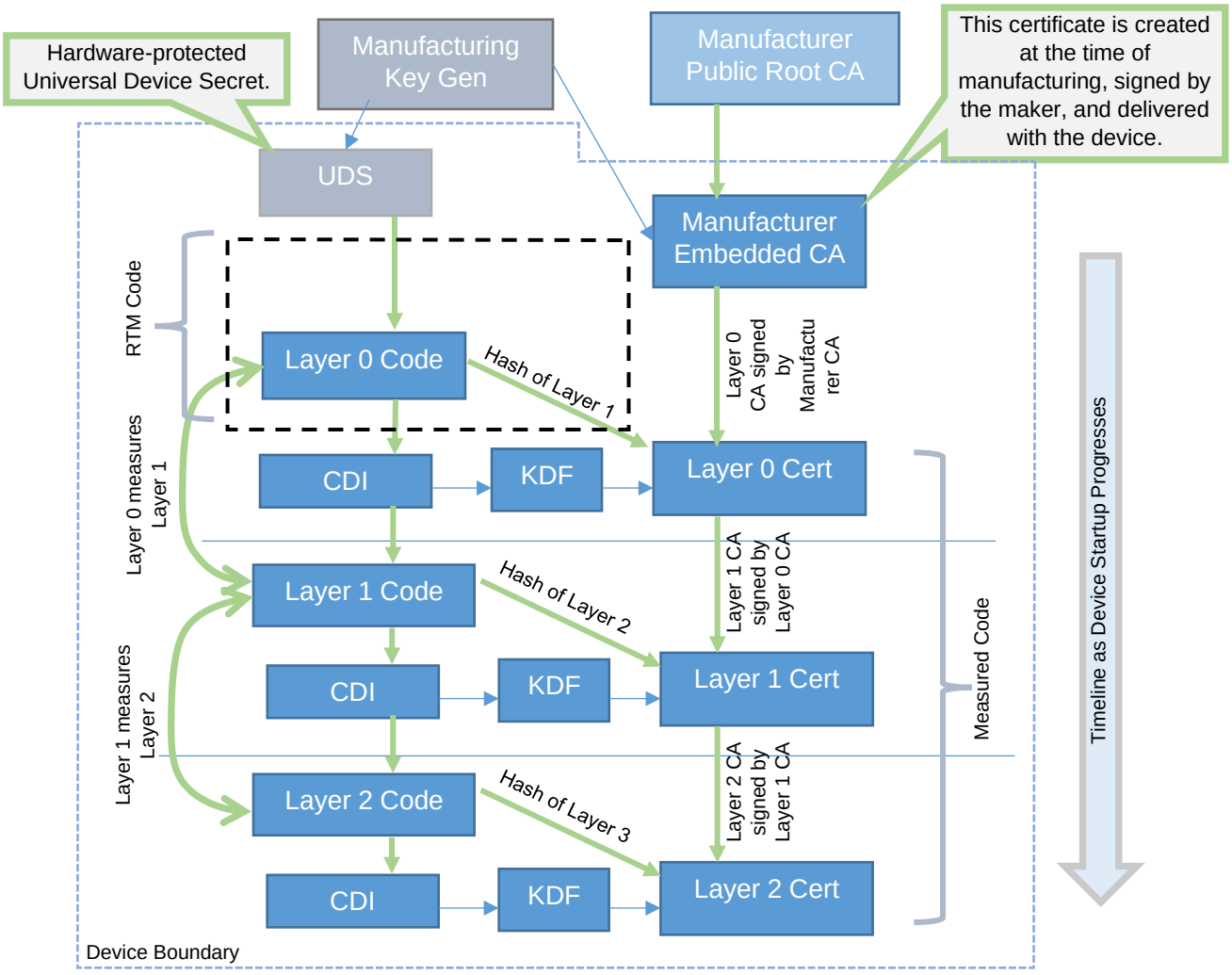

Figure 6: DICE Layering Model

[54] The DICE root of trust is usually assumed to be firmware code in immutable ROM, but that's up to the implementer. As long as it's 'protected', we're good.

[55] As long as each layer correctly "forgets" its incoming CDI after generating the next one, then subsequent layers cannot exfiltrate secrets from previous layers. DPE (Section 5.3.4) provides Shielded Locations to decrease the chances of a mistake in this handoff.

The net result is that, at the end, there's a chain of certificates, each linked to the previous layer, identifying the unique device and all the software modules that make up its TCB in one chain.

Although DICE protects the layers of secrets involved in attestation, it should be noted that, unlike the TPM, shielded storage in the general sense is out of scope for DICE specifications (DPE excepted: see Section 5.3.4). Applications that need keys that should be protected by hardware or a Trusted Execution Environment may need additional help, in the form of a TPM, Dice Protection Environment (Section 5.3.4) or other vendor-specific mechanism to provide Protected Capabilities and Shielded Locations.

DICE also does not require non-volatile (e.g., flash) storage, and as a result, specifies no mechanism to store persistent information (beyond the Unique Device Secret).[56]

5.3.1 Verifying the DICE Attestation chain

DICE uses a model for attestation that's substantially different from MARS and TPM, based on a chain of x.509 certificates created and signed during the startup of a DICE device.

- DICE inherently combines identity with attestation[57]. The measurement of the first mutable code is combined with the UDS to create the first CDI, after which the UDS is locked away by a hardware latch and the CDI passed to the next layer, to continue the attestation calculation. As a result, two otherwise-identical devices with identical software loads will produce different values of attestation evidence, despite using identical methods[58].
- Rather than storing the attestation results in a hardware register (one or more PCRs), DICE produces a chain of x.509 certificates, each recording the attestation evidence for a particular layer, signed by the previous layer. This chain can be transferred to a Verifier for analysis, confirming both the identity of the device and its software load.

The DICE trust chain inherently embodies both device identity and evidence of software integrity, so no two devices will produce the same leaf certificate. But, when shipped, the device can be equipped with a device identity Embedded Certificate Authority that forms a local root, linked to a public root maintained by the manufacturer, as shown in Figure 6. Using the manufacturer-supplied Embedded CA, the verifier can work backwards from the end of the certificate chain, all the way back to the manufacturer's public Root Certificate to verify that the device was manufactured by the expected supplier, and that it started up with known software at each layer (Figure 7).

[56] Of course, designers are free to include any kind of non-volatile storage that their application may need, it's just not specified by DICE (or MARS).

[57] Of course, both Identity and Attestation must always be authenticated, or else it's impossible to tell whether the attestation results apply to any specific device. But the different approaches may make it easier or harder to determine whether incorrect results are caused by a problem with identity or attestation.

[58] In simple TPM use-cases, it may be possible to determine whether a measurement is "good" or not simply by validating the attestation key then looking for a known value in the relevant PCR; in a DICE implementation, identity is mixed in with measurements, so there won't be one simple "final hash value" that applies to all devices, so analysis of the full evidence chain will be needed.

