Trusted Platform Module in Windows 11 – Teaching Aide

Prepared By-

Brandon Ma (30290589)
Huma Naiman (30303242)
Arunima Negi (30281210)
Karanveer Singh (30282286)

Introduction and Context

Modern computing faces escalating threats to system integrity and data confidentiality, driving the need for stronger security measures. The Trusted Platform Module (TPM) conceived by the Trusted Computing Group (TCG) in the early 2000's was created to establish the foundation for secure operations in an ever-changing world. TPM technology started with the release of TPM 1.1b, evolving into TPM 1.2 and TPM 2.0, each standardizing critical mechanisms for trusted computing environments.

A TPM is a dedicated, secure, and trusted crypto processor in modern Personal Computers (PC) that generates strong encryption keys, securely stores and restricts access to keys and other data, verifies signatures, performs security attestations, generates good quality random numbers, uses a unique RSA key burned into the chip for device authentication, and ensures platform integrity by measuring and storing security-relevant data during the boot process [1]. According to Microsoft, TPM includes multiple physical security mechanisms to make it tamper-resistant and malicious software cannot tamper with the security functions of the TPM [1].

In this resource, we examine the TPM standard, focusing on TPM 2.0 and its usage in Windows 11's Secure Boot feature and the attacks that TPM's can be vulnerable to.

Definitions

Root of Trust (RoT) - The trusted hardware or software component that provides the foundation for all security functions. A TPM RoT serves as the first link in a chain of trust that subsequent links build upon [1].

Trusted Execution Environment (TEE) - A secure area within a processor that runs code in isolation from the rest of the system to protect sensitive data and operations from attacks, including from the host Operating System (OS).

Historical Development

In the late 1990's to early 2000's, software-based security was increasingly insufficient against sophisticated attacks such as rootkits and malware targeting firmware. Software alone could not guarantee the trustworthiness of a system. A hardware-based RoT was needed to establish a trusted computing base, thus TCG was formed to develop standards ensuring hardware-based security, leading to the development of the TPM specification.

TCG's Objectives

When developing and evolving the TPM standard, TCG carefully weighed multiple variables beyond just technical security considerations. User privacy was a paramount concern and the design emphasized giving users control over their data and limiting the possibility of misuse or unauthorized surveillance [2]. Market adoption factors also influenced the design, as TCG sought

broad interoperability with various hardware and software vendors to promote widespread adoption across OS platforms. This meant balancing advanced cryptographic functionality with ease of deployment to lower barriers for Original Equipment Manufacturers (OEM) and enterprises. Cost and manufacturability were critical commercial considerations as well [3]. TPM's had to be feasible to produce at scale and integrate into mainstream devices without prohibitive expense. Additionally, the evolving threat landscape necessitated updates for cryptographic agility and flexible authorization mechanisms to meet diverse use cases, from consumer PC's to enterprise security scenarios [3]. Collectively, these variables shaped TPM into a robust, versatile hardware-based security foundation.

Microsoft steadily integrated TPM support into Windows platforms, culminating in Windows 11's mandate of requiring TPM 2.0 to protect against contemporary threats.

TPM 1.1

Released around 2003 as an early standard by TCG, TPM 1.1 was the first version to see widespread deployment in enterprise laptops and servers.

Key Features:

- Provided basic mechanisms for secure key storage and protection.
- Supported measured boot; the ability to record and report integrity measurements for firmware and bootloader code.
- Used RSA and SHA-1 algorithms for cryptographic operations.

Limitations:

- Not as standardized when compared to later versions.
- Lacked support for modern cryptographic standards.
- Restricted authorization mechanisms and lacked strong policies.
- Had limited flexibility in terms of use cases and integration.

TPM 1.2

Finalized between 2003 and 2011, TPM 1.2 became the industry benchmark for hardware-based trust.

Key Features:

- Supported robust hardware protection for keys, leveraging tamper-resistant silicon.
- Continued to use RSA (2048-bit keys) and SHA-1 hash algorithms exclusively.
- Introduced sealed storage, enabling encryption keys/data to be bound to a specific platform state and released only when the device is in a trusted configuration.
- Used a single hierarchy, which limited the flexibility for complex enterprise scenarios.

