Challenge-Response Based Remote Attestation with TPM 2.0

by

Huzaifa Hashim

Senior Year Project in Computer Science

Submission: January 15, 2021 Supervisor: Prof. Jurgen Schonwalder

Jacobs University Bremen | Department of Computer Science and Electrical Engineering

English: Declaration of Authorship

I hereby declare that the thesis submitted was created and written solely by myself without any external support. Any sources, direct or indirect, are marked as such. I am aware of the fact that the contents of the thesis in digital form may be revised with regard to usage of unauthorized aid as well as whether the whole or parts of it may be identified as plagiarism. I do agree my work to be entered into a database for it to be compared with existing sources, where it will remain in order to enable further comparisons with future theses. This does not grant any rights of reproduction and usage, however.

This document was neither presented to any other examination board nor has it been published.

German: Erklärung der Autorenschaft (Urheberschaft)

Ich erkläre hiermit, dass die vorliegende Arbeit ohne fremde Hilfe ausschließlich von mir erstellt und geschrieben worden ist. Jedwede verwendeten Quellen, direkter oder indirekter Art, sind als solche kenntlich gemacht worden. Mir ist die Tatsache bewusst, dass der Inhalt der Thesis in digitaler Form geprüft werden kann im Hinblick darauf, ob es sich ganz oder in Teilen um ein Plagiat handelt. Ich bin damit einverstanden, dass meine Arbeit in einer Datenbank eingegeben werden kann, um mit bereits bestehenden Quellen verglichen zu werden und dort auch verbleibt, um mit zukünftigen Arbeiten verglichen werden zu können. Dies berechtigt jedoch nicht zur Verwendung oder Vervielfältigung.

Diese Arbeit wurde noch keiner anderen Prüfungsbehörde vorgelegt noch wurde sie bisher veröffentlicht.

Huzaifa Hashim
January 15th, 2021.

Abstract

The advent of technology has brought a considerable amount of good to the world and has changed the way humans interact and work. That has meant that the reliance of humans on systems and platforms has increased. From personal computers, to smartphones, and now with the advent of the Internet of Things, automated systems have surrounded the world for all sorts of mundane tasks. With the proliferation of devices that humans possess, it has made become essential to ensure reliability and trust in the security of these devices.

For ensuring the safety of the systems that people use, it becomes important to verify the correctness of software and hardware that runs on a computer beyond proofs of correctness. What the world of computing is moving towards is a real-time solution to knowing what systems to trust. To achieve that, the goal of Trusted Computing has been to find solutions which could potentially isolate systems that have been compromised from those which have not. The Trusted Platform Module is one such possible solution which could barely hold a fraction of the data that humans interact with on a daily, however, could potentially be a milestone in making sure that huge amounts of data is never compromised.

Contents

1 Introduction

The proliferation of networked systems has made it essential to ensure reliable trust in devices connected over a network before any exchange of information can occur. Remote Attestation is the process of verifying the integrity of a host system before it is allowed access to a network by exchanging information on the device state with a trusted third-party. The principal goal of the scheme is to build trust in the configuration state of a system. This makes it possible to ascertain if a host is in a known state, has not been compromised, and can safely be given access to resources without the possibility of any deviations or misbehaviour. These measures have been made possible through work on Trusted Computing which has greatly seen an increase in research and adaptation by device manufacturers in recent years.

Throughout this paper, we look at Remote Attestation which is the process of attesting to a remote machine over a network. Local Attestation is a rather different problem with the same goal: the ability to determine if a machine is in a trusted state would require the certainty that the machine or the user locally possesses the ability to certify the ”correct” state and is unable to modify it.

Intuitively, a system cannot be trusted to report on itself and provide values which represent the status quo of the machine state. Thus, there is a need for an external infrastructure to ensure the integrity of the information that is shared between the trusted and non-trusted parties in a Remote Attestation scheme. To make this possible, the Trusted Computing Group has been working on tamper-resistant hardware that is soldered onto almost all devices that are shipped today. The goal of this microprocessor of sorts is to assist its host device to partake in an attestation scheme which produces attestation results. These results are then compared to verifiable data from which decisions on whether a device is in a trustworthy state can be extrapolated.

To this end, almost every personal computer sold now has a Trusted Platform Module which enables the device to accurately identity itself through a set of cryptographic keys and calculate hashes over its own boot process and store them for verification. This enables a trusted verifier to initially corroborate device identity, and more importantly, the device state. This data can then be compared to known good values of the digests over the boot process, and in the instance of a mismatch, can be used to effectively initiate countermeasures by either removing a device from the network or initiating a reset. This makes it possible to corroborate the state of remote devices through measures which are based on secure cryptographic primitives, and hardware support which is minimalist in its computational power, but thoroughly effective at its intended purpose. Even though the information on the current state of a device may not be seem a sufficient metric for trustworthiness in isolation, it is much more effective to build decisions from it.

This project aims to analyze the remote attestation scheme with a Trusted Platform Module (TPM) on a PC platform. In section 2, this paper analyzes Remote Attestation and the fundamental notions around it by codifying the terminology which will be used in the rest of the paper. The discussion then moves onto the TPM itself in section 3, where the design and taxonomy of the device is analyzed. This is also the part of the paper which will delve into ideas of trust. In section 4, the discussion shifts to an implementation of the procedure that is a work in progress by the IETF Working Group on Remote Attestation [], with an acute focus on the current internet drafts that aim for standardizing the formats of claims and assertions on system components.

In section 5, the paper looks into the libraries from the TCG TSS2 specification which have made it possible to interact with the TPM on PC platforms. This work has helped the community around Trusted Computing immensely since all implementations stem from the API's that allow access to the resources to initiate a remote attestation scheme. This section looks into a bare bones implementation of Remote Attestation written in scripts by the open-source community.

At the end, this paper analyzes the way forward by looking into the development of the drafts which aim to standardize operations, and simplify the interactions with a TPM for programmers. The ultimate goal of the community around the TPM, and Remote Attestation is to increase its usability by flattening the learning curve, and providing access to resources which can be utilized and adapted as the norm in Trusted Computing.

2 Remote Attestation

In a typical attestation mechanism a *Prover (P)* and the *Verifier (V)* exchange information to qualify the trust that should be attributed to a remote system. This procedure follows a challenge-response protocol in which information shared between the parties must provide a meaningful measure of the memory contents of an untrusted device[1]. A compromised device will produce invalid responses and can be restricted in its access or be removed from the network altogether.

This high-level description conceptualizes the problem fairly well. Over any network, knowing the characteristics of a peer before connection can be useful. Incoming network packets that can be analyzed on the wire provide insight into data shared and TLS Client Certificates can be used to establish authentication, however, there is no alternative insight into the device state to ensure that the endpoint is not occupied by an attacker.

Moreover, relying on a device to provide evidence about its own state is not any better since any malicious attacker can report forged evidence under the pretense of reliability. Thus, for this procedure to be authentic there is a need for a trusted party that assists in the sharing of information, can vouch for the authenticity of data reported for verification and can corroborate the device identity to ensure that the device is the one that it claims to be. Currently, the Trusted Platform Module is the piece of hardware that ensures this for most platforms.