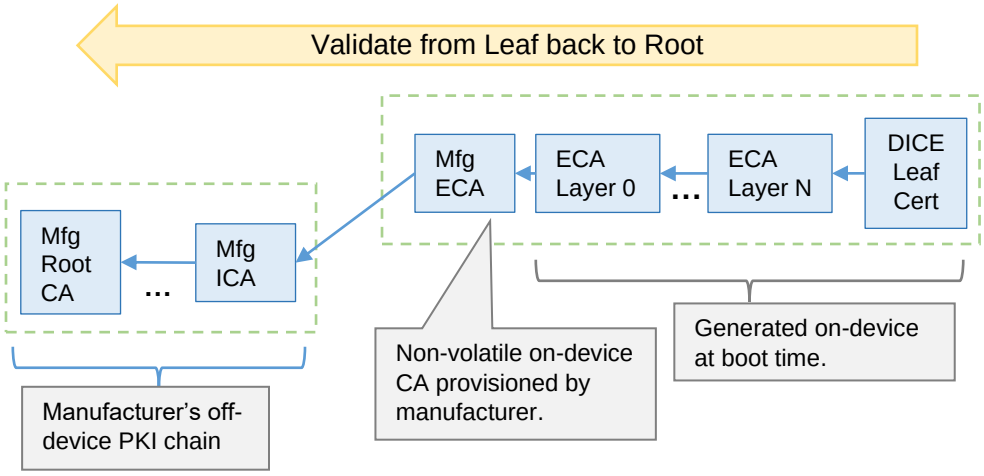

Figure 7: Validating a DICE Attestation Certificate Chain

5.3.2 Proving Identity and Freshness

The DICE certificate chain proves that a device was made by the designated manufacturer, and that it was running authentic software or firmware with the expected keys or other state at the time of measurement. But the Relying Party needs two more critical bits of information for any form of attestation:

- The Relying Party needs to know that the evidence was actually generated on the machine being attested (i.e., proof that the evidence was not generated on a different device and transplanted.) This requires proof that the leaf certificate has access to its private key, and that the chain of attestation certificates is valid.
- The Relying Party also needs to know that the evidence is current, i.e., it reflects the state of the machine during the current boot cycle, not the state from some boot cycle in the past (i.e. "replay" attacks should be detected).

With DICE, verification of identity can only happen when attestation is complete, and the verifier can validate the chain of certificates from the leaf signer, through the manufacturer-supplied Embedded CA Device certificate, to the manufacturer's public root.[59]

Validation of the attestation chain provides proof of the claimed identity, but there are two common techniques to prove that a result is current (not simply replayed from a previous boot cycle):

- The Verifier may supply a nonce[60] and request the device under attestation to sign the nonce with a key known only to the Attester. For DICE, this key would have to be held by the outermost layer of the TCB, and of course that TCB layer must be available for execution at the time of attestation.
- The Attester may include a certified time stamp in the attestation message. Certifying a time source can be a bit complicated, but see IETF's TUDA draft [35] for one technique.

5.3.3 DICE with Symmetric Keys

DICE can be used with symmetric cryptography, an option that's attractive for resource-constrained devices.

[59] Of course, Identity is critical to all forms of attestation. The TPM and MARS provide identity for attestation by signing attestation results with a key known only to the TPM or MARS, making it much more challenging for compromised device software to misrepresent its identity.

[60] "Number used Once", i.e., an unguessable, random, big number that the verifier makes up on the fly and sends to the attester.

Attestation procedures follow the same layering model, but with symmetric calculations at each layer. The resulting evidence is a hash from the last DICE layer, which can be validated by a verifier with access to the device's Unique Device Secret.

The procedure is outlined in TCG document *Symmetric Identity Based Device Attestation* [24].

5.3.4 DICE and DPE

For small devices such as single-chip embedded controllers, the strict DICE requirement that secrets must be either destroyed before the layer exits, or protected by some mechanism out of scope (e.g., a dedicated key store) does not pose much problem; but for larger systems this gets quite challenging. In recognition of this, TCG has developed the DICE Protection Environment (DPE) [25], a block that provides an interface to Shielded Locations and Protected Capabilities, compatible with DICE layering (see [21]). The DPE has been specified so it can be implemented in a variety of ways, including as firmware in a trusted execution environment or as hardware in an on-chip block of intellectual property to be included in some larger silicon device, along with the DICE UDS secret and its protection mechanism.[61]

The DPE does not require non-volatile memory. It's used to safely pass secrets from one layer to another, but is cleared on a power cycle or reset.

The DPE provides a small set of commands to manage its protected resources. In summary, the API covers these functions:

- Store and manage DICE secrets (CDIs) through layer transitions
- Control access to keys with configurable policies
- Generate and sign Embedded CA and Alias certificates during the layer transition
- Manage and prove possession of keys derived from a layer secret
- Generate leaf certificates and key pairs for attestation and identity
- Sign external objects using a key derived from a layer secret
- "Sealing" of sensitive data[62]

The addition of DPE to DICE is illustrated in Figure 8, showing that the layering architecture stays the same. The difference with DPE is that layers no longer have to manage the handoff of CDIs from one layer to another; instead, the DPE manages the secrets passed from one layer to the next.

[61] DPE may also be implemented in firmware in a Trusted Execution Environment, if such facilities are available.

[62] Sealing provides a way to ensure that a given secret will only be released by the hardware if the machine is in a certain known state (for example, allowing use of a disk decryption key only if the BIOS meets specified security criteria). Sealing is also a TPM capability.

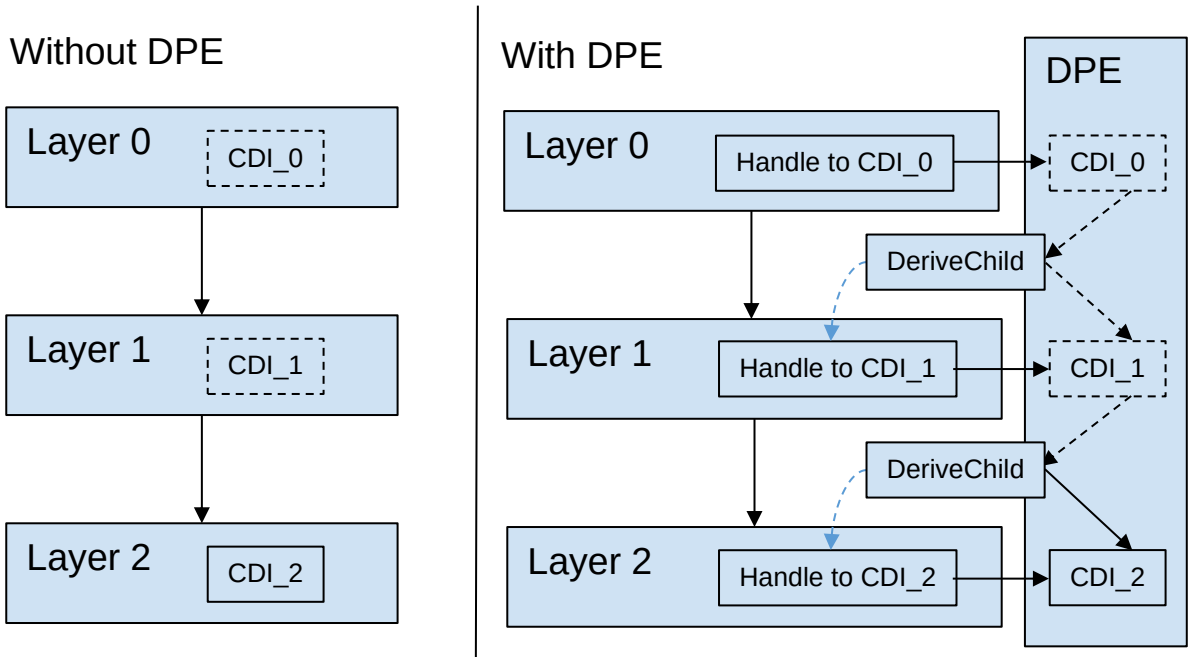

Figure 8: DICE with DPE

As a result, the DPE does not change the DICE architecture; it simply provides a safer mechanism to pass secrets (such as the CDI) from one layer to the next, with reduced chance of leakage.