Limitations:

- Security limited by sole use of SHA-1, now considered obsolete due to vulnerabilities.
- Only supported discrete hardware TPM.
- Did not allow for algorithm flexibility or more advanced authorization models.
- Many modern security requirements cannot be met by TPM 1.2.

TPM 2.0

First announced in 2014 and finalized in 2019, TPM 2.0 is now the global security baseline. As Microsoft states, "Windows 11 requires TPM 2.0 as a part of its hardware security baseline to ensure systems are resilient against modern attack vectors" [1].

Key Features:

- Supports multiple cryptographic algorithms: RSA, ECC, SHA-256, AES, and more, allowing much stronger security and crypto-agility.
- Implements flexible authorization and access controls, supporting more complex security policies and enterprise use cases.
- Permits multiple hierarchies, so various use cases and domains (e.g. firmware, application, virtualization) may be independently managed.
- Can be implemented as discrete hardware, within a TEE, or software.
- Provides comprehensive support for remote attestation, device health verification, and compliance with modern platform security requirements.

Limitations:

- Not backwards compatible with TPM 1.2 devices.
- Software must be updated for newer protocol capabilities.

Enterprise and Practical Considerations

Modern PCs generally include TPM 2.0 either as a discrete chip or firmware TPM integrated into a Central Processing Unit (CPU). Windows 11 OEM's chip with TPM implemented and available to use in their hardware, such as Intel's Platform Trust Technology or Advanced Micro Systems (AMD) fTPM. Users and administrators can verify TPM status through Windows Security or the TPM management console. Additionally, Microsoft has automated TPM provisioning to reduce configuration complexity during deployment [1].

TPM-backed attestation allows enterprises to verify device health before granting access to sensitive resources; essential for zero trust architectures that require device integrity assurance [4].

Deep Dive into the TPM 2.0 Standard by TCG

A TPM is a logical component in a system, where it has a separate state from its host and is implemented on physical resources whether direct or indirect [5, p.25-26]. It contains a processor, Random Access Memory (RAM), Read Only Memory (ROM), and flash memory [5, p.26]. TCG defines the TPM implementation in such a way where the actual construction is flexible, if all requirements in the specification are adhered to [5, p.26]. It is permitted for a TPM to have all its components contained in a physical chip, separate distributed components within the system, or even on a CPU where there is a TEE that is separate from the rest of the CPU operations [5, p.26]. The goal of the TPM is to establish a hardware RoT to which all future trust of the system is built upon.

The Concept and Establishment of Trust

For the purposes of the TPM 2.0 specification as described by TCG, **trust** means that an expected behaviour is conveyed, but it does not necessarily constitute that the behaviour itself is worthy of trust (e.g. we expect a bank will behave like a bank and that a thief will behave like a thief) [5, p.24]. Windows 11 requires a certified TPM 2.0 and TCG certifies TPM 2.0 implementations for compliance with its specification [6-7]. Therefore, a platform for which Windows 11 is installed with a TPM 2.0 is considered to be trusted.

TPM 2.0 Components and Features

Shielded location, Protected Capability, and Protected Object defined below form the basic idea that the TPM itself is a highly secure environment in which all data that enters and leaves the TPM is considered secure and correct, while all operations that manipulate data inside the TPM are itself also highly secured and correct [8, p.36].

Shielded Location – A location on a TPM that contains data that is shielded from access by any entity other than the TPM and can only be operated on by a Protected Capability [8, p. 27]. By definition, the TPM itself is also a Shielded Location [8, p. 37].

Protected Capability – An operation that is performed on data in a Shielded Location in response to a command sent to the TPM [8. p. 26]. Protected Capabilities are composed of TPM commands available to the host system through TCG's TPM Application Programming Interface (API), as well as internal Protected Capabilities only available to the TPM itself as defined by the vendor's implementation [8, p. 36].

Protected Object – An object with an encrypted sensitive portion of which the TPM will only decrypt when it is in a Shielded Location [8. p. 26]. Protected Objects can exist outside of Shielded Locations where the integrity and confidentiality are protected cryptographically [8, p. 36]. It's important to note that when referring to "access", it is defined by the non-disclosure of the sensitive portion of the object and does not consider protections against modification or loss

[8, p.36]. Integrity is maintained by the TPM performing an integrity check on Protected Objects that come from outside the TPM and not loading it into its shielded environment and throwing an error if it fails [8, p. 36].

