TRUSTED® COMPUTING GROUP

REFERENCE

MARS Use Cases and Considerations

Version 1
Revision 26
October 29, 2020

Contact: admin@trustedcomputinggroup.org

PUBLIC REVIEW

Work in Progress

This document is an intermediate draft for comment only and is subject to change without notice. Readers should not design products based on this document.

DISCLAIMERS, NOTICES, AND LICENSE TERMS

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

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

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

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

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

Acknowledgements

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

Tom Broström, Cyber Pack Ventures, Inc.

Dave Challener, Johns Hopkins University, Applied Physics Lab

Ga-Wai Chin, Infineon Technologies

Brian Dziki, United States Government

Guy Fedorkow, Juniper Networks, Inc.

Sukhjinder Hayer, The Boeing Company

Steve Luther, United States Government

Graeme Proudler, Invited Expert

Josh Schiffman, HP Inc.

Vadim Sukhomlinov, Google Inc.

CONTENTS

1 INTRODUCTION

Microcontrollers and SoCs supporting embedded and more complex devices are available with many different feature sets. At the high end, microcontrollers may contain sophisticated 32-bit microprocessors, abundant volatile and non-volatile memories, a large cache, floating point support, protected execution modes, I/O options including support for audio, video, networking and DSP. At the low end, there are microcontrollers with 8-bit microprocessors with minimal additional support.

Embedded systems manufacturers, especially those considering use of lower end microcontrollers, may find it difficult to build devices that comply with the TCG's measurement and attestation (M&A) framework. The Roots of Trust that protect the resources and mechanisms that support measurement recording and reporting are found in a Trusted Platform Module (TPM), but the inclusion in an embedded system of a separate TPM may be untenable due to power, space, and cost. Another solution is needed.

The purpose of Measurement and Attestation RootS (MARS) is to specify the means by which microcontroller manufacturers may directly incorporate the required Root of Trust (RoT) logic in their designs in an appealing, low-overhead manner. The result should be a device that natively supports the "minimum functionality necessary to describe characteristics that affect a platform's trustworthiness" as required by the TCG [1].

The performance of measuring, recording, and reporting comprise the essential features of a MARS-equipped device.

1.1 Purpose

The set of use cases and considerations described in this reference document determine the requirements for a Measurement and Attestation RootS (MARS) specification.

1.2 Scope

The MARS Charter requires the MARS specification to enable M&A. Other features are considered (per the Charter) and may be included in the specification as Recommended or Optional if they are deemed particularly useful to embedded systems and add a minimum to the complexity for implementors. The Use Cases presented in this reference will help determine the capabilities and commands to be support by MARS in order to meet M&A and "other" needs.

1.3 Audience

The intended audience for this reference document is manufacturers of microcontrollers and those considering incorporating MARS in their design. Readers are encouraged to understand the Trusted Platform concepts detailed in the TPM Architecture specification [1].

1.4 Document Organization

In this reference, a set of use cases is first laid out in section 2. This is followed by a set of resource constraints in section 3. The implementation considerations in section 4 discuss how MARS can satisfy the use cases in light of the resource constraints. In general, the subsections in sections 2 and 3 have matching subsections in section 4.

2 USE CASES

As the name implies, Measurement and Attestation RootS is intended to support those use cases that thrive on trusted measurement and attestation primitives. The umbrella use case for MARS was summed up originally by the TPM specification when describing the need for roots of trust – to “describe characteristics that affect a platform’s trustworthiness.” [1] That description is reported to a remote challenger for the purpose of providing access control in a process known as remote attestation. A small subset of a TPM’s extensive capabilities are essential for ensuring that this can be done securely. It is this subset that forms the mandatory capabilities of a MARS-compliant device. Other use cases described in this section may be supported in the MARS Specification as either Recommended for highly desirable features or Optional for those with minimal impact on design.

2.1 Anti-Counterfeit (Device Identity)

Providing reliable remotely-verifiable device identity for each device, be it large or small, is a prerequisite for most security-related use cases. Device Identity may be used as a simple access control mechanism, to prove ownership, or to prove that the device is genuine. A device identity is meaningful when the device can prove that the identity belongs exclusively to that device. While the identity is not secret, it may be privacy-sensitive. The identity may be generated locally or provisioned remotely and stored, as needed, by the host device. Malicious alteration would only yield denial of service as the device would be incapable of proving association with a different identity.