5.4 The RTM Requirement

The Root of Trust for Measurement function is crucial to attestation, but the complete RTM function is not specified by the TCG, and must be assembled by the system designer, perhaps using TCG-specified components, along with other components. This section outlines expectations for the RTM.

5.4.1 Just How Many Roots of Trust Are There?

The TCG Glossary defines a number of Roots of Trust. In each case, a particular Root of Trust is the function the system designer points to as the function which must be implicitly trusted before any of its derivatives can be trusted. For example, the Root of Trust for Measurement is the function that takes the first attestation measurement; all subsequent measurements rely on that first measurement to prove their trustworthiness (more on this below).

Although definitions of Roots of Trust are out of scope for this document (see TCG Glossary [1]), here are several that are involved in attestation:

- A Root of Trust for Measurement (RTM) measures the First Measured Code (see below).
- A Root of Trust for Storage (RTS) provides Shielded Locations and Protected Capabilities to store security-sensitive attestation data.
- A Root of Trust for Reporting (RTR) certifies that the reported measurements of attestation evidence are authentic.
- A Root of Trust for Identity (RTI) proves that the device is the one it claims to be.

5.4.2 What is an RTM?

All TCG attestation technologies depend on producing a chain of measurements, where each software stage must be trusted to take a measurement of the next stage to run, and record that measurement, before launching the next stage. The resulting series of measurements can be assembled by a Verifier into a "chain of trust", where each stage has been measured before it starts, so that, during attestation, each stage can be shown to be trustworthy enough to measure the next stage.

But of course, a series of measurements making up a chain of trust must have a starting point, and that starting point is known as the Root of Trust for Measurement (RTM). Among functions for collecting attestation measurements, the RTM is unique in that its operation must be inherently trusted – there's nothing that comes before it to validate its function.

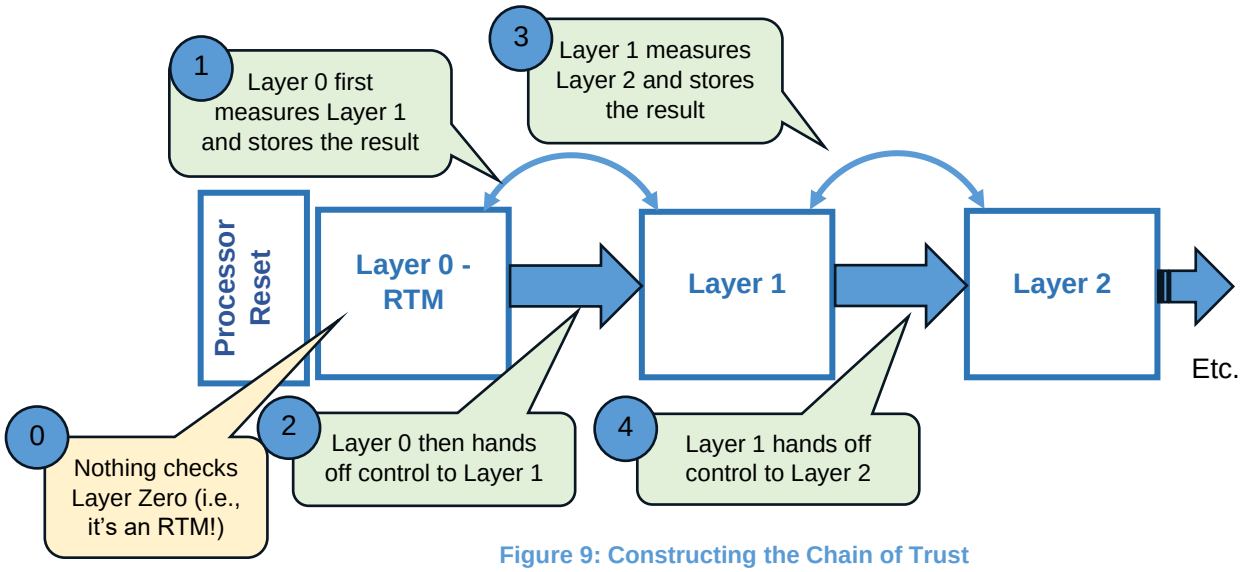

Figure 9: Constructing the Chain of Trust

Ideally, the RTM has exactly one function. Immediately following a reset of the processor complex we want to attest, the RTM should run, measure (i.e., compute the hash of) the Layer 1 code, configuration and keys (also called the First Measured Code (FMC), give the result to the Root of Trust for Storage (RTS; see notes below), then launch the FMC.

The astute reader will notice "ideally"… what does that mean? In many current processor designs, there are a lot of things that must run "first". For example, the processor probably can't do anything until its microcode has been loaded. It probably can't measure anything until its memory controller is working. All of that initialization is inside the RTM.

This leads to a view that an RTM isn't a single thing, but is a collection of components that exist inside a security boundary. There are multiple components inside the RTM boundary (processor instructions, code, logic, initialization functions, etc.), *all* of which must be implicitly trusted to produce that initial measurement of the FMC, but unlike other roots of trust (e.g., the TPM's Root of Trust for Storage), few of the RTM components inside the RTM boundary are dedicated to RTM duties[63]. Designers always strive to minimize the amount of functionality inside the RTM, but processor initialization is highly vendor-specific, so there's no normative definition of what must or must not be inside the RTM boundary, just that it be trustworthy enough to make the first measurement.

[63] For example, if a processor is used to compute the first measurement, then all of the processor's execution pipeline is inside the RTM boundary.

Figure 10 illustrates conceptual models of how the Roots of Trust (including the RTM) might be constructed for a system using TPM, MARS or DICE, using a ROM to hold measurement code, and a processor to execute the code. In the DICE case, as the figure shows, the RTM, RTR and RTS all share one Roots of Trust boundary, with the UDS inside that boundary, with careful controls to ensure that the secret (the UDS) can't leak. In the TPM and MARS cases, explicit hardware RTR and RTS components are provided to store, and protect, and report the measurement, outside the RTM boundary. But in all cases. all the items inside Root of Trust boundaries must be assumed to function correctly to generate a valid first measurement, and ultimately, a valid signed attestation.

Figure 10 also illustrates the effect of the DICE approach of merging measurements and identity. In the DICE example, the same Root of Trust boundary must protect the measurement function (RTM), and also the roots of trust for identity (RTI), reporting (RTR) and storage (RTS). TPM and MARS designs typically use the RTM simply to do the first measurement, and rely on the TPM or MARS to safely store the result and protect relevant keys.