According to the taxonomy presented by Steiner and Lupu[1] there are three main types of remote attestation:

1. *software-based attestation* which operates with an attestation scheme closely tied to timing constraints to ensure authenticity and freshness of the data transmitted by an untrusted device.

2. *hardware-based attestation* which relies on a tamper-resistant piece of hardware to assist in attestation. This piece of hardware is argued to have a trusted execution environment, which executes instructions in a secure manner already hard-wired in the design specification.

3. *hybrid attestation* is the composite of both techniques. This employs the use of hardware already present on most devices, such as ROM, as well as the security that comes with encryption in modifiable software to execute attestation.

In the study conducted by Steiner and Lupu[1] on Remote Attestation in Wireless Sensor Networks, the attestation procedures in discussion are constrained by the limiting computational complexity of the nodes. However, the primitives around attestation schemes remain relatively the same over platforms. For a successful attestation procedure, there are two critical parts which must hold true[2]:

1. An Attester must be able to provide proof of its identity that is tied to its hardware according to the standards defined in IEEE 802.1AR[3].
2. An Attester must be able to provide a reasonable measure of the process through which the information shared was generated and the associated methods to protect it against tampering.

2.1 Terminology

Even though the procedures involving Remote Attestation directly involve an untrusted attester, and a trusted verifier that checks a device against information that is provided, even a bare-bone implementation requires a great deal of entities and artifacts to perform the operations necessary. This section introduces key roles of the procedure and lists important entities for semantic consistency throughout the research project which are referred from the IETF's Remote Attestation Architecture[4].

- An Attester is a device whose evidence must be appraised to infer trustworthiness in the device.
- Appraisal Policies for Evidence are a set of rules against which the information provided by the Attester is checked for validity.
- Appraisal Policies for Attestation Results are the set of rules that a Relying Party uses to analyze the Attestation Results produced by Verifier.
- A Verifier appraises the validity of the evidence generated by an attester as defined by the Appraisal Policy. The verifier produces Attestation Results.
- A Verifier Owner initializes the Verifier with the Appraisal Policies that enables it to act as a trusted entity in the Attestation Process.
- Evidence is the information generated by an Attester which is to appraised by a Verifier. This could include configuration states and measurements across critical components of the system.
- Claims are asserted information that build up the structure of the Evidence that is generated by an Attester.
- Reference Values are generated and provided by manufacturers to Verifiers which are used to compare the data generated by an Attester. These are also known as Golden Measurements or known-good values.
- Reference Value Provider is an entity which has the authority to declare values which determine that a system is in a known good state. This is normally the manufacturer.
- Endorsements are statements which prove the ability of an Attester to generate information over its memory contents and cryptographically sign the information that is relayed through an Attestation Procedure.

- An Endorser provides Endorsements which help Verifiers ascertain the ability of the Attester to generate the evidence that is produced.
- The Relying Party acts on Attestation Results produced by Verifiers and depends on the validity of the information provided by an Attester.
- Relying Party Owner configures the Relying Party to consume Attestation Results and apply the Appraisal Policies. This is normally an administrator in a network.

To encapsulate the mentioned terminology, it helps to visually imagine the linkages throughout the chain of entities that interact with each other during the process.

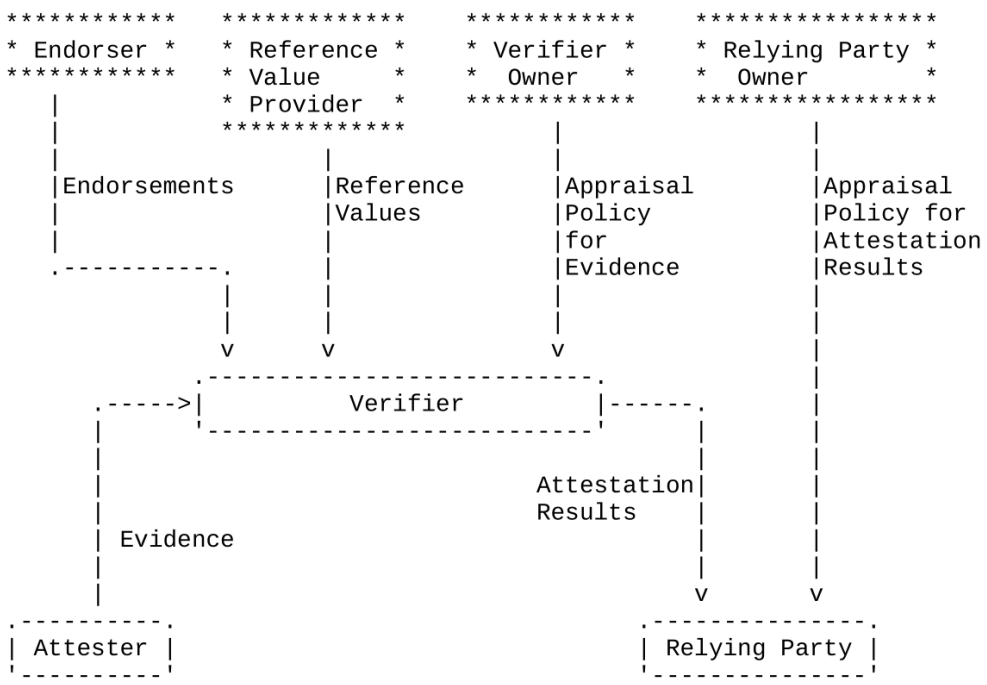

Figure 1: Conceptual Data Flow[5]

Apart from providing a sense of clarity, the diagram from the Architectural Design provisioned by the IETF Working Group demarcates the entities which are involved in an Attestation Scheme. Variants have been implemented by different groups, however, the underlying structures remain the same. An Attester creates Evidence, the Verifier consumes this Evidence and refers to its configuration and computes a result which can be used to provide some form of output which determines the state of the Verifier. The Relying Party may be a separate entity, and may even be coupled together with the Verifier, and perform its tasks in unison with a Verifier. The nuances around Attestation schemes is what this paper aims to investigate, using resources that are work in progress documents and code bases that are still being improved in real-time.

3 Trusted Platform Module 2.0

Back in the 1980's, the US Department of Defense in their "Orange Book" defined the Trusted Computing Base (TCB) as the block of software, firmware, and hardware that you have to rely on for enforcing a computer security policy[6]. Any failures on the integrity

of these parts in a system would inevitably lead to undesirable consequences and would compromise the security of the device.

Since the security of a system is synonymous with the trust that can be attributed to its components, it is important to verify the components of a device. This helps to conclusively judge the identity and configuration state of a device before it is allowed access to data or a shared network, or even used on its own. For PC platforms, attestation schemes could well enough be a measure of making sure that the boot process is authentic, and the user is in complete control of the system.

The TCG defines trust in a platform as the expectation that a device will behave in a particular expected manner for specific purposes[7]. Furthermore, a platform with established trust should be able to provide access to secure storage, record integrity measurements, and report said measurements. Since a system cannot be trusted on its own, there is a need for a tamper-resistant component which assists a device to prove whether it can be trusted and can itself be trusted to not be compromised at any given time.