2.2 Boot Health for Access Control Decisions

At the moment when two devices begin interaction, each may need to make an access control decision before that interaction proceeds. In this use case, that decision is based upon an understanding of the other device’s health. Here, that health is understood to reflect the integrity of the components used to boot the device (e.g. boot ROM, second stage boot loaders, firmware). Concerns about boot health may be related to the firmware or configuration being malicious, unauthorized, obsolete, or unknown. Firmware and configuration need to be measured and recorded locally, and eventually reported to another device so that it, or an authorized agent, can make an appropriate health assessment of the platform.

2.2.1 Measuring Device Integrity

The TCG method for representing firmware and configuration modules is hashing their contents to produce digests. As a digest is statistically unique, it is said to identify its module. The usage of a module by a device is recorded as an event. The collection of digests representing these events is called the Event Log.

2.2.2 Recording Device Integrity

The Event Log generally summarizes a device’s integrity. However, since the Event Log can grow large, it is unreasonable to expect that a small RoT would store it. Consequently, its storage has to be managed by the host. As this Event Log is a potential target for malware that might like to cover its tracks, the log’s integrity needs to be managed by a separate, trusted agent such as MARS. Unauthorized alterations to the Event Log should be detectable, rendering the log unusable. See section 4.5.2.

2.2.3 Reporting Device Integrity

When two devices need to communicate with each other to access and provide a service or some other peer-to-peer relationship, one or both may need to prove their identity and configuration so the other can make an appropriate access control decision. The process of preparing a verifiable statement of identity and configuration to be assessed by, or on behalf of, a relying party (e.g. service provider) is known as remote attestation. It is also possible for a device to assess itself in a process known as local attestation. A form of local attestation is covered in section 2.3.

2.3 Sealed Storage

Software modules that manage sensitive (secret or private) data may need to store that data securely. Like the Event Log, it is expected that the data would be stored outside of the RoT and be integrity protected. However, since the data is sensitive, that data will also require confidentiality protection. The device should only be able to access (decrypt) the data if it is on the same device and in an authorized state. Sealed storage is a form of local attestation.

An example of sealed storage commonly occurs in a firmware TPM (fTPM). The fTPM needs to be able to seal its sensitive state to the host device so that the fTPM can be protected at rest. A MARS equipped host could generate a key that is bound to the host device and the host's state. This key could then be used to by the fTPM to encrypt/decrypt its state.

2.4 DICE Emulation

A Device Identifier Composition Engine (DICE, [2]) produces a Compound Device Identifier (CDI) that is derived from a Unique Device Secret (UDS) and the digest of the First Mutable Code (FMC). While the specific method to combine the values is the manufacturer's choice, because it does not affect interoperability, it may be implemented with methods such as

CDI = Hash(UDS || Hash(FMC))

or

CDI = HMAC(UDS, Hash(FMC))

DICE-based applications subsequently use values derived from the CDI in unique ways specified by the DICE Architectures Working Group. If MARS could produce a CDI, then a device equipped with MARS would be compatible with DICE infrastructure, so both DICE and MARS would benefit from economy-of-scale.

2.5 Deep Quote

Applications (e.g. virtualized TPM, fTPM, DICE-based applications) that manage secrets outside of a hardware RoT may need to prove that these secrets are [still] being used on the correct device in the correct state. A deep quote (or deep attestation, [3]) process would enable these applications that already produce attestations of their own to reach down to MARS to provide this proof via an additional attestation from a hardware RoT.

2.6 Setting a GPIO, based on device state

A General Purpose Input Output (GPIO) output could be wired to signal a peripheral, likely tightly integrated to the device, to perform some action – e.g. [un]lock, power-up, blink, etc. Access to the GPIO mechanism would only be possible when the device is in an acceptable state.

2.7 Locking external peripherals to a device

A peripheral (external to MARS or the device) could be configured to respond to a stimulus that only a specific device can produce, regardless of that device's state.

An example of a lockable peripheral would be an SD card encrypted with a key that can only be generated by a specific MARS. When separated from that specific MARS, the SD card cannot be read.

2.8 Locking use of peripherals to a device state

A peripheral could be [un]locked by configuring the peripheral to respond when MARS can produce the correct stimulus. This stimulus would only be produced by a device with a specific MARS and when the device is in the correct state.