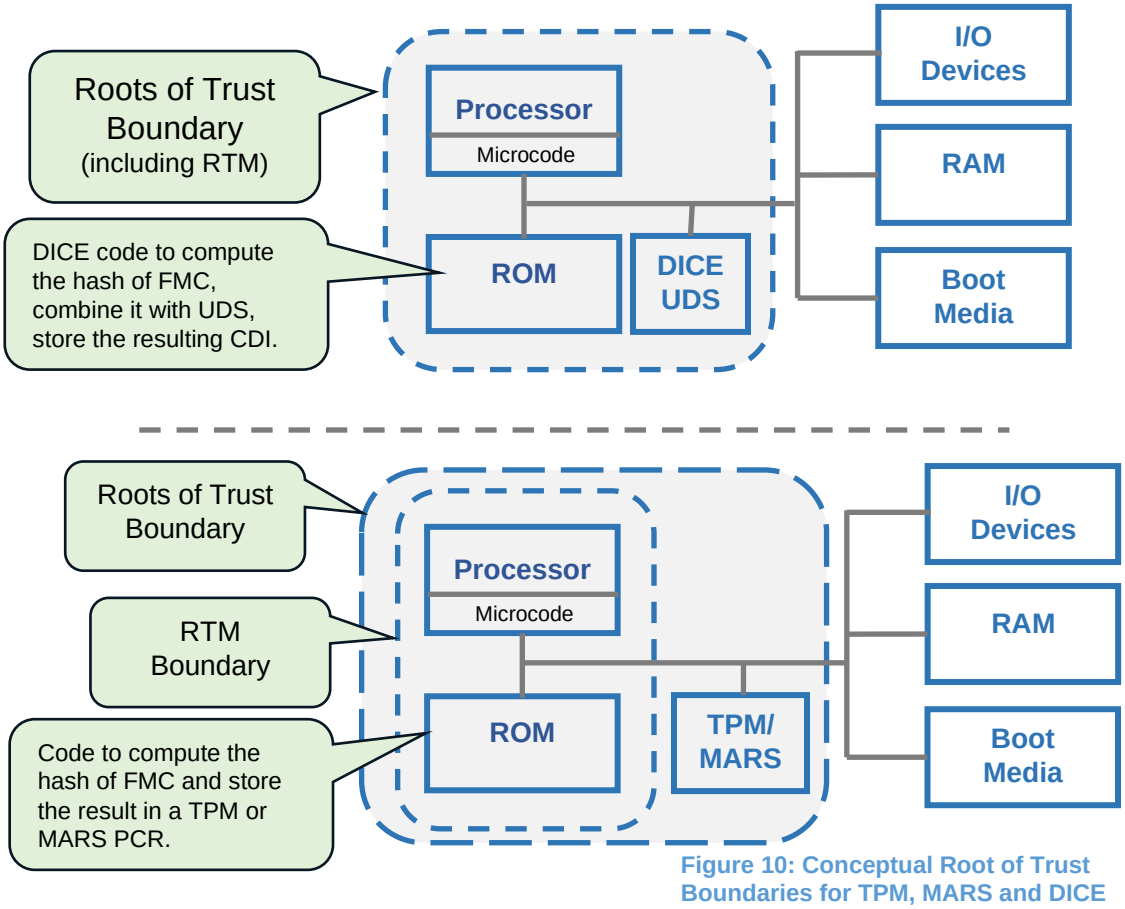

Figure 10: Conceptual Root of Trust Boundaries for TPM, MARS and DICE

(Your Mileage *Will* Vary!)

It should be noted that "real" RTMs would usually be larger, as processors often need microcode to be loaded before running, or objects like memory controllers must be initialized before a measurement can be taken. Conversely, it's also possible that an RTM might not use a processor at all – the First Measured Code measurement function could be "hardware" in FPGA logic or equivalent[64].

[64] The key point being that the elements inside the RTM boundary are completely dependent on device architecture, and not at all amenable to standardization.

When embedded in an SoC, the DICE methodology was designed to minimize the functionality needed to implement Roots of Trust[65]. The DICE Composition Engine must have access to the Universal Device Secret (UDS), but that's a simple hardware register. The RTM must measure the FMC, but that's a hash function. And rather than requiring a hardware RTS, the DICE RTM passes a single value (the Compound Device Identifier) that expresses both the device identity and FMC fingerprint, and erases or hides all other secrets.

In contrast, a system equipped with a TPM may end up with a lot more function inside the RTM boundary; TPMs are usually visible to the operating system, and may need more software initialization to make them usable, enlarging the size of the RTM boundary.

As with DICE and TPM, the MARS specifications are silent on RTM requirements, but, as with other comparison points in this document, MARS lands part way in between – when MARS is implemented as part of an SoC, its RTM will likely be simpler than that of a TPM-based system, but still possibly requiring more RTM capabilities (and certainly more hardware) than DICE to complete and store the first measurement.

5.4.3 RTM and Mutability

Any Root of Trust is defined as a function that must be trusted because it can't be verified in the running system. As such, TCG specifications (e.g., [2]) require the RTM to be immutable from the point of view of someone using or attacking the device.

Manufacturers can decide if they want to make the RTM literally immutable[66] (in which case a bug or cryptographic weakness[67] found later cannot be fixed) or make the RTM code field-updatable, but protected by hardware, coupled with cryptographic means so it can't be changed without manufacturer authorization. Mechanisms and procedures to manage RTM updates are out of scope for all three technologies, although the *TCG Root of Trust Specification* [2] gives background on requirements for Roots of Trust. NIST SP800-193 *Platform Firmware Resiliency Guidelines* [39] also gives detailed guidance on protection of RTMs.

Systems using embedded TPMs, MARS or DICE, where the hardware required will often be implemented as part of an SoC or other silicon device, have the additional advantage that any code used to implement the RTM would be on the same chip as the processor being secured, making it much harder to probe the interaction between the two. TPMs may be implemented as physically-independent chips, which are by nature devices separate from the RTM, yielding more opportunity for probing attacks on the connection between the measurement collection code outside the TPM, and the Shielded Locations inside. See [10], [11] for mitigation strategies.

5.5 The Requirement for Shielded Locations

All Roots of Trust must use memory locations that are shielded in some way to keep secrets and prevent tampering, but the three TCG variants have rather different assumptions.

The TPM implements a rich set of capabilities to offer controlled access to Shielded Locations of various sorts for safely storing keys and measurements.

[65] Note "was designed to"… TCG does not specify RTMs at all; consequently, TCG can say no more than "this approach *should* require less hardware / firmware / software than that one", but it's up to individual system designers to asses whether that's useful advice when mapped to their specific system requirements.

[66] e.g. Read-Only-Memory or hardware state machines

[67] A mathematically-perfect implementation in hardware relying on an MD5 hash might not have a bug, but it would no longer be viewed as secure either.

MARS is based on hardware protection of a unique Primary Seed, plus protection of one or more PCRs. MARS has no capability for Shielded Locations capable of storing arbitrary keys or other data, although keys derived from the Primary Seed may be deterministically generated, used on demand, and forgotten.

The DICE *Hardware Requirements for a Device Identifier Composition Engine* [20] implies a hardware mechanism to hide the device secret (UDS), but has no further requirements for Shielded Locations.

However, it should be noted that quite a few cases in the DICE Layering Architecture [21] assumes some hardware-based mechanism beyond the DICE hardware specification for implementing additional Shielded Locations. This could be DPE (DICE Protection Environment), but many designers will choose to augment DICE designs with vender-specific protection mechanisms such as Intel SGX® or ARM Trust Zone®, as noted in Section 5.7 of the DICE Layering Architecture. Designers and analysts must be cognizant of the security of their Protected Capabilities and Shielded Locations!

5.6 Comparison

TPM, MARS and DICE form several variations of the basic patterns of TCG Root of Trust technology, but they differ on quite a few different axes. Table 2 gives the name of each pattern and a top-level functional summary. Table 3 compares the technologies in terms of their approach to the RTM, RTS and RTR requirements, while subsequent sections outline differences in several specific topic areas.

Pattern	Keying	Summary
TPM Integrated TPM	Both	Shielded storage, signing, measurement and attestation primitives as a self-contained crypto-processor
DICE	Both	Root of Trust for Measurement (RTM) and Identity
DICE and DPE	Both	RTM plus shielded storage
MARS	Either[68]	Measurement and Attestation in a microcontroller IP block

Table 2: TCG Roots of Trust Technologies

	TPM	MARS	DICE
RTM (Root of Trust for Measurement)	All three require the designer to provide, and protect, a functional unit (typically protected firmware) to actually compute the first measurement. DICE hardware provides a unique, secret identity value to be combined with the first measurement by the RTM.		
RTS (Root of Trust for Storage)	TPM and MARS use one or more PCRs to store measurements. Both provide Protected Capabilities allowing the PCRs to be updated, but not arbitrarily changed.		DICE requires each layer to secure its own secrets, typically by computing a new storage value with a one-way function, and erasing the previous value. DPE adds functionality to pass protected storage values from one layer to the next with less chance of leakage.