The Trusted Platform Module (TPM) acts as a secure processor capable of performing cryptographic operations such as the ability to sign with a key and provide secure storage for the purposes of establishing trust.

Moreover, to provide data on the configurations of a device, measurements of the boot process must be stored in secure locations. These measurements are a hash calculated over the entirety of the memory contents of the firmware and bootloader which are extended into the secure storage on a TPM.

To ensure the functionality that is required, along with the required level of security, the TPM is designed as a very simplistic and cheap device. It does not operate with the same read/write capability of memory registers and offers a small but non-volatile memory storage and can only be accessed by predefined interfaces. It sits on a low-bandwidth bus which protects its contents from side-channel attacks and offers a considerably reliable storage of platform-specific keys. It is this requirement which bounds Trusted Computing to hardware solutions, since other functionality of the TPM is easily implementable in software alone.

The TPM operates with prescribed commands that perform prescribed actions on the data that is held on the TPM. A proper implementation for the Roots of Trust, that is defined by the TCG, is crucial for the TPM to perform its expected tasks. However, the TPM is not a trusted computing base in a system. It acts as a component which helps a trusted entity to determine if the TCB is compromised, and can potentially prevent the system from a startup if the TCB is not ”instantiated”[7].

3.1 Concept of Trust

Understanding the chain of trust is imperative because computing platforms build on transitive trust. By establishing trust in an immutable component, the following software and firmware are extended into the trust chain by calculating a hash and extending the hash into TPM storage. The TCG defines multiple components associated with the boot process of a system which depends on the implementation of the Roots of Trust.

Building trust chains from one piece of software to another is only reliable if the root of the chain is certifiably reliable. Since this part of the chain forms the TCB, it is neither possible to check nor verify that the initial component from which the chain of trust originates from

is in a known ”good” state. This is because if the root of trust chain can be measured and checked, there has to be system component that sits on a lower layer to measure and check its validity. Moreover, trusting the initial component is only good enough if the trust can be extended to the range of components that determine the state of a machine. The trust chain allows us to bootstrap from a root of trust to a higher-level component and establish trust in secondary and tertiary components and so on and so forth[8].

Roots of Trust are the initial components in the trust ecosystem. In a PC platform, we normally have the following Roots of Trust:

- Roots of Trust for Storage (RTS): This component provides a shielded location to store data that should only have qualified access, such as a private key. Since the TPM memory cannot be accessed by anything other than itself, it acts as a RTS in a trusted platform. The TPM can also layer the shielding around data by allowing access to a shielded location only if access to another shielded location has been granted.
- Root of Trust for Reporting (RTR): This component is responsible for accurate reporting on the data that it holds. According to the TCG, the data reported by the RTR is a digitally signed digest of selected values within a TPM[7]. The RTR is binded to the TPM by means of a Platform Certificate. This is to make sure that the values reported accurately represent the state of the platform[7].
- Root of Trust for Measurement (RTM): The RTM is responsible for measuring pieces of software in a device and storing them in a secure location. In the PC platform, the RTM is normally tied to the boot process, where the CPU measures the initial measurements marking the start of the chain of trust.

The Static Root of Trust for Measurement (SRTM) initiates the measurement of the state of the hardware and software environment in a platform. The (SRTM) is a block of code rooted in the BIOS/UEFI and does not exist on the TPM. It is responsible for establishing the starting point of the measurements that are eventually extended and stored in a configuration register. It is also referred to the Core Root of Trust for Measurement (CRTM) and is responsible for bootstrapping the process of building a chain of measurement run up until the bootloader[8].

The SRTM is the default RTM for all components because the SRTM measurements occur during boot by default which can include the firmware and the software that is executed on the computer system[9]. The implementation is heavily dependent on the host system BIOS. On Windows systems, the bootloader may also extend a measurement of the Operating System Kernel, whereas on Linux, Grub or ”tboot” can be installed which measures the kernel upon start up. Other mechanisms, like the Linux IMA extends these measurements into run-time attestation capabilities by extending a measurement of application binaries before they are launched and extending them into a TPM[8]. The control from the SRTM is relinquished only after the executing entity has measured and stored the data on the entity that it transfers control to.

3.2 Platform Configuration Registers

The TPM uses the Platform Configuration Registers (PCRs) to store the measurements that are pivotal to the attestation procedure. In the TPM 2.0 configuration, the PCRs are of variable length and use the SHA-256 hashing algorithm for the extension of the mea-

surements taken over the contents of memory on a system component on the platform. The implementation is different compared to normal registers, in that, they do not operate with the same read/write calls. Instead, the only way to store data into a TPM PCR is by extending the contents of the PCR which is done by concatenating with the existing value, and then passing it through a hash algorithm. This ensures that the resulting value stored is representative of all measurements taken, and the previous values held in the registers. This prevents the possibility of an attacker forging representative values in the PCRs, even with access to known good measurements for the software, firmware, and the boot process in general.

The PCR operations that are most relevant to discussions surrounding attestation are the extend (`TPM_PCR_Extend`) and reset (`TPM2_PCR_Reset`). The extend operation takes a value, and an index, concatenates the new value with the existing data in the register at the given index value, hashes the result of the concatenation, and stores this value in the register. The reset operation, which is triggered automatically upon the first boot and the invocation of the DRTM.

In theory, all of the values could be hashed into one PCR. This would establish all integrity measurements across the boot process and inextricably link them to each other. However, for verification purposes, there has to be a way to recalculate the hashes that are stored into PCRs and to record all measurements that are extended into a single PCR would just not scale very well.

Secondly, within a platform, there are measurements that rarely change compared to measurements that change on a regular basis because of updates and patches shipped to software components. In a measurement chain that goes up to the application layer, it would become increasingly difficult to keep track of changes that are predictable and expected, compared to changes that would require action[7].

The PCRs on a TPM 2.0 introduce the idea of PCR banks. These are grouped together and implement the same hash algorithm which improves the configuration of PCRs in terms of the access and authorization policy implemented. The possibility to dynamically allocate PCRs also exists in the TPM 2.0 specification which lay outside the non-volatile memory of the TPM. These can be used by a user of the system for authentication purposes, however it does not have a lot of use cases for Remote Attestation.

3.3 Keys and Measurements

The ability of the Trusted Platform Module to successfully execute operations surrounding Remote Attestation relies on its secure keys. Since the TPM has measures to ensure that keys are not compromised throughout its lifetime, a significant amount of trust can be attributed to the TPM's cryptographic ability to tie a key with a device identity, and ensure that a compromised endpoint cannot permeate through.

The TCG and its current draft on TPM 2.0 keys define a key as the public part of an asymmetric key pair, a certificate as a X.509 certificate, and a credential as the private part of an asymmetric key pair along with the certificate which binds the key to its public part. In the case of a credential, a private key in isolation cannot be referred to as a credential, and neither can a certificate[10].

Moreover, besides the keys that are used for the purposes of Attestation, the Device Identity credentials are inherent to the security of the TPM. These credential remain the same

throughout the lifespan of a TPM, and are used for authentication over a network[10] to prove to a relying party of device identity.

The IEEE 802.1AR[3] which sets the standards on device identity establishes that an initial device identity certificate must be present, however, local device identity certificates can be created if needed and bounded to the device.