Another example of a lockable peripheral is a fingerprint reader that can only be used if the host firmware that will verify the fingerprint is in an expected state on the expected host.

2.9 Identifying how long a system has been booted

An enterprise might want to know how long a device has been running. If that device can relay its boot time accurately, an enterprise with concerns such as device stability or measurement staleness (the likelihood that the digests of what booted no longer represent what remains running or what will boot again) can make well-informed decisions.

2.10 Verified Boot

The TCG emphasis on *measured boot* defers verification to an external entity trusted by the relying party. However, in a *verified boot* (sometimes called "secure boot"), the verification is performed locally. The "known good" or "golden measurement" for a module is attached to (or associated with) that module in the form of a digital signature. During boot, the module is measured and compared to the decrypted signature. If equal (and the signer is trusted) the module is verified. A similar mechanism could be supported by MARS.

2.11 Verified Update

Many devices have upgradable firmware. Signed firmware can be authenticated before applying it as an update. As with the Verified Boot use case, MARS could provide some local verification support. Both Verified Boot and Verified Update are useful layers of security in preventing and detecting unauthorized modifications.

2.12 Update Protection

When a device prepares for an update, it should only be able to overwrite sensitive firmware (e.g. from section 2.2.1) when executing the First Update Engine (FUE, [4]). Attempts to overwrite protected firmware outside of FUE's context should not be possible.

2.13 Securely Proving Knowledge of a Password

A password control mechanism can be used to authenticate local administrators or other users. A Shadow Password table, implemented on many platforms, typically contains:

$$\text{Userid, HashAlgorithm, number of iterations, salt, Hash}^{\text{interations}}\text{(salt || password)}$$

The table contains all the information necessary to perform a remote dictionary or hammering attack on the passwords. A stronger mechanism that guards against these attacks is needed. See section 4.10.

2.14 Chain of Custody

A distributed ledger technology (any kind of append-only database - with immutable records) may be used to track change in ownership transactions for a particular device. A device that presents its ID to a relying party can be verified by locating its most recent owner from the ledger. If that owner is trusted, a verifier can challenge the device to quote its identity or derive a new attestation key. The device should be capable of supporting the change in ownership and be verifiable to its new owner.

3 RESOURCE CONSIDERATIONS

3.1 Cryptographic Services

As MARS seeks to provide a trusted computing solution for resource constrained devices, MARS must reflect the fact that asymmetric cryptography is virtually non-existent in this space. Asymmetric cryptography, typically RSA and ECC, is universally accepted as the preferred means to support digital signatures and key exchange. However, constrained devices effectively lack the computing power needed to generate keys or to use them in encryption and signing applications. They also typically lack hardware acceleration for asymmetric operations, due in part to RSA hardware implementations being larger than the microprocessor itself.

In this limited trusted computing context, hashing and signing are required for measurement and attestation. Encryption/decryption services may also be needed to support related, highly desirable features. Standard symmetric algorithms are lightweight enough to implement measurement, attestation, and encryption/decryption on even severely resource constrained devices.

3.2 Die Area

Conventional TPMs are essentially specialized firmware running on an isolated microcontroller. However, in a very small embedded device, integrating another microcontroller may be infeasible due in part to die area minimization requirements.

3.3 Non-Volatile Memory

A MARS-enabled device, at a minimum, must support non-volatile memory (NVM) required to retain cryptographic secrets, such as seeds or keys, needed for mandatory features. NVM options may be programmable (such as conventional or embedded flash) or read-only (e.g. fuse, mask, PUF). The decision to adopt a particular type of NVM may be influenced by fab availability, manufacturing cost and die area vs device features. To the extent possible, the choice in type and amount of such memory should be left to the manufacturer.

3.4 Random Number Generator

A good source of entropy and a strong RNG are foundational to reliable cryptography. While common practices require (and MARS is no exception) the generation of a secret seed, that generation may occur on an external provisioner. Other secrets on MARS could be derived from the seed. In both cases, an on-chip RNG is unnecessary. Consequently, an RNG should not be required as it might needlessly increase complexity, area and manufacturing cost. Nevertheless, a manufacturer may include an RNG at their discretion and offer interesting and useful services around it.