[68] The MARS assumption is that the designer will configure the MARS IP block for one or the other, although "both" could be supported with enough silicon. See notes in Section 5.6.7.1.

RTR (Root of Trust for Reporting)	Through restricted properties set on Attestation Keys, the TPM can sign attestation reports (“quotes”) in a way that can’t be mimicked by an agent outside the TPM.	MARS hardware signs attestation reports (“quotes”) using a unique key derived from a device secret.	Each layer develops a new reporting key, derived from the reporting key of the previous layer, to serve as the RTR for that layer.

Table 3: RTM, RTS, RTR Comparison

It should be noted that a complex product may contain more than one of the Root of Trust technologies. For example, a product might use DICE or MARS to protect embedded controllers in peripheral units, while using TPM technology to protect a control-plane processor.

5.6.1 Trust Model

All three Roots of Trust, TPM, MARS and DICE, provide a mechanism for attestation as a core feature, but there are two different approaches, illustrated in Figure 11 and Figure 12:

- TPM and MARS use an approach optimized for implementation with dedicated resources[69], that relies on a limited number of PCRs (shielded locations). Each PCR is used to record a hash of all the measurements since boot for the category or layer of objects assigned to that PCR (e.g., measurements of code, keys and configuration for BIOS, loader, and OS, etc). PCRs are used to validate a companion log recording each measured object.
- DICE uses a decentralized approach, optimized for implementation with software, rooted in the Device Identity Composition Engine, but with each layer responsible for managing its own measurements, secrets and keys. Because there’s no centralized resource, designers can record any number of measurements, in any number of categories, as best fits the application.

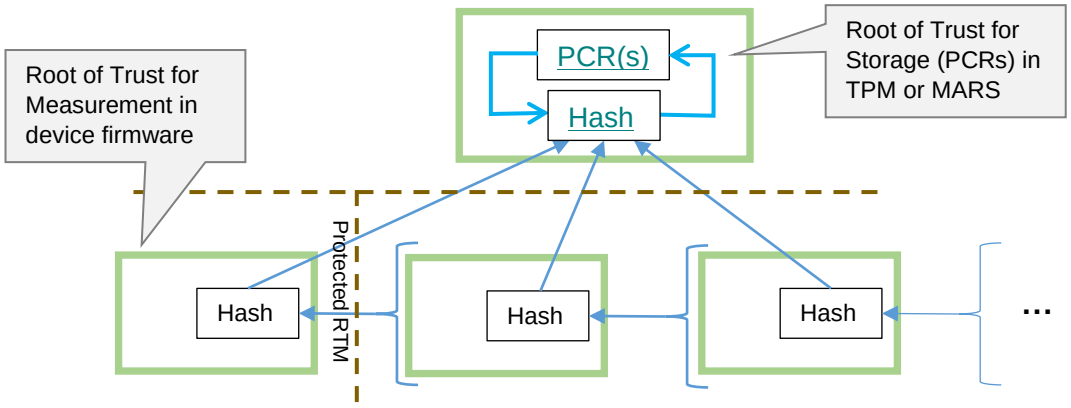

Figure 11: Chain of Trust via PCR

[69] “Dedicated resources” might mean a chip (i.e. a TPM) or dedicated gates in an ASIC, or less-obvious dedicated resources such as protected memory and firmware for an embedded TPM.

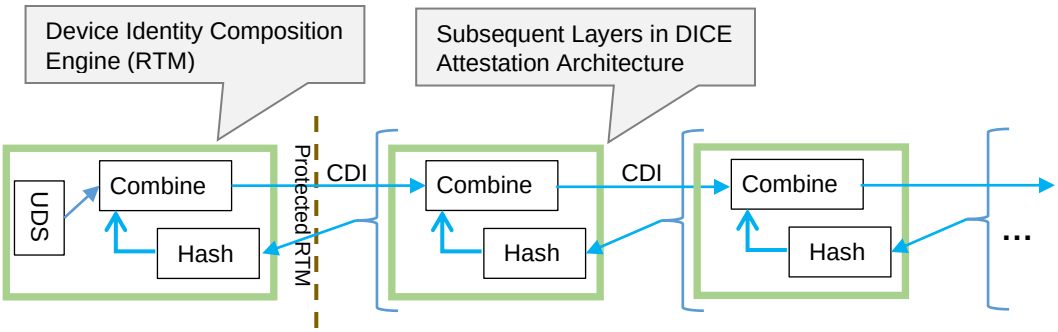

Figure 12: Chain of Trust via CDI

5.6.2 Attestation

TPM, MARS and DICE can all be configured for attestation.

The TPM defined the original model for TCG Attestation – measurements stored in PCRs, then retrieved later as a signed Quote for analysis at a Verifier (see IETF RIV [31]). MARS follows the same model, scaled back to match its resource constraints. Validation is done in two steps:

- The device identity is validated by examining the Attestation Key certificate, signature and freshness.
- The authenticity of software loaded on the device is validated by checking the TCG event log against the quoted PCR values, and then comparing that to a set of reference measurements.

DICE uses a different model for attestation. As with the TPM, a 'chain of trust' model is used, starting with a Root of Trust and requiring each layer to measure the next before starting it. But unlike the TPM or MARS, the result is the chain of x.509 certificates created during the startup of a DICE device. This chain can be transferred to a Verifier for analysis, confirming the identity of the device and its software load. Section 5.3.1 gives an outline of how the DICE attestation chain is validated.

5.6.2.1 Comparison

The two attestation models have differing properties:

- TPM and MARS encourage an attestation implementation with all attestation results collected in one place (the PCRs and accompanying log), all accessed by a single mechanism (e.g., see IETF CHARRA [32]).
- The DICE approach to storage of attestation is highly distributed, and requires no coordination of central resources. If extended beyond a singular TCB, each DICE application can manage its own layers and attestation chains.

If extended to large systems, DICE may require a lot of certificates to be generated and signed during each initialization, possibly posing a scaling issue.

5.6.3 Selective Release of Attestation Evidence

TCG Attestation assumes, as a minimum, that enough attestation evidence is collected to demonstrate the integrity of a base level of level of system software[70], which then is relied upon to ensure that the rest of the device's software meets security expectations. In this view, the TCG-specified roots of trust are used to collect enough measurements to verify the trusted computing base, and then hold that evidence for later analysis by a Verifier. However, there's no rule that says that the same TCG mechanisms can't be used beyond the establishment of a

[70] This base level of system software is sometimes called a Trusted Computing Base

trusted base, and in fact, Linux IMA can be configured to do just that, potentially recording measurements of every file system object as they are used or activated.

In cases where the attestation mechanism is used beyond the establishment of a base level of security, the DICE and TPM approaches differ in their transfer of attestation evidence from the device. Of course, it's up to the device's firmware and operating system to select what attestation information will be collected[71], but:

- Given the centralized PCRs of a TPM or MARS, attestation must report every measurement extended into a given PCR, with implications concerning the ability of the device to select which evidence to send to which verifier[72]. This is generally the desired result in many Enterprise-like environments, where computing resources are centrally managed, and the administrators have an interest in knowing that applications that should be there are intact, and that applications that shouldn't be there aren't.
- The decentralized DICE approach does attestation by following a certificate chain from leaves back to the root. As such, DICE can report attestation evidence for one leaf (e.g., one application) without revealing the rest[73]. Indeed, DICE has no standardized mechanism to even find all the leaves that have been attested. In effect, DICE allows verifiers for different applications to work without revealing each other's attestation evidence.