To move past the TPM 1.2 specification which mainly used a set of constrained keys, the TPM 2.0 defines a hierarchy within its keying structure. The taxonomy defines the following control domains, each with its own set of keys along with the ability to generate them[10]:

- The Endorsement Hierarchy is responsible for storing objects that are sensitive and key to establishing the TPM's authenticity, and consequently, the authenticity of the data that is produced and shared by the TPM. The Endorsement Hierarchy contains the Endorsement Primary Seed which is passed onto a key derivation function `TPM2_CreatePrimary` which outputs an Endorsement Key[11].
- The Platform Hierarchy is designed to be under the control of the platform manufacturer which can only be accessed by a platformAuth value. This is to ensure the integrity of the firmware (BIOS/UEFI).
- Storage Hierarchy is a user process such that applications running on the platform have access to this hierarchy for the protection of user data. In the TPM 1.2 specification, this was the only hierarchy present within the TPM.

3.3.1 The Endorsement Key

The Endorsement Key and its associated certificates grants authenticity that the PCR values a verifier receives are from a TPM. The EK for TPM 2.0 is an asymmetric key pair, and can either be an RSA or ECC key. The private part of the EK is never revealed to any party outside the TPM, and the key-pair is not used for signing. The EK is purely defined to be a decryption ”Storage” key[10].

The primary use-case for the Endorsement Key is for Attestation purposes. Since the EK cannot be used to sign data that is stored in the PCR, there is a need for a signing key. However, tying signing keys to a TPM like an EK does not scale very well. To assist Attestation, signing keys are tied instead to the EK and the EK credentials are then used to provide evidence that the Attestation Signing Keys originate from the same TPM that has a specific Endorsement Key. This ties the data to a TPM, which is tied in turn to a specific device which can be identified, streamlining the process of tying the evidence that a verifier receives to data signed and stored by a TPM installed on a device that it was intended for. This complex structure is ensured through the security primitives over the Endorsement Key

3.3.2 Attestation Keys

Attestation Keys are keys which are used to sign the digests stored in the PCRs. However, the AK is restricted to sign data that is only produced by the TPM. This ensures that when a digest is signed by a restricted signing key, it is undoubtedly produced within the TPM ecosystem. An unrestricted signing key is only good enough for authentication purposes.

The Attestation Key certificate can be instantiated by the device manufacturer, or created through user processes locally as well.

4 Challenge-Response Based Remote Attestation

The standardization work currently in progress by the IETF and the TCG on Remote Attestation is focused on multiple areas of improvement, including, but not limited to, interoperability, standardization of claims shared between devices, and the protocols involved in conveying evidence to relying parties. Since the work is currently in progress, it is important to mention that this project will look at documents that are in the process of being improved, and the following information may need revision in the near future.

4.1 Remote Attestation Architecture

The architecture for Remote Attestation follows a normative description that is not bounded by a specific protocols or use cases. It allows for mapping a flexible architecture that allows the inclusion of ”various existing and emerging Remote Attestation Procedures[4].”

Once an Attester produces evidence and sends it to a Verifier, the Verifier uses an Appraisal Policy and Endorsements to asses the data and produce Attestation Results. The results are then relayed to a Relying Party which uses its own Appraisal Policy to make specific decisions regarding the attesting device.

An Appraisal Policy that is used by both parties to assess the evidence may be a set of reference values which are known to be trustworthy. The evidence generated by the Attester is compared against these values for consistency. Thus, any alterations to the memory of an untrusted device would indicate an unreliable state. However, this assumes the existence of a set of known reference values that are communicated to the parties that are responsible for drawing up comparisons. As such, a reference value provider is crucial to provide the data which equips a Relying Party and a Verifier to execute their respective roles.

Endorsements and Reference Values are assumed to be provided by the manufacturer of a device and should be accessible to a Verifier and the Relying Party. The data format and specifications are platform and protocol specific, however, work has and is being put into increasing the interoperability of the data that is made available to components responsible for corroborating the evidence generated by an Attester.

To this end, the TCG has defined a Trusted Attestation Protocol (TAP) which defines the information used by Verifiers in the attestation process and Reference Integrity Manifest (RIM) structures which a Verifier uses to validate expected values provided by an attester against actual values provided by the manufacturer of the attester.

The Trusted Computing Group’s Provisioning requirements for the TPM 2.0 describe a set of Golden Measurements. These are a set of measurements that reflect the expected default values of the integrity measurements that the boot firmware and the subsequent code generates and extends into the Platform Configuration Registers in a TPM. The TCG requires that Platform Manufacturers provide measurements across the BIOS, firmware, and other binaries, which are also included in boot firmware updates[12].

Understanding an Attester’s environment is crucial to moving ahead with Attestation. We have assumed that the Attester takes measurements, and passes it to secure storage in

a TPM, however, there are structures which need a better understanding to proceed with the Remote Attestation model. The measurements over code, registers, and memory are done on the Target Environment of an Attester. This is carried out by a nested Attesting Environment which is responsible for the formatting and transmission of the data that is collected.

This creates a layered attestation environment in the platforms that wish to be verified. The component which remains essential is the root of trust from which this chain of measurements begins and builds transitive trust through an Attester's nested environment. In essence, this compounds the idea of transitive trust by ensuring that each environment measures the next before handing off control to the following component. The following diagram shows a model through which a Layered Attester first measures its succeeding environment, ensures that the measurements are stored in storage where the following entity cannot alter measurements once it is given control, and then hands off control to the Target Environment B which passes on values collected on itself and previously held values to the Verifier[4].

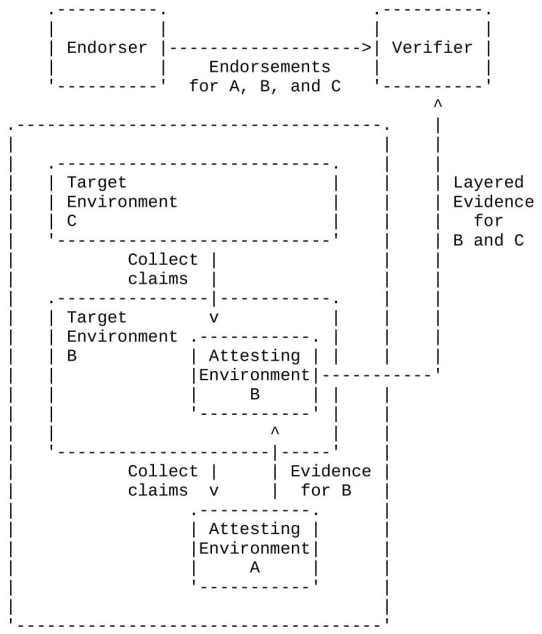

Figure 2: Layered Attestation[5]

A more specific primitive which needs to be established for Attestation is the need to ensure that the evidence generated and being consumed by a Verifier represents the actual state of the device as it is, and does not represent a snapshot of a known good state right before a malicious attacker took control of the device. Intuitively, this can be done by ensuring that the information that is carried through to the Verifier carries with it a timestamp which identifies the exact moment in time when the measurements were taken. Ensuring this property proves the "Freshness" of the evidence that is generated, however, it is much more complex than having a timestamp since that would require additional claims about a device's internal clock mechanisms.