3.5 Clock

Readers familiar with TPM attestation structures know that each includes the TPM's clock. Moreover, implementations of the clock require NVM. In keeping with the NVM constraints of 3.3, MARS attestations should not require a clock.

4 IMPLEMENTATION CONSIDERATIONS

This section describes considerations for building a MARS device that satisfy the use cases in light of the resource constraints.

4.1 Provisioned Seed

Though MARS' scope excludes the provisioning of secrets, MARS cannot avoid the necessity of exclusive access to a securely generated secret known as the Provisioned Seed (PS). The PS is analogous to a TPM's Primary Seed that serves as the root of a key hierarchy. PS, which will ultimately be used to derive other keys, must itself be generated as a cryptographic key. During startup, MARS would use the PS and possibly other manufacturer-determined values to derive a Derivation Parent (DP). The DP would be the key used for subsequent derivations to fulfill MARS use cases, including [re]generation of the Attestation Key (AK).

During the provisioning process, a provisioner would generate the AK as a secret key shared between the device and an Endorser. The Endorser [5] would be responsible for verifying attestation signatures. This is detailed more in 4.5.3.

4.2 Lightweight Cryptographic Services

Symmetric cryptography provides a viable, though less flexible, solution for digital signatures. Though key generation is simple, key management is complicated because symmetric keys must be shared. MARS devices would require three cryptographic primitives: hash – for recording device integrity, MAC – for signing attestation claims, and KDF – for deriving keys and other values for other use cases.

Within the TPM, a variety of hash algorithms are supported, including those from SHA-2 and SM3. The TPM supports a variety of MACs, but for signing attestation claims and verifying signatures, the only symmetric algorithm supported is HMAC (over the available hash algorithms). Three KDFs are supported, including SP800-108 Counter Mode (based on HMAC).

While these algorithms may be reasonable to implement on resource constrained devices, there are efforts to find lighter weight alternatives. While TCG develops MARS, NIST is conducting a Lightweight Cryptography Standardization Process [6]. The scope of this process is to produce "Authenticated Encryption with Associated Data (AEAD), with optional hashing." The tag produced from AEAD is a type of a MAC. NIST expects that their process will yield a single primitive that can be used for encryption, hashing, message authentication, key derivation, DRBG, etc. MARS should be crypto agile and incorporate these new NIST recommendations where applicable.

4.3 Die Area

To reduce the impact on microcontroller manufacturers, a MARS device should be capable of being integrated without the use of an additional microprocessor. That is, MARS may be implemented as a hardware state machine with few, but valuable, capabilities. Manufacturers that are already integrating cryptographic accelerators should find that adding a small state machine around accelerators designed specifically for lightweight environments will impose little overhead.

4.4 Anti-Counterfeit (Device Identity)

Proof of the association between an identifier and a device is typically done via a cryptographic binding using a secret managed by a trusted agent (the TPM's Root of Trust for Reporting). Verification of that proof may determine ownership, that the device is genuine, or some other relationship based on a trusted identity. In the case of more capable systems that can perform asymmetric cryptography, a certificate can be used to claim an identity. The public portion of an attestation key is used as an identity. A private key, paired with the public key signed in that certificate, is used to prove that identity.

In resource constrained devices, asymmetric cryptography is either impossible or impractical. Symmetric cryptography should be used. In this case, the trusted agent (MARS) possesses a secret shared between itself and a verifier. The secret and the identity are created together – one possibly derived from the other, or the identity is simply assigned. A signature produced via a shared secret must be verified by the verifier who possesses that shared secret for that identity.

The association between a device's identity and its signing key is maintained by the verifier. If needed, possibly to protect against corruption or deletion, MARS can generate a value suitable as an identity via a KDF.

4.5 Boot Health for Access Control Decisions

4.5.1 Measuring Device Integrity

A Root of Trust for Measurement (RTM) initiates the creation of the first digest, delivers that digest to MARS, and optionally populates the Event Log. Though RTM is not part of MARS, both are required to build a trusted platform.

4.5.2 Recording Device Integrity

The Root of Trust for Storage (RTS) is the TCG's solution to recording device integrity. The RTS maintains a set of Platform Configuration Registers (PCRs) that represent a collection of events, and are used as integrity checks of subsets of the Event Log. The PCRs and Event Log are updated immediately after a module is measured and before it is used. The first digest is produced and delivered to the MARS RTS by the host's RTM. MARS allows for resource constrained devices to implement only a single PCR. More PCRs are allowed and useful (see section 4.7.1).