Designers should note that privacy considerations may require mechanisms to allow end-users granular control over use of identity, e.g., in some cases, it may be necessary to carry out attestation while protecting or masking the identity of the attester, while in others, there may be a requirement to prevent anonymous operation. All of the TCG technologies provide primitives to control the use of identity information, but the choice of how to use the primitives may require subtle application-specific tradeoffs by the designer.

5.6.4 System vs SOC Implementation

One obvious difference is that the TPM is available as a physical chip, while DICE and MARS are intended to be implemented as logic blocks in an SoC[74]. As such, OEMs and system designers who do not make their own silicon must either use a TPM, or chose a silicon vendor's SoC that already contains the technology.

- TPM is well-suited for use by system designers who do not design their own silicon, and may require a multi-vendor solution.
- DICE and MARS are both well-suited to embedding in silicon projects such as SoCs or microcontrollers.
- DPE could be a separate component like a TPM chip, but is more likely suited to integration into an SoC or implementation in a vendor's trusted enclave technology.

Of course, applications where the cost or space budget preclude the addition of even a small TPM chip (e.g., consumer IoT or mobile phones), then a Root of Trust integrated into the SoC using MARS, DICE or an embedded TPM, is the obvious choice.

5.6.4.1 Associated Functions

Most stand-alone TPM 2.0 chips meet both the TPM2 Library Specification (defining the API), and the TCG PC Client Profile [4], which requires a number of ancillary functions, such as:

---

[71] The TPM can also exercise granular control over which keys are usable in what situation, yielding further control over signed attestation reports; see Section 5.6.7.3.

[72] Different categories of measurements can be isolated from one another by careful allocation of the 24 PCRs. But in the conventional implementation [4] there's only one PCR allocated to all measurements made by the operating system. That would be, for instance, PCR 10, authenticating the contents of the IMA log in Linux.

[73] For example, on your phone, you might want your banking app to be subject to attestation by the bank, without revealing the presence of other applications on the phone. Conversely, an admin might want to know everything that is running on a router, to make sure there are no unexpected processes implanted to exfiltrate user secrets.

[74] We note that pretty much any function that could be implemented in hardware can also be implemented in firmware, microcode, or software, with resulting differences in resistance to attack. Mind your Threat Model!

- An Entropy source for generating new keys[75].
- Non-volatile memory to provide persistent storage for keys and other sensitive information through reboot cycles.
- Sealing, the ability to specify policies for circumstances under which sensitive information can be used or accessed.

Neither DICE nor MARS include these functions, so they would have to be provided by mechanisms that are not specified by TCG if they're needed by the application[76].

5.6.5 Persistent Identity Key

The TPM provides considerable flexibility for creating and storing long-term persistent identity keys, allowing a variety of mechanisms for identity (including exact compliance with IEEE 802.1AR). TPMs are usually shipped with a unique Endorsement Key (EK), which also can be used as a form of persistent identity[77].

When equipped for asymmetric keys, the MARS KDF can be used to transform its initial random seed into unique long-term stable key pairs suitable for identity and attestation.

The fundamental structure for DICE is the *Compound Device Identifier*, a value that reflects both the identity of the hardware and also of the currently-installed layers of TCB software, so keys developed by DICE by default depend on software version as well as hardware identity[78].

The TCG document *Implicit Identity Based Device Attestation* [23] provides an alternate mechanism for Device ID that does not rely on hardware-protected Shielded Locations, but introduces a new chain of certificates parallel to the DICE Attestation chain, ultimately relying on the final layer of the DICE layer stack to protect a derived private identity key.

5.6.6 Platform Certificate

For supply-chain tracking, the 2.0 version of TCG Platform Certificate Profile [7] specifies that the Platform Certificate and its ancillary Delta Certificates may be bound to the TPM by either an EK or a DevID.

There's currently no TCG specification for binding platform certificates to devices using DICE or MARS.

5.6.7 Key Management in Manufacturing Supply Chain

5.6.7.1 Symmetric vs Asymmetric Keying

All cryptography relies on an ability to keep secret keys secret. But the two common techniques, Symmetric Key cryptography (e.g., AES) and Asymmetric Key cryptography (e.g., RSA, Elliptic curve) have significant differences in the way keys must be managed.

For symmetric key cryptography, the two parties involved in an exchange must have access to the same secret key. For example, to prove identity, the Attester might compute a Message Authentication Code of a nonce with its copy of the secret key, allowing the Verifier to verify the nonce using its (identical) copy of the secret.

[75] While TPMs are expected to include an entropy generator, it should be noted that some applications require only a limited amount of entropy (e.g. a disposable smart transit card), which could be derived from the DICE UDS or MARS primary seed.

[76] Although not always required, cryptographic-quality entropy for key generation is especially difficult. See NIST SP 800 90B [41].

[77] Although the EK is burdened with usage restrictions to protect privacy by default. (i.e., the EK can decrypt, it can certify other keys, but it can't sign)

[78] Some would view this as a feature, not a bug, circumventing, as they see it, a fundamental weakness in 802.1AR by allowing (and ultimately requiring) that the device identity key be refreshed if anything changes in the TCB, including the addition of new crypto algorithms.

In asymmetric key cryptography, the same exchange allows the Attester to sign the nonce with its private key, and the Verifier to verify it with the Attester's public key. The secret is never shared, and may not be known to anyone outside the Attester's key store.

This difference can have an important impact in the manufacturing flow; in the asymmetric case, the manufacturer may not ever know the private key that's unique to each device, whereas in the symmetric case, the manufacturer may be responsible to generating and configuring the unique secret for each device, and may also be responsible for retaining that secret in a way that it's accessible to the authorized users, but doesn't leak into the wrong hands.

Asymmetric cryptography does come at a cost: circuitry to implement asymmetric encryption typically more gates, and may be considerably slower.

Symmetric cryptography is believed to be more resistant to quantum-computing attacks, while existing, standardized asymmetric algorithms such as RSA and ECC are thought to be vulnerable[79].

5.6.7.2 Comparison

All three technologies can be used with symmetric or asymmetric cryptography, but there are differences.

- The TPM can carry out functions with symmetric or asymmetric cryptography, but TCG's comprehensive ecosystem around the TPM, such as specifications for TPM key-provisioning, currently focuses mostly on asymmetric applications.
- MARS hardware can be configured for either key type, with a smaller silicon footprint if the symmetric option is exercised.
- The choice between symmetric or asymmetric cryptography has no impact on DICE hardware requirements.

The TPM is also capable of generating keys internally, with the result that it's possible to create private keys that cannot be exposed[80]. MARS and DICE require external generation of a secret cryptographic seed during a provisioning process, either by the silicon vendor or the system builder. None of the TCG specifications cover that initial provisioning step, or offer guidance on managing symmetric keys, if they are provisioned.

5.6.7.3 Conditional Access to Keys

The TPM has a variety of mechanisms to control access to keys. For example, keys may be bound to a particular machine state, and the TPM's AK will not sign data external to the TPM that could be confused with a Quote. In addition, the TPM2.0 Policy mechanism (see Section 19.7 *Enhanced Authorization* in *Trusted Platform Module Library Part I* [3]) can be used by the system designer to implement arbitrary state machines to control access to TPM keys and resources.