On the contrary, if the responsibility of determining whether the information generated by an Attester is fresh is shifted to the Verifier instead of an Attester, it would be much easier

to ensure that any clock mechanisms on the Verifier is trustworthy since the Verifier is assumed to be in a trustworthy state.

In the approach recommended by the TCG, however, timestamps are foregone and instead an unpredictable nonce is transmitted by the Verifier which is then included in the data that is sent back by the Attester with a signature from the Attestation Key. If the nonce is the same that was transmitted, it ensures that the signature over the claims generated by an Attester was done after receiving the nonce. The IETF Working Group on RATs regard this as a rough guarantee of freshness since it uniformly applies across an entire set of evidence, and does not provide an insight into the time between the generation of claims and their respective collection[4].

4.2 Interaction Models

The interaction models serve as a stencil from the IETF for the specification design of Remote Attestation Procedures. This allows a standardized means of communicating evidence that is generated by the Attester to a Verifier which maintains privacy and trustworthiness of the information shared. Whereas the evidence that is generated may be purely judged over by Appraisal Policies that are acutely defined by the TCG, interaction models, like the challenge-response protocol, specifically deals with the conveyance of this evidence between the parties in an attestation procedure[5].

To ensure that evidence is conveyed successfully, the IETF define normative prerequisites which establish a list of supporting mechanisms[5]:

- Attester Identity must unambiguously identify the device that the evidence is generated from, namely the Attester. This can be in form of a signature, or a zero-knowledge proof.
- Attester Evidence must be authentic and must be correct. This is proven by tying it to the identity of the attester by crypotgraphic association through the means of a certificate, or a reference to an identity document.
- An Authentication Secret that proves the authenticity of the evidence claims generated must first be established before any procedure can go ahead.
- Evidence generated must include a means to prove the freshness of data that is shared, and similarly, any interaction model must include means to share this proof with a verifier such that it can be used in the appraisal process.
- And, evidence generated must be protected and formatted by the attester such that it can be conveyed and analyzed by the verifier in a ”tamper-evident” manner.

Furthermore, a Handle is a statement which could be a nonce or a signed timestamp[5] that can be used for proving the freshness of the evidence generated by an Attester. However, different uses of handles may be employed so far as there exists a corresponding understanding between a Verifier and an Attester.

A Challenge-Response Remote Attestation, as defined by the IETF’s Working Group is initiated by a Verifier request. This includes a Handle, a selection of Claims, and a list of Authentication Secrets.

The Handle consists of a cryptographically generated nonce, which guarantees Evidence Freshness. The list of Authentication Secret ID’s selects the Attester Environment, and

thus the key with which the Attester signs the evidence. A selection of claims selects the information which must be transmitted from the PCR's of an Attester's TPM. For a boot integrity verification, the Verifier could request measurements across the BIOS and firmware, and exclude evidence on currently running applications on the Attesting Platform.

Once this data is signed, it is sent over to the Verifier for appraisal. Upon reception of the evidence, the Verifier validates the signature on the data, checks for device identity, confirms that the nonce generated represents the current state of the system, and appraises the claims with use of Reference Integrity Manifests. This produces Attestation Results which can be consumed by Relying Parties to carry out an assessment on the Attester's trustworthiness[5].

The following sequence diagram explains the procedures of this schematic as represented in the Interaction Models produced by the IETF Remote Attestation.

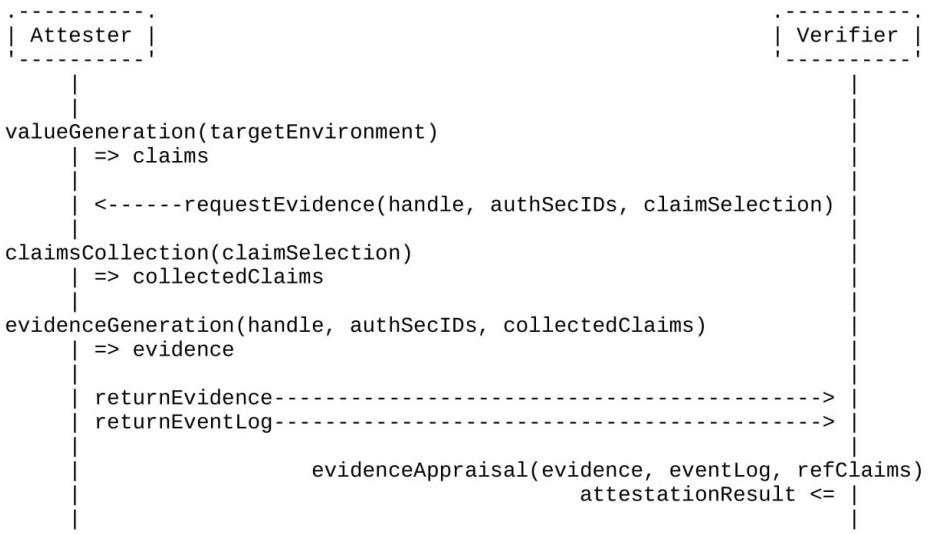

Figure 3: Challenge Response Remote Attestation[5]

4.3 Reference Values

In the process of provisioning a TPM, the design specification by the TCG require the manufacturers describe a set of ”reference measurements”[13]. This represents the default integrity measurements of the boot firmware and subsequent code known as the Golden Measurements. In the attempts to guide manufacturers, the TCG provides an information model using a XML Schema which aims to provide a set of data items that could be used to generate a set of these measurements.

In the following sections, we look into the protocols which the TCG has defined for Verifier's that are responsible for appraisal, and furthermore, the process of generating the values against which a Verifier can compare evidence it receives.

4.3.1 Trusted Attestation Protocol

The Trusted Attestation Protocol (TAP) is intended to specify Information Elements that could possibly be shared between an Attester and a Verifier [12]. Since it is a guide to enterprises in designing the TPM, it is an extensible version of what the TCG believes are best practices in the design of the hardware support for Attestation. However, since Quotes are retrieved from the TPM according to the Information Model defined in TAP, it is important to note that their retrieval be done in a way that is compatible and does not invalidate the signature as specific in[14] and the trust model is preserved.

The information elements defined by the TCG are protocol independent and do not specify a particular method or protocol to deliver elements to the Verifier, however, the TCG does require that each element has a number of basic characteristics[12]:

- The data shared depicts an accurate state and is not a replay of previously created data that may not be relevant anymore. This ensure the ”freshness” of the data shared.
- The data shared provides meaningful information of the current state and can be used by the verifier for the purposes of analyzing the state of the untrusted device.
- The data is bound to one specific device and provides a verifiable proof of device identity.

Each element follows the preceding standards with a means to be identified by the Verifier. The TCG specification presents the elements in type-length-values (TLV) to illustrate the key elements which are assembled into an attestation report received by the Verifier.

The description of the information elements that follow do not include the TLV illustrations, but present a high level description of the elements and associated components in the TPM Architecture Model that allow for the Verifier to unpack and determine the state of a platform.