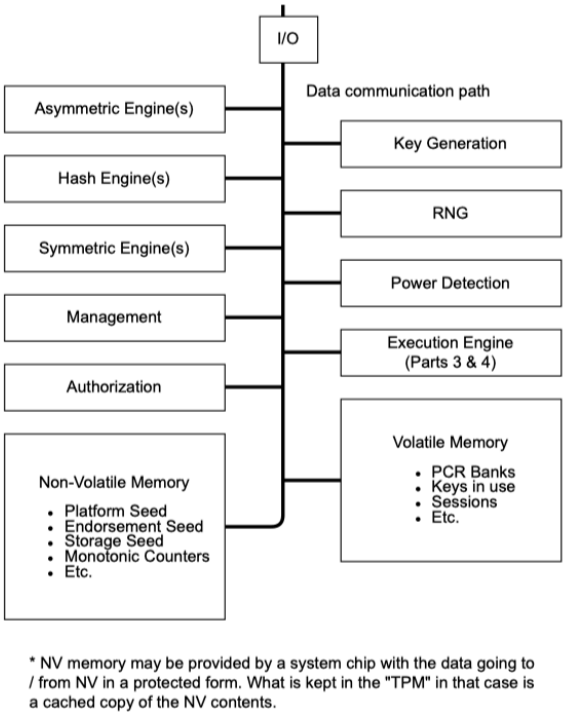

Figure 1- Architectural Overview of TPM 2.0 components [8].

Secure I/O

The I/O buffer is the only area where communications occur between a TPM and the host system and is not considered part of the TPM itself [5, p.25][8, p.43]. Commands are placed on the I/O buffer and the response from the TPM is retrieved [8, p.43]. The TPM 2.0 specification does not require physical isolation from the rest of the system but does require the implementation to have the appropriate protections on the data that is passed on the buffer (i.e. incoming data must be a Protected Object that is checked for integrity before being permitted to enter the TPM and any data must be encrypted and transformed into a Protected Object before leaving the TPM) [8, p.43].

Cryptography Subsystem

The cryptography subsystem is a logical set of components (asymmetric engine, hash engine, symmetric engine, random number generator, key generation) that are described below that implement cryptographic algorithms or primitives that are used together to execute cryptographic processes on an as needed basis for the TPM or the host system.

Asymmetric Engine(s)

The TPM 2.0 specification requires at least one asymmetric algorithm to be implemented, which is used for attestation identification, and secret sharing [8, p.44]. The specification supports RSA or ECC using prime curves and defines the methods used with each algorithm [8, p.44].

Hash Engine(s)

The Hash Engine implements TPM 2.0 approved hashing algorithms to provide integrity checks and authentication either for the TPM in its internal functions or it can be made available for the host system to utilize [8, p.43]. The guidance from TCG is to implement a hash algorithm that has comparable security strength as the strongest implemented asymmetric algorithm [8, p.43]. The TPM also implements the Hash Message Authentication Code (HMAC) algorithm [8, p.43].

Symmetric Engine(s)

This part of the cryptographic subsystem implements symmetric encryption algorithms to encrypt some command parameters (usually authentication information) and Protected Objects [8, p.47]. The TPM 2.0 specification only requires Cipher Feedback mode (CFB) to be implemented as the block Cipher, as it is used for encryption of Protected Objects when they are not stored in the Shielded Location [8, p.46-47]. The specification also supports Counter (CTR), Output Feedback (OFB), Cipher Block Chaining (CBC), and Electronic Code Book (ECB) block cipher modes [8, p.48].

Random Number Generation (RNG)

One of the most critical components for ensuring the proper security can be achieved for our cryptographic algorithms is the generation of sufficiently random numbers. To ensure an appropriate level of randomness that reduces bias when compared to a pseudorandom number generator (PRNG), the TPM 2.0 specification requires at least one internal source of entropy that can be collected from noise, clock variations, air movement, or some other event or measurement of the physical world [8, p.54]. The entropy collector is a sensor that provides the entropy measurement to a state register that is not visible to an outside process or even another TPM component, which is then passed to a mixing function that is used to generate random numbers [8, p.53-54]. The mixing function itself is permitted to be a PRNG, with the principle that the entropy source is updated often and potentially combined with multiple entropy sources that will effectively remove the bias that is an undesired property [8, p.54]. Random numbers are used in nonce generation, key generation, and randomness in signatures [8, p.53].