Conversely, DICE (and DICE/DPE) have keys which are unique to a given layer, and "forgotten" once that layer exits, but no standard mechanism exists for more granular access controls.

5.6.8 Additional Hardware for RTM and Shielded Locations

Each of TPM, MARS and DICE require hardware assistance to protect the RTM function, but none give detailed requirements for the expected RTM. TCG does offer guidance on considerations in selecting a Root of Trust for Measurement (see [2]).

See additional RTM notes in Sections 5.4 and 5.5.

The three technologies take different approaches on Shielded Locations:

- The TPM provides a rich set of functions for Shielded Locations, i.e., NVRAM, PCRs, key storage, etc.

[79] NIST reports on status of quantum-resistant cryptography development at [42]

[80] Although manufacturing processes that require very fast key generation for pre-provisioned keys (e.g. EK, IDevID, IAK) might rely on an external key generator to create key pairs, program the TPM, then forget the private keys.

- MARS provides hardware to protect attestation state (one or more PCRs), the Primary Seed and any keys derived from the Primary Seed. Other forms of Shielded Locations are out of scope for MARS.
- DICE relies upon hardware shielding for the device secret (UDS), but expects that each layer will protect its keys and secrets in its own way as appropriate for the application and use case. Many applications will require additional mechanisms for shielding, such as vendor mechanisms like ARM's Trust Zone®, Intel's SGX® or the DICE Protection Environment (DPE).

5.6.9 Requirements for Silicon Resources

Any Root of Trust clearly has a hardware impact, but the different options outlined above have different levels of impact on the design of a device, as sketched in Table 4.

DICE	Small Impact in an SoC, requiring a 256-bit register and latch to manage the Universal Device Secret
MARS and DICE/DPE	Modest Impact in an SoC - fits into a corner of an SoC, but designers may be well advised to search for existing intellectual property. MARS and DPE each may be suitable for firmware implementation in a Trusted Execution Environment.
TPM Chip	Expect a separate packaged chip, with maximum power less than 0.1 Watt in a 5mm-square BGA package[81].

Table 4: Hardware Impact of Root of Trust

5.6.10 Software Support

A hardware Root of Trust may be the only way to achieve trusted computing, but hardware alone isn't enough without software to use it. The TCG Roots of Trust offer rather different approaches to software support.

[81] Alternative TPM implementations such as Firmware TPM or the Automotive Thin Profile [6] may require a smaller silicon profile.

5.6.10.1 TPM

The TPM offers both a well-developed set of documented APIs, and an open-source implementation. In addition, Microsoft Windows® ships with a built-in TPM stack, integrated with the platform's key management utilities.

Within TCG, there is a set of TPM API documents, starting with an overview document, *TCG TSS 2.0 Overview and Common Structures Specification* [9]. Documents describing various layers in the software stack are shown in Figure 13.

The Linux implementation of a TPM stack can be found at

https://github.com/tpm2-software/tpm2-tss

A reference implementation of the TPM2 itself can be found at https://github.com/microsoft/ms-tpm-20-ref.

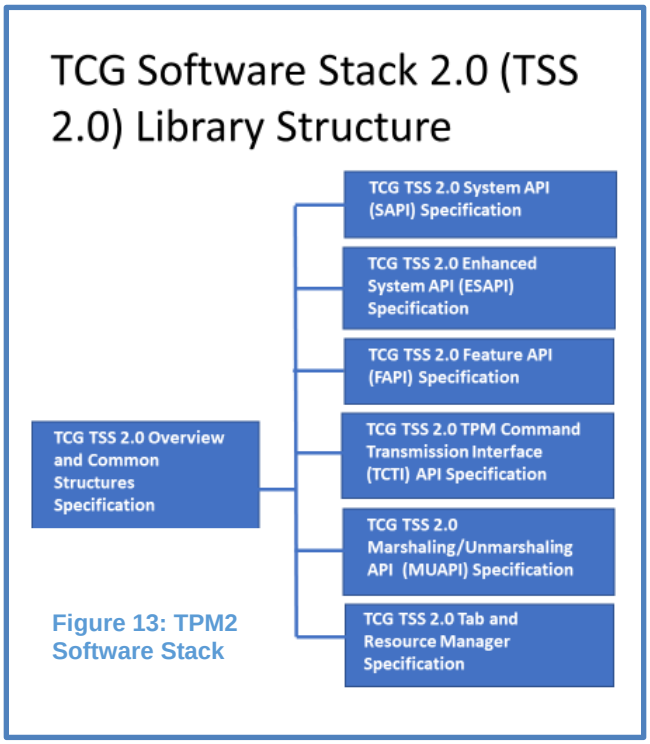

Figure 13: TPM2 Software Stack

5.6.10.2 MARS

There are several TCG documents that describe the MARS API and a library of routines to access the function:

MARS API https://trustedcomputinggroup.org/resource/tcg-mars-api-specification-version-1-revision-1/

MARS Library https://trustedcomputinggroup.org/resource/mars-library-specification/

MARS C and Python Emulators: https://github.com/TrustedComputingGroup/MARS

5.6.10.3 DICE

There are currently no TCG API documents for DICE, although several different source code bases are available

- Microsoft has published DICE code under the code name RIoT: https://github.com/Microsoft/RIoT
- Android code contains an implementation of DICE: https://pigweed.googlesource.com/open-dice
- Another implementation of DICE can be found at Veraison: https://github.com/veraison/veraison

5.6.11 Tamper Resistance

Discrete TPM chips (as opposed to firmware implementations) by definition are independent chips attached to a circuit board, offering an attacker the opportunity to probe or interpose the interfaces to the chip. This doesn't imply that a TPM is easy to compromise, but it does enlarge the attack surface (see [10], [11] for mitigation strategies). In

addition, TPMs that meet TCG Compliance and Security Evaluation requirements[82] are required use techniques that resist physical tampering.

DICE and MARS are intended to be implemented as part of an SoC or silicon device, and in that context, can often be hidden, along with the Root of Trust, completely inside the chip, with no pins to probe or components to swap, thus rendering physical attacks more difficult.

As a separate chip, devices using a TPM are often at risk of having the TPM replaced with a different one, effectively changing the device's identity, increasing reliance on vendor-specific approaches to physical tamper resistance. Conversely, SoC designs containing Integrated TPMs, DICE or MARS are more likely to be self-contained, and harder for an attacker to swap out.

Of course, in all cases, the choice of technologies should be predicated on the threat model envisioned by the designer.

6 Summary

System vendors have a number of choices to implement trusted computing, allowing solutions to be tuned for specific applications. These applications are often derived from various security use cases – such as supply chain security, remote attestation of device health/trustworthiness, the protection of a Shielded Location, etc. – each of which suggests different requirements. Vendors must also consider customer expectations for how a device will be used to determine the extent to which privacy, scalability, updatability, etc. must be incorporated into how trusted computing technologies are incorporated into a product.

Trusted computing is not binary; a device is not necessarily either "trusted" or "untrusted". Trusted computing requires architects, designers and users to evaluate the threats posed by their specific environments, and then to exercise considerable judgement as to what's trustworthy and what's not. Ultimately, users of trusted computing must evaluate trustworthiness of the suppliers of the technology: specifications, compliance testing, data books, etc., can only set out the guard rails. Vendors, though, ultimately hold the responsibility for making sure that secure solutions are in fact adequately secure. Vendors should leverage the information in this document, in conjunction with their product uses cases and requirements, to make the right choice for incorporating trusted computing into their products.