4.5.3 Reporting Device Integrity

Reporting device integrity can be accomplished by relaying the device identity, PCRs and Event Log with integrity and freshness proved by a digital signature. Digitally signing a claim is the role designed by the TCG for the Root of Trust for Reporting. The RTR signs a digest of the PCR(s) and a challenge nonce using an appropriate asymmetric private key or symmetric shared secret.

With asymmetric attestation, the signature is accompanied by an attestation certificate that itself contains a signature of the signing key. Assuming that the relying party already trusts a certificate authority that chains to the attestation certificate, the signature on the integrity claims can be trusted and the claims themselves are considered accurate.

With symmetric attestation, there is no public key to be signed in an attestation certificate. Instead, the claim includes the location of an Endorser who, given the device identity, can retrieve the shared signing secret to verify the signature. The verifier's determination can be trusted by the recipient over an asymmetrically negotiated channel, assuming that the recipient trusts the verifier. Then, the process of assessing the claims can begin.

As the PS is being installed on a MARS device, the provisioner may derive the attestation key (AK) used for signing attestations and provide the AK to the Endorser. During a quote operation, the AK could be recomputed in MARS via the KDF, and used immediately to sign a challenge nonce via the MAC.

4.6 Deep Quote

An application, such as one described in section 2.5, would have a certificate that certifies its signing key. The certificate could also indicate that the key can only be unsealed on a specific platform in a specific state, and that this process is backed by an active RoT that can be queried for proof.

The quoted PCR and Event Log should be examined to show that the device was in the correct state to allow the key to be unsealed. If the key and certificate had been leaked and used on a different device, that device would be unable to perform a deep quote correctly.

4.7 Bound Derivations

The following use cases can be implemented using a secure, deterministic mechanism that derives bits from the PS, PCR and some context from the root of trust and the caller. The result of such a derivation is cryptographically bound to the inputs provided, and is referred to as a “bound derivation.” Methods to do so will be available in MARS to support other mandatory algorithms, e.g. to derive keys (KDF) and quote PCRs (MAC). MARS would provide an API giving secure access to KDF and/or MAC to support a caller’s “derive” needs. Repurposing these functions for other use cases fits within the MARS scope (see section 1.2) as the extra logic required would add a small and acceptable amount of complexity.

4.7.1 Sealed Storage

The TPM natively supports up to 128 bytes of sealed storage. For larger amounts of sealed storage, the same service could be used to hold a cipher key. That key could then be used to encrypt/decrypt much larger amounts of external data. Instead of supporting two mechanisms for sealed storage, MARS could employ the latter because that can handle large and small amounts of data.

Instead of using the TPM to store and unseal data or a key, the KDF mechanism can be used directly to generate a key. The label input to the KDF would indicate the type of key to create, e.g. “seal” or “key1”, etc. If the device state unexpectedly changes because either the configuration of the device changed, or the data was moved to a different device, a useless key would be generated and unsealing would not be possible.

For example, a MARS equipped host could seal data for a firmware TPM (fTPM). The key would be sealed (bound) to the host device and the host’s state. This key could then be used to by the fTPM to encrypt/decrypt its state

4.7.2 DICE Emulation

The requirement from DICE that a CDI be derived from a device secret and the digest of the First Mutable Code (FMC) can also be handle by a KDF function. Since the PCR would be a reflection of the FMC’s digest once extended, a KDF that incorporates the PCR (i.e. a bound derivation) would generate a compliant CDI. Once generated, the PCR should be capped so that the CDI can no longer be generated. After capping, then FMC can be executed.

4.7.3 Locking external peripherals to a device

MARS can be used to create a bound derivation that excludes non-deterministic PCRs. That can be accomplished by using PCRs with a fixed value (e.g. zero, before extended) or by using no PCRs in the bound derivation / KDF. A label input to the KDF representing the peripheral in question would be reasonable. The resulting value can be used as the key to the peripheral's lock.

4.7.4 Locking external peripherals to a device state

This use case is much like the previous (4.7.3) except that extended PCRs are used in the KDF when creating the lock's key. This key would be bound to the device and its state.