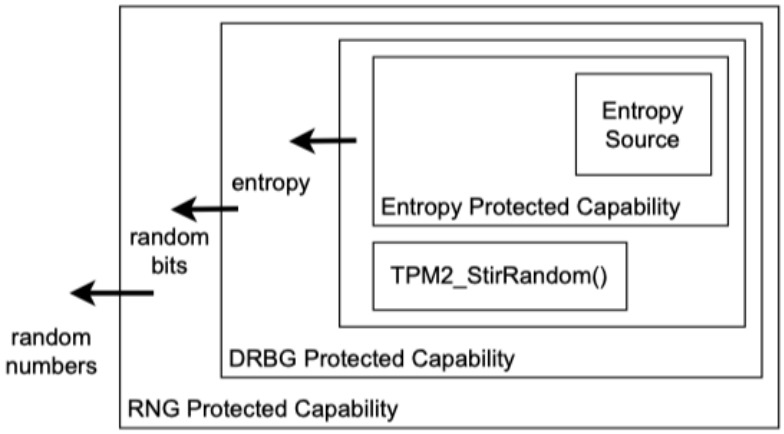

Figure 2 - Process of Random Number Generation in TPM 2.0. An entropy source is provided to a mixing function to remove bias to generate quality random numbers [8].

Key Generation

Depending on the use case of the key, the TPM 2.0 spec defines certain methodologies for generating a secure key, derived from the RNG component directly, from a derivative seed utilizing the RNG component, or another secret value [8, p.49-50]. There are multiple Key Derivation Functions (KDF) as defined by the specification and it also describes the situations in which each KDF should be used, with mechanisms to detect and discard known weak keys for certain algorithms to ensure secure key generation [8, p.50, 53].

Authorization

The authorization subsystem checks for the appropriate authorization for the use of each Shielded Location has been provided before a command can be executed [8, p.56].

Non-Volatile (NV) Memory

Provides persistent data storage that can be moved to volatile memory for quicker access for other operations [8, p.59].

Volatile Memory

Essentially RAM in the TPM that stores data temporarily, but with a unique definition when compared to RAM on the host system where a power loss event does not have to guarantee the loss of data, which is implementation dependent [8, p.57]. It is possible that data can be stored in a single location with both volatile and non-volatile copies, as long as the properties of random access and unlimited endurance hold [8, p.57]. Not all the volatile memory is a Shielded Location, as it is possible that a portion is part of the I/O buffer [8, p.57].

Platform Configuration Register (PCR)

PCR's are Shielded Locations that are used to validate the contents of a log of measurements that can be stored in volatile or NV memory [8, p. 57-58]. The TPM's role is to maintain a log of a record of events that affect the security state of the system, where new entries (or their hash) are sent to the TPM when they occur [8, p.57]. When log data is sent to the TPM, it can make an attestation as to the value in a PCR, which is used to verify whether the content in the log data matches the record [8, p.57]. PCR's can also be used to restrict access to an object by only allowing access if the selected PCR has the correct value [8, p.57]. TPM's can contain multiple PCR's as well as PCR banks, which are collections of PCR's that are extended with the same hash algorithm, comparable to an array [8, p.57].

Secure Firmware Upgrades

TCG defines the process of upgrading the firmware on a TPM as a **field upgrade** and provides a baseline of parameters in which implementing vendors must adhere to remain secure, namely that a field upgrade cannot cause the exposure of any data that is specific to the TPM instance, which includes primary seeds and their derivative keys, hierarchy values, PCR values, NV Index allocations, persistent object contents, and clocks [8, p.67, 70]. The specification also requires that field upgrades can only accept upgrade commands and data while in progress with safeguards to abandon upgrades and recover to a state where the TPM is still functional in the event of a field upgrade failure [8, p.67-68].