The nominal method of establishing trust in a key is with an associated certificate, which provides the information on the generation of the key and the protection surrounding it[7]. The Attestation Key Certificate Chain contains the certificate of the Attestation Key (AK) which is used to attest the PCR values and other certificates that the Verifier may use in the process[12]. The X.509 Certificate Chain of an Attestation Key contains the number of certificates that a Verifier can expect to receive, followed by a list of tuples with the byte size and associated certificates.

According to the specification in the TPM architecture, the Attestation Key can only be used if a system is in a particular state. Moreover, a ”Session” is defined in the TPM Architecture as a collection of the state of the TPM which changes as the session progresses[7]. The information element tied to the signing key in the TPM contains the data that is returned by the TPM on the invocation of the `TPM2_Certify` command from the `tpm2_tools`. For the purposes of attesting the signing key, the information element must contain the public part of the signing key, along with the result of the `TPM2_Certify`.

The informational element on the PCR values for the TPM provides the data derived from the execution of the `TPM2_PCR_READ`. This could include the values for the entire PCR's, however, a Verifier could specify the PCR's for the purposes of attestation by sending the information element back with zeros in place. In the case of a hibernating platform, and

a TPM configuration that is concerned with the restoration of the Operating System from non-volatile storage, the Verifier will request all PCR values that have been extended.

The TAP model extends additional information elements for the purposes of implicit and explicit attestation. However, the key takeaway is that a TPM Quote which reads the current values of the PCR registers, couples them together with the Verifier's nonce, and produces a structure that is signed with an Attestation Key that never leaves the TPM is defined according to the standards mentioned in the TAP Information Model. This allows the Verifier to use the Quote and the Event Log to analyze the results and produce effective results on the Attester's reliability.

4.3.2 Reference Integrity Manifests

The Verifier needs access to information to validate a TPM Quote it receives during an Attestation Procedure. The model defined by the TCG contains an abstract representation of elements and their respective attributes. This adds up to form a Reference Integrity Manifest (RIM) which contain assertions about the configuration and state of a particular platform, which in turn contain the values that have been identified by the manufacturer as correct.

The Information Model for RIM defines an abstract representation of a manifest. The TCG defines a Binding Specification which defines the instance of a RIM. This binds the manifest to a set of protocols, formats, and delivery methods used to convey the information to a Verifier. This could potentially define a model for the information manifest to be marshalled for delivery over an IP-based communication protocol or as a collection of files in a file system[13].

Reference Integrity Manifests come in collections of bundles. The Base Manifest in a bundle cryptographically identifies the manufacturer and identifies the supplementary bundles within a collection. The Base Manifest also possesses a unique identifier for a set of bundles and cryptographic hashes for all the information packed into the manifest.

For the purposes of Attestation, Support Reference Integrity Manifests are much more relevant. Whilst the Base Manifest defines the metadata of the information that is made available to a Verifier, the hashes and measures of all information that a Verifier actively uses to corroborate the information it receives is included in the support RIMs [13]. The TCG requires that a specification contain a list of integrity measurements which should be encapsulated within support RIMs:

- A hash which captures the firmware and software components of the boot process,
- a hash over critical event components that a Verifier must evaluate,
- and a hash over configuration items.

The Reference Measurements that are essential for Remote Attestation are as follows:

1. Measurements relating to the system configuration after startup. These include the BIOS, the bootloader, and the Operating System Kernel which effectively exchange control once a system is started in a single-threaded known sequence. Effectively, this means that the PCR values after a sequence has completed would be known by the vendor, and can be provided to the verifier to check for once a part of the boot process has been executed.

2. Once a system is in operation, it gets unpredictable to gauge the PCR values at a given point. Linux defines an Integrity Measurement Architecture (IMA) which does routine measurements in high volumes, and thus PCR values may constantly change. In this scenario, the verifier should have enough credible information to reconstruct the final hash values and signed reference measurements from a trusted authority.

The format of the information is defined by the TCG in the form of Software Identification Tags (SWID) which is defined by the ISO-IEC 19770-2. The IETF Working Group currently has a work-in-progress report on extending the standards defined in NIST Internal Report (NISTIR) 8060: "Guidelines for the Creation of Interoperable SWID Tags" to Concise Software Identification Tags (CoSWID) which provide additional functionality but with better memory efficiency.

The Attestation procedure that is in development by the IETF Remote Attestation Procedures Working Group uses the CoSWID format. This is because whereas the SWID format only has few required fields, optional information fields can add up to be a magnitude higher than what Remote Attestation requirements would permit, especially on resource constrained devices[15]. On the contrary, CoSWID tags are much more efficient in occupying memory because of the use of Concise Binary Object Representation (CBOR)[16]. This maps human readable text to concise binary labels which greatly reduces the amount of memory that software identification tags use, and extends the use of CoSWID and SWID in a broader ecosystem[15].

4.4 Challenge-Response Based Remote Attestation on a TPM Simulator

The prototype that models a Challenge-Response Attestation on top of a TPM simulator uses the structures defined in the previous sections. It can be used to create an abstract example of a Remote Attestation carried out on the same device, with the Verifier and Attester running in the same Docker container[].

The eventual aim of the open-source project is to extend this utility to separate devices, and carry it out on a network. In the most recent update to the code base that was made this week, the implementation extends the verification of the PCR values that are shared between the Attester and the Verifier.

The implementation uses the Constrained Application Protocol (CoAP)[17] by employing the libcoap library to communicate between the Attester and the Verifier. The CoAP Protocol provides an efficient "request/response" interaction which serves this implementation well. Moreover, since it is designed for resource constrained devices, it offers little overhead, and can be used in conjunction over the Web.

However, it does have a maximum transmission unit (MTU) of 1500 bytes and uses UDP and the Datagram Transport Layer Security. The resulting fragmentation is no good for the transmission of packets which exceed the smaller payloads that CoAP is designed for. However, implementation over the Block-Wise Transfers in CoAP[18] is currently in progress and would improve the transmission of the data from the Attester to the Verifier, especially when the functionality is extended to operate on separate devices.

The Verifier begins by setting an event log with CBOR, and creating a CoAP client session by determining an endpoint and initializing the Enhanced System API from the Trusted Software Stack. After marshalling the information that it wishes to send to the veri-

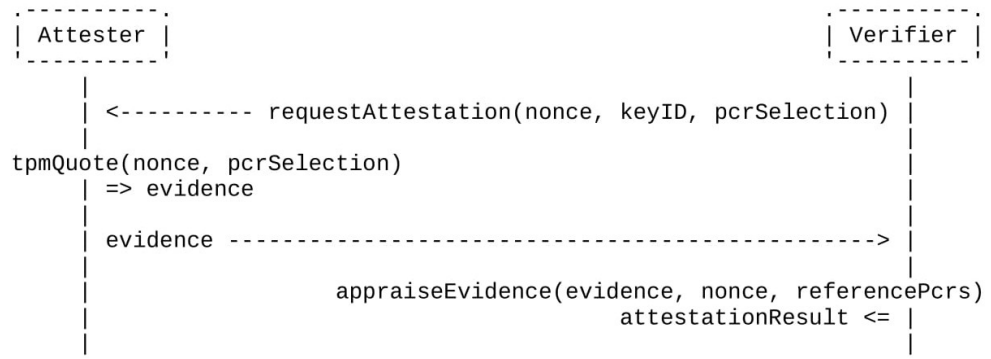

Figure 4: Prototype of a Challenge Response Remote Attestation

fier, it adds the data to an attestation request and sends a CoAP message through the `coap_send()` function. The attestation request is formulated by first generating a nonce by accessing the random number generator on the TPM simulator, and then building the attestation request by combining the selected PCR's which are to be checked for the values they hold.