7 References

[1] TCG Glossary Version 1.1 - https://trustedcomputinggroup.org/resource/tcg-glossary/

[2] TCG Root of Trust Specification - https://trustedcomputinggroup.org/wp-content/uploads/TCG_Roots_of_Trust_Specification_v0p20_PUBLIC_REVIEW.pdf

[3] TPM 2.0 *Trusted Platform Module Library Family "2.0" Specification - Parts 1-4 and Code, Revision 1.59* https://trustedcomputinggroup.org/resource/tpm-library-specification/

[4] *TCG PC Client Platform TPM Profile Specification for TPM 2.0, Version 1.05, Revision 14*, September 4, 2020, https://trustedcomputinggroup.org/resource/pc-client-platform-tpm-profile-ptp-specification/

[5] *TCG PC Client Specific Platform Firmware Profile Specification*; https://trustedcomputinggroup.org/resource/pc-client-specific-platform-firmware-profile-specification/

[6] TCG *Protection Profile Automotive-Thin Specific TPM for TCG TPM 2.0 Automotive Thin Profile Family "2.0"* https://trustedcomputinggroup.org/resource/protection-profile-automotive-thin-specific-tpm-for-tcg-tpm-2-0-automotive-thin-profile-family-2-0-level-0/

[82] See https://trustedcomputinggroup.org/membership/certification/ for information on the TCG TPM Certification Program.

[7] *TPM 2.0 Mobile Reference Architecture*, Revision 142, 16 December 2014, https://trustedcomputinggroup.org/resource/tpm-2-0-mobile-reference-architecture-specification/
[8] *TCG Platform Certificate Profile*, Version 1 - https://trustedcomputinggroup.org/resource/tcg-platform-certificate-profile/
[9] *TCG TSS 2.0 Overview and Common Structures Specification* https://trustedcomputinggroup.org/resource/tss-overview-common-structures-specification/
[10] TCG *CPU-TPM Bus Protection Guidance for Active Probing Attacks* https://trustedcomputinggroup.org/resource/tcg-cpu-to-tpm-bus-protection-guidance-for-active-attacks/
[11] TCG CPU-TPM Bus Protection Guidance – Passive Probing Attacks [public review] https://trustedcomputinggroup.org/wp-content/uploads/TCG_CPU_TPM_Bus_Protection_Guidance_Passive_Attack_Mitigation_8May23-3.pdf
[12] *TCG PC Client Platform Firmware Integrity Measurement Specification, Version 1.0 Revision 43*, May 7 2021, https://trustedcomputinggroup.org/resource/tcg-pc-client-platform-firmware-integrity-measurement/
[13] *TCG Reference Integrity Manifest (RIM) Information Model*, Version 1.0, Revision 0.16, Nov 12, 2020 https://trustedcomputinggroup.org/resource/tcg-reference-integrity-manifest-rim-information-model/
[14] *TCG PC Client Specific TPM Interface Specification (TIS), Version 1.3, 21 March 2013*, https://trustedcomputinggroup.org/resource/pc-client-work-group-pc-client-specific-tpm-interface-specification-tis/
[15] IETF Remote Attestation ProcedureS (RATS) Architecture, https://datatracker.ietf.org/doc/rfc9334/
[16] *TPM 2.0 Keys for Device Identity and Attestation* https://trustedcomputinggroup.org/resource/tpm-2-0-keys-for-device-identity-and-attestation/
[17] TCG TPM v2.0 Provisioning Guidance https://trustedcomputinggroup.org/resource/tcg-tpm-v2-0-provisioning-guidance/
[18] *Virtualized Trusted Platform Architecture Specification, Version 1.0, Revision 0.26*, September 27, 2011, https://trustedcomputinggroup.org/resource/virtualized-trusted-platform-architecture-specification/
[19] TCG Canonical Event Log Format https://trustedcomputinggroup.org/resource/canonical-event-log-format/
[20] DICE *Hardware Requirements for a Device Identifier Composition Engine* https://trustedcomputinggroup.org/resource/hardware-requirements-for-a-device-identifier-composition-engine/
[21] DICE Layering Architecture - https://trustedcomputinggroup.org/wp-content/uploads/DICE-Layering-Architecture-r19_pub.pdf
[22] DICE Attestation Architecture - https://trustedcomputinggroup.org/resource/dice-attestation-architecture/
[23] DICE Implicit Identity Based Device Attestation - https://trustedcomputinggroup.org/resource/implicit-identity-based-device-attestation/
[24] DICE Symmetric Keying - https://trustedcomputinggroup.org/resource/symmetric-identity-based-device-attestation/
[25] DICE Protection Environment - [public review] https://trustedcomputinggroup.org/wp-content/uploads/TCG-DICE-Protection-Environment-Specification_14february2023-1.pdf
[26] *MARS API Specification* https://trustedcomputinggroup.org/resource/tcg-mars-api-specification-version-1-revision-1/
[27] *Measurement and Attestation RootS (MARS) Library Specification*, https://trustedcomputinggroup.org/resource/mars-library-specification/
[28] *TCG Guidance for Securing Industrial Control Systems Using TCG Technology* - https://trustedcomputinggroup.org/resource/tcg-guidance-for-securing-industrial-control-systems-using-tcg-technology/
[29] TCG Network Equipment - https://trustedcomputinggroup.org/resource/tcg-guidance-securing-network-equipment/
[30] *Trusted Computing Platforms: TPM2.0 in Context*, Graeme Proudler, Liqun Chen, Chris Dalton, Springer 2014
[31] IETF RIV *TPM-based Network Device Remote Integrity Verification* - https://datatracker.ietf.org/doc/draft-ietf-rats-tpm-based-network-device-attest/
[32] IETF CHARRA *A YANG Data Model for Challenge-Response-based Remote Attestation Procedures using TPM* - https://datatracker.ietf.org/doc/html/draft-ietf-rats-yang-tpm-charra-21
[33] [IETF] Secure Asset Transfer (SAT) Interoperability Architecture - https://datatracker.ietf.org/doc/draft-hardjono-sat-architecture/
[34] IETF *Secure Zero Touch Provisioning* RFC-8572 - https://datatracker.ietf.org/doc/rfc8572/
[35] IETF *Time-Based Uni-Directional Attestation* (TUDA) https://datatracker.ietf.org/doc/draft-birkholz-rats-tuda/
[36] IETF TLS *The Transport Layer Security (TLS) Protocol Version 1.3* RFC-8446 https://datatracker.ietf.org/doc/rfc8446/
[37] IEEE 802.1AR-2018 IEEE Standard for Local and Metropolitan Area Networks - Secure Device Identity https://standards.ieee.org/ieee/802.1AR/6995/
[38] Linux Integrity Measurement Architecture (IMA), http://linux-ima.sourceforge.net, https://wiki.gentoo.org/wiki/Project:Integrity
[39] NIST SP800-193 *Platform Firmware Resiliency Guidelines* https://csrc.nist.gov/publications/detail/sp/800-193/final
[40] NIST-SP800-147 *BIOS Protection Guidelines*, https://csrc.nist.gov/publications/detail/sp/800-147/final
[41] NIST- SP 800-90B *Recommendation for the Entropy Sources Used for Random Bit Generation*, https://csrc.nist.gov/publications/detail/sp/800-90b/final
[42] NIST Post-Quantum Cryptography Overview https://csrc.nist.gov/projects/post-quantum-cryptography
[43] SERIES X: DATA NETWORKS, OPEN SYSTEM COMMUNICATIONS AND SECURITY – The Directory: *Public-key and attribute certificate frameworks; Recommendation ITU-T X.509*, https://www.itu.int/rec/T-REC-X.509-201910-I/en