TCG's TPM Software Stack (TSS)

In addition to the specification defined by TCG for the implementation of a TPM and its components, it also defines an API that allows for Operating Systems and user applications to use the functions provided by the TPM in a secure and restricted manner [9]. Combined with the TPM 2.0 specification, it is an isolated system that is separated from its host system to ensure a high level of security and assurance.

Forms of TPM Implementations

TCG designed the TPM 2.0 specification such that it describes all the commands and features that a TPM could contain but allows for the vendors to use it as a library in which it can select the features it wishes to implement to tailor it to its use case [10]. This allows vendors a greater deal of flexibility to enable platform security in a variety of use cases, while allowing them to take into consideration other factors such as cost and assurance levels [10]. Below are the forms in which a TPM implementation can take.

Discrete TPM

Provides the highest level of security and is implemented as a special purpose piece of hardware that is self-contained with protections against physical tampering, probing, and freezing [10].

Integrated TPM

An integrated TPM is also a physical piece of hardware that is built into another chip that also performs non-security related functions [10]. It provides the next best security assurance level after a discrete TPM and resists software attacks but is not designed to be tamper resistant [10].

Firmware TPM

A non-physical implementation of TPM in the form of protected software, usually as firmware running in TEE inside of a CPU [10].

Software TPM

Implemented as a software emulator that does not provide much security protection as it is vulnerable to software attacks and doesn't have the benefits of hardware security. Although generally not considered secure, it does have a use case in application development, prototyping, or testing [10].

Virtual TPM

A software implementation of the TPM, usually by a hypervisor for use with Virtual Machines (VM) that has higher security assurance levels due to its ability to provide VM and host isolation [11-12]. While a hardware TPM is not required to have a virtual TPM, it is possible to virtualize TPM hardware for use with multiple VM hosts [12].

TRUST ELEMENT	SECURITY LEVEL	SECURITY FEATURES	RELATIVE COST	TYPICAL APPLICATION
DISCRETE TPM	HIGHEST	TAMPER RESISTANT HARDWARE	$$$	CRITICAL SYSTEMS
INTEGRATED TPM	HIGHER	HARDWARE	$$	GATEWAYS
FIRMWARE TPM	HIGH	TEE	$	ENTERTAINMENT SYSTEMS
SOFTWARE TPM	NA	NA	¢¢	TESTING & PROTOTYPING
VIRTUAL TPM	HIGH	HYPERVISOR	¢	CLOUD ENVIRONMENT

Figure 3 - A comparison between different TPM 2.0 implementations [10].

Real World Windows 11 TPM Hardware: Infineon TPM SLB9672 TPM 2.0 FW15

Infineon is one of the leading manufacturers of TPM chips that are used in a variety of applications. Below are the key features of the SLB9672 TPM 2.0 FW15 model that has applications that are appropriate for PC's and servers, which should look familiar and give readers an idea of how they have decided to implement the aspects of the TPM 2.0 standard that we have discussed [13]:

- Compliant with TCG TPM Library specification revision 1.59 and PC Client Platform TPM Profile (PTP) version 1.05.
- Meeting Intel TXT and Microsoft Windows certification criteria for successful platform qualification.
- Random Number Generator (RNG) implemented according to NIST SP 800-90A using entropy source according to NIST SP 800-90B.
- 24 PCRs (SHA-1, SHA-256 or SHA384).
- 51 kB NV memory.
- Unlimited amount of NV counters (only depending on NV memory utilization).
- Pre-generation of up to 7 RSA key pairs.
- RSA (1024, 2048, 3072 and 4096 bit).
- ECC (NIST P256, NIST P384, BN P256).
- SHA-1, SHA-256, SHA-384.
- AES-128, AES-256.

Secure and Measured Boot in Windows 11

Vulnerabilities of the Boot Process

PC's face significant security threats during the early stages of the boot process. Before the OS is fully loaded, the system executes firmware instructions and initializes critical components. Most security applications such as antivirus, intrusion detection systems, and endpoint protection are loaded after the OS, making it an attractive target for adversaries because the system has not yet established the controls that protect it. [14]

Threats to the Boot Process

Rootkits – A type of malware that can run in kernel mode, using the same privileges as the OS. Due to this, they can completely hide themselves from being detected by security software. Often, rootkits are part of an entire suite of malware that can bypass local logins, record passwords, keystrokes, transfer private files, and capture cryptographic data [14].