The CoAP resource handler is responsible for handling the evidence once it receives the TPM Quote from the Attester. Upon arrival of the quote, it unmarshals the data by converting it from binary to C Structures, and then proceeds to verify the information enclosed within the evidence. It uses a comparison to the size of `TPM2B_ATTEST` to verify the length of the signature and the attestation data, and upon success loads the TPM key. This allows the Verifier to check the attestation signature and the nonce, after which it moves to check the PCR values and compares them against values it has access to. Note that the Verifier asks for the SHA-256 PCR Bank from the Attester, and thus loads the reference PCR values and computes a digest over them itself by using the public part of the key. It then compares them to evidence it receives from the Attester, and upon a valid match declares the PCR digest valid. Upon success, it flushes the handles, and moves to free the Enhanced System API objects and print to the standard output the results of the attestation.

On the contrary, the Attester begins by listening on the port address and initializes the CoAP resource handlers and waits for a set amount of time for a connection to be established. The attestation handler begins by setting up the Enhanced System API, and begins to take the CoAP PDU data. After converting the CBOR binary representation to structures that it can parse, it begins by reading the nonce and copying it back into the a field that can be set with the evidence that the Attester will eventually send back.

It selects the PCRs that the Verifier requests and loads the TPM key into its environment. It makes a call to the `TPM2_Quote` function and gives in the parameters necessary to take the measurements from the PCRs, sign with the Attestation Key, and adds the nonce back to the data. It then prepares to send the response back by converting it back into CBOR and marshalling the data and adding it back to the CoAP functions on the established connection with the Verifier. After sending the data back, it flushes its handles and finalizes the ESAPI.

This finishes a protocol flow of the Remote Attestation scheme in a Docker container. The

efforts that are currently being put into are concerned with creating a complete library which shall simplify the dependency matrix as it exists currently. Moreover, the addition of block-wise CoAP data transfers allows the possibility of larger amounts of data to be transferred between the parties for a more representative attestation scheme.

5 The Trusted Software Stack

The Trusted Platform Module Software Stack is the software specification produced by the IETF Working Group. It provides a set of standard API's to access the TPM and has a scalable implementation which can be designed for high end systems as well as resource constrained low end systems[19].

The TCG has a separate Working Group that works on the TSS, operating in conjunction with the TPM Working Group to create an ecosystem where programmers do not have to worry about the low-level interactions with the TPM. The TSS consists of the following layers:

- The lowest level of the stack has an OS specific TPM device driver. This driver is responsible for the reading and writing of the data to and from the TPM. It also establishes the handshaking protocols with the TPM to link it to the CPU. On UEFI systems, for instance, the TCG protocol[20] provides the driver functionality. For systems running Windows, the TPM Base Services (TBS) [21] provides the TPM resources by linking the kernel to the user mode.
- The Resource Manager is the virtual memory manager for the TPM responsible for context management. It swaps objects, sequences, and sessions from the limited TPM memory. The functionality could be passed onto a higher level in the stack, and for devices that are resource constrained, the stack may not have a resource manager.

  The Access Broker, similarly, is another abstraction to ensure safe memory access by synchronizing processes that access the TPM. This ensures that a process can successfully complete an operation on the TPM, without interfering another process.
- The TPM Command Transmission Interface (TCTI) is responsible for handling the information from the lower parts of the stack. The TCG specifications regarding the TCTI attribute the responsibility of standardizing an interface with which the upper layers of the stack can interact with. This is important because, whereas the TPM structure is uniform, there are different types of TPM's on the market and in use which are device specific, and different than the PC platform.
- The System API (SAPI) is part of the stack that provides the entire functionality of the TSS to the user. It can handle low-level calls made from the firmware, the BIOS, the Operating System etc. It does this by taking structures in C and converting them to TPM command byte streams by using the TCTI which is configured for the respective platform, and vice versa upon receiving the byte streams and converting them back into C structures. However, even though it provides a greater loci of functionality, it requires a decent amount of expertise with the interface to the TPM to use.
- The Enhanced System API (ESYS) sits above the the System API and is intended

to facilitate application calls to the TPM. As the name suggests, its principal goal is to enhance the System API. It does this by reducing the complexity of accessing the TPM and providing support for cryptographic operations to application that wish to access secure sessions by performing a Hash-based Message Authentication Code (HMAC). It can also dynamically load the TCTI modules on a configured system that has the packages for the TSS set up according to the distribution package that it is running on.

Whereas it provides a considerably less complex interface, it would be amiss to recommend that accessing the TPM with the ESYS API can be done without a considerable understanding of the TPM.

- The Feature API (FAPI) provides a higher level abstraction to application developers who wish to access the TPM. This API captures most of the functionality associated with the TPM by a single library: libtss2-fapi.

5.1 Remote Attestation with TPM2-TSS

The community surrounding Remote Attestation using a TPM has grown quite significantly with open-source software implementations. Google has been working on an open-source remote attestation scheme which allows the validation of machine identity and state written in Go[22]. The TPM2-TSS working group at IETF manages the TPM2-Software Community which has a schematic explanation of using the software stack for a bare bones implementation of Remote Attestation, along with bash scripts that implement a ”simple attestation framework”.

Moreover, the IETF Remote Attestation Procedures Working Group has been working on developing a prototype of the challenge-response protocol over a TPM simulator written in C. It is a proof-of-concept from the Remote Attestation Architectures [7] and aims to work towards standardizing the formats for claims and evidence, and the procedures which are used on this data for verification.

A bare bones implementation from the TPM2 Software defines stages to the attestation framework, and a simplistic model consisting of the following entities:

1. The Device Node is the Attester whose system state the protocol aims to verify. The TPM on this device enables and assists the development of the `TPM_Quote` which includes the digests of the PCR values. The platform then uses a cryptographic signing key to sign this digest. This signing key is linked to the inextricable Endorsement Key, which is tied to the unique device identity of the platform.
2. Privacy Certificate Authority is the trusted ”Verifier” that validates the data generated by the ”Attester”. It holds the ability to verify that the key used to sign the PCR values is linked to the Endorsement Key.
3. The Service Provider is what the entity the Attester would like to access in the events proceeding a successful attestation. Accordingly, the Service Provider needs assurance that the Attester has a recognizable device identity, and that the system is in a known, acceptable state.

The implementation is divided up into three parts. Initially, to start with the Attestation procedure, the device node sends its key to the service authority which evaluates the genuineness of the Endorsement Key of the platform. Upon successful corroboration of

the device identity, the Service Provider produces a registration token which it shares with the Certificate Authority. This is shared with the device node only if it can prove the association of its attestation key with the endorsement key, which if it does, proves the device identity.