4.8 Setting a GPIO, based on device state

A GPIO subsystem can be configured to react when a specific access value is provided. When that GPIO subsystem is integrated with MARS, that access value could be linked exclusively to the PCR. Access to the GPIO only succeeds when the device is in the correct state.

4.9 Identifying how long a system has been booted

The TPM records *time* in a 64-bit count of elapsed milliseconds since being powered on. This value is included in every attestation structure signed by the TPM (in addition to the clock, see 3.5). In a lightweight implementation, this could instead be achieved by dedicating one of the "PCRs" to contain time. Obviously, the PCR would not be extendable, but it could be MAC'd.

4.10 Securely Proving Knowledge of a Password

The shadow password table, shown in 2.13, could be implemented with a MAC instead of hash, as in:

Userid, MAC_Algorithm (using MARS resident key), 1, salt, MAC_{key}(salt || password)

This could counter an offline dictionary attack as the MARS key is unavailable to an attacker.

4.11 Verified Boot

Assuming that the verifier possesses a known good measurement that matches the event being attested, the verifier can sign the measurement using an appropriate symmetric algorithm and signing key shared with the device. The signature is then given to the device. On subsequent boots, the signature is used locally to verify the image prior to its usage.

4.12 Verified Update

This use case reuses the verifier logic from Verified Boot, but prior to the update being applied.

4.13 Protected Update

A MARS-enabled device could support the “Restrict updates to minimal FUE” use case via the GPIO mechanism. Refer to [4].

4.14 Chain of Custody

The AK shared between the current owner and the device can be derived from the set of previous owners’ device IDs on the device, and independently derived by owners. The most recent previous owner essentially becomes the AK provisioner for the next owner by sharing updating the ledger. A device that can produce the correct attestation key from the (optionally cached) ledger can be verified by its owner and confirm the ledger.

5 ACRONYMS

The table below defines acronyms used in this document. See the TCG Glossary [7] for definitions of TCG terms.

AEAD	Authenticated Encryption with Associated Data
AK	Attestation Key
CDI	Compound Device Identifier
DICE	Device Identifier Composition Engine
DRBG	Deterministic Random Bit Generator
ECC	Elliptic Curve Cryptography
FUE	First Update Engine
GPIO	General Purpose Input Output
HMAC	Hash-based Message Authentication Code
ID	Identity
KDF	Key Derivation Function
M&A	Measurement and Attestation
MAC	Message Authentication Code
MARS	Measurement and Attestation RootS
NVM	Non-Volatile Memory
PCR	Platform Configuration Register
PS	Provisioned Seed
PUF	Physical Unclonable Function
RoT	Root of Trust
RTM	Root of Trust for Measurement
RTR	Root of Trust for Reporting
RTS	Root of Trust for Storage
SD	Secure Digital
TPM	Trusted Platform Module

6 BIBLIOGRAPHY

[1] Trusted Computing Group, "TPM 2.0 Library Specification," 8 Nov 2019. [Online]. Available: https://trustedcomputinggroup.org/resource/tpm-library-specification/.

[2] Trusted Computing Group, "Hardware Requirements for a Device Identifier Composition Engine," 22 Mar 2018. [Online]. Available: https://trustedcomputinggroup.org/resource/hardware-requirements-for-a-device-identifier-composition-engine/.

[3] Trusted Computing Group, "Virtualized Trusted Platform Architecture Specification," 27 Sep 2011. [Online]. Available: https://trustedcomputinggroup.org/resource/virtualized-trusted-platform-architecture-specification/.

[4] Trusted Computing Group, "TCG Guidance for Secure Update of Software and Firmware on Embedded Systems," 10 Feb 2020. [Online]. Available: https://trustedcomputinggroup.org/resource/tcg-guidance-for-secure-update-of-software-and-firmware-on-embedded-systems/.

[5] Internet Engineering Task Force, "Remote Attestation Procedures Architecture, Draft v04," 21 May 2020. [Online]. Available: https://tools.ietf.org/html/draft-ietf-rats-architecture-04.

[6] NIST, "Lightweight Cryptography," [Online]. Available: https://csrc.nist.gov/projects/lightweight-cryptography.

[7] Trusted Computing Group, "TCG Glossary," [Online]. Available: https://trustedcomputinggroup.org/resource/tcg-glossary/.