Firmware Rootkits – Overwrite the firmware of the PC's Basic Input/Output System (BIOS) or Unified Extensible Firmware Interface (UEFI) so that it can start before the OS [14].

Bootkits – Replace the OS's bootloader (the piece of software that starts the OS) so that the PC loads the bootkit before the OS [14].

Kernel Rootkits – Replace a portion of the OS kernel so the rootkit can start automatically when the OS loads [14].

Driver Rootkits – Pretend to be one of the trusted drivers that the OS uses to communicate with the PC hardware [14].

These threats can compromise the bootloader, firmware, or kernel to gain privileged access to control the system and establish persistent access. Once embedded at this level, such threats are extremely difficult to detect or remediate, as they operate beneath the OS and can subvert traditional defense mechanisms. The result is a loss of system integrity, confidentiality, and trustworthiness.

Secure Boot

Secure Boot works utilizing cryptographic checksums and signatures. Each program that is loaded by the firmware includes a signature and a checksum. Before allowing it to execute, the firmware will verify that the program is trusted by validating the checksum and the signature against a trusted database. When Secure Boot is enabled on a system, any attempt to execute an untrusted program will not be allowed. This stops unexpected and unauthorized code from running on the system. [16]

Unified Extensible Firmware Interface (UEFI) – A specification for the firmware architecture of a modern computing platform. It is a software interface that starts before the OS and is responsible for managing the initial boot process and initializing hardware components. The UEFI firmware resides in an embedded Non-Volatile Random Access Memory (NVRAM) ship on the motherboard of a PC.

UEFI Secure Boot – A verification mechanism for ensuring that code launched by a computer's UEFI firmware is trusted. It is designed to protect a system against malicious code being loaded and executed early in the boot process before the operating system has been loaded [14].

Objectives of Secure Boot

Ensures Integrity – Guarantee that only verified and authorized code executes during system startup. By enforcing digital signatures and cryptographic validation, the system prevents the execution of tampered or untrusted boot components.

Establish a Chain of Trust – Create a sequential verification model in which each component of the boot process validates the next before transferring control. This chained approach ensures that the trustworthiness of the system extends from the firmware through the OS kernel and onward to critical drivers and services.

Provide a Verifiable Proof of Integrity – Enable systems to produce verifiable evidence of their boot state. By recording cryptographic measurements of boot components within a TPM, administrators and remote services can determine whether the platform has started in a secure and uncompromised state.

Transparency – Deliver these protections seamlessly, without requiring user intervention during startup. This approach ensures that even non-technical users benefit from strong security guarantees while minimizing the risk of human error or misconfiguration.

Secure Boot Components

Platform Key (PK) – Contains the public/private key pair owned by the OEM. It is a top-level owner key for a platform and issued to establish a trust relationship with the system's UEFI and controls access to other key databases. [16]

Key Enrollment Key (KEK) - A database containing keys from the OS or other vendors. These keys can be used to sign and update the authorized and forbidden signature databases. They are used to sign the key-update packages. [16]

Authorized Signature Database (db) – Contains cryptographic certificates, keys and signatures for trusted firmware, OS loaders, and other software components that are permitted to run during the boot process. Entries are typically X.509 certificates and hashes. [16]

Revoked Signature Database (dbx) – Contains a blacklist of cryptographic signatures for software such drivers, firmware, and bootloaders that Secure Boot should no longer trust or allow the system to load. It is regularly updated to prevent security bypasses and protect systems from known vulnerable or malicious software. [16]

UEFI Variables – Contains firmware variables where PK, KEK, db, dbx and Secure Boot state are stored. They persist across reboots and firmware updates. Manufacturers use the tamper resistant NVRAM storage inbuilt into TPMs to store these variables. [16]

Endorsement Key (EK) – A permanently stored key pair inside a TPM, configured the OEM. Typically RSA-2048 is immutable, providing a fixed hardware identity for the TPM. It serves as the foundation for the TPM's security functions, being the secure root from which other keys