In a secondary stage, a device node wishes to access the service provider. To do this, it must prove that the Attestation Key it is using is produced by a TPM, and that the Attestation Key is bounded to the corresponding Endorsement Key of the TPM. This stage produces a Service Token, which is only shared with the Device Node if it can legitimately prove its identity.

In the last stage, the Service Provider requests an attestation quote from the Device which is signed by the Attestation Key. The attestation quote is validated by the Service Provider, and the device node is provided access to resources as a result.

6 Conclusions

Trusted Computing offers a broad spectrum of use cases to be deployed in systems that are networked together in huge data centers, or even small scale companies which would not like their resources to be accessed by a competitor. It ensures that the state of the platform that is being used is known and trusted to a larger extent than it is without any primitives in place.

The principal goal of this project was to touch upon the state of the art that currently exists in status quo regarding Remote Attestation procedures with the Trusted Platform Module 2.0. Whereas it is difficult to completely capture the extent of adaptation of these measures, the research and the progress on operations to standardize and increase the usability of these procedures do show a sign of an increased reliance on Trusted Computing. With open source software implementations, and an increase in the work that is going into increasing access to attestation capabilities in resource constrained devices, it would not be amiss to say that Remote Attestation, and the primitives around the procedures have compounded itself into a necessity, which increases as the amount of devices and the data that is collected increases.

The potential of harm, and the ability of being exposed has been a concern with the advent of technology since the start, however, with increase in the complexity of systems and the reliance that the humankind has on systems like self-driving cars, and networked speakers with microphones that are constantly listening, there is an absolute need to focus on proactive measures which reduce the propensity of harm to an acceptable level.

The TCG, along with the community of individuals that have contributed to the Working Group, have set out goals that do not seem unattainable, all things considered. The development, however, much similar to any standardized operations within the tech community will take a decent amount of time and revision to make sure that the methods and operations are highly reliable. The current adaptation of measures that exist have loopholes which have arguably constrained the adaptation of the existing infrastructure. With work that is being put into increasing interoperability, and improving the ability of devices to connect with each other and interact with standardized operations, the adaptation of trusted computing may not be that far off.

This project was an attempt at understanding where progress currently is. What follows is a better understanding of the operations and prerequisites that go into building better and secure systems, with an insight into where work should be put into next. Remote Attestation, with its ability to ascertain a system's components through a set of standardized operations shall only increase in its use-cases and as a consequence its adaptability, which is why the work that is being put into making it more secure and reliable.

Establishing the TPM as a root of trust is perhaps the most important part of understanding the ecosystem around which remote attestation is built. If the hardware support in hybrid attestation schemes fails to live up to its ability, then software solutions can only offer so much. However, with the advent of the Trusted Software Stack, and the Trusted Platform Module, both working in unison, better solutions could be worked upon which bridge the gap in possible vulnerabilities that exist in the current standard.

References

[1] Rodrigo Steiner and Emil Lupu. “Attestation in Wireless Sensor Networks: A Survey”. In: *ACM Computing Surveys* 49 (Sept. 2016). DOI: 10.1145/2988546.

[2] *Remote Attestation With Tpm2 Tools*. https://tpm2-software.github.io/. Accessed: 2021-01-13.

[3] *IEEE 802.1AR-2018 - IEEE Standard for Local and Metropolitan Area Networks - Secure Device Identity*. Reference. Published. 2018. URL: https://standards.ieee.org/standard/802_1AR-2018.html#Standard.

[4] Henk Birkholz et al. *Remote Attestation Procedures Architecture*. Internet-Draft draft-ietf-rats-architecture-08. IETF Secretariat, Dec. 2020. URL: http://www.ietf.org/internet-drafts/draft-ietf-rats-architecture-08.txt.

[5] Henk Birkholz et al. *Reference Interaction Models for Remote Attestation Procedures*. Internet-Draft draft-ietf-rats-reference-interaction-models-01. Work in Progress. Internet Engineering Task Force, Oct. 2020. 22 pp. URL: https://datatracker.ietf.org/doc/html/draft-ietf-rats-reference-interaction-models-01.

[6] “Department of Defense Trusted Computer System Evaluation Criteria”. In: *The ‘Orange Book’ Series*. London: Palgrave Macmillan UK, 1985, pp. 1–129. ISBN: 978-1-349-12020-8. DOI: 10.1007/978-1-349-12020-8_1. URL: https://doi.org/10.1007/978-1-349-12020-8_1.

[7] *Trusted Platform Module Library Part 1: Architecture*. Specification. Trusted Computing Group, 2019.

[8] Ariel Segall. *Trusted Platform Modules: Why, when and how to use them*. The Institution of Engineering and Technology, London, United Kingdom, 2016.

[9] *BIOS Integrity Measurement Guidelines (Draft)*. Specification-Draft. National Institute of Standards and Technology, 2011.

[10] *TPM 2.0 Keys for Device Identity and Attestation*. Work in Progress - Specification. Trusted Computing Group, 2020.

[11] *TCG TPM v2.0 Provisioning Guidance*. Reference. Published. Trusted Computing Group, 2017. URL: https://trustedcomputinggroup.org/wp-content/uploads/TCG-TPM-v2.0-Provisioning-Guidance-Published-v1r1.pdf.

[12] *TCG Trusted Attestation Protocol (TAP) Information Model for TPM Families 1.2 and 2.0 and DICE Family 1.0*. Specification. Trusted Computing Group, 2019.

[13] *TCG Reference Integrity Manifest (RIM) Information Model*. Specification. Trusted Computing Group, 2020.

[14] Henk Birkholz et al. *A YANG Data Model for Challenge-Response-based Remote Attestation Procedures using TPMs*. Internet-Draft draft-ietf-rats-yang-tpm-charra-05. Work in Progress. Internet Engineering Task Force, Jan. 2021. 53 pp. URL: https://datatracker.ietf.org/doc/html/draft-ietf-rats-yang-tpm-charra-05.

[15] Henk Birkholz et al. *Concise Software Identifiers*. Work in Progress draft-ietf-sacm-coswid-08. IETF Secretariat, Nov. 2018.

[16] C. Bormann and P. Hoffman. *Concise Binary Object Representation (CBOR)*. RFC 7049. RFC Editor, Oct. 2013.

[17] Z. Shelby, K. Hartke, and C. Bormann. *The Constrained Application Protocol (CoAP)*. RFC 7252. http://www.rfc-editor.org/rfc/rfc7252.txt. RFC Editor, June 2014. URL: http://www.rfc-editor.org/rfc/rfc7252.txt.

[18] C. Bormann and Z. Shelby. *Block-Wise Transfers in the Constrained Application Protocol (CoAP)*. RFC 7959. RFC Editor, Aug. 2016.

[19] *TCG TSS 2.0 Overview and Common Structures Specification*. Specification. Trusted Computing Group, 2019.

[20] *TCG EFI Protocol Specification*. Specification. Trusted Computing Group, 2016.

[21] *Microsoft TPM Base Services*. https://docs.microsoft.com/en-us/windows/win32/tbs/about-tbs. Accessed: 2021-01-13.

[22] *Go Attestation*. https://github.com/google/go-attestation. Accessed: 2021-01-13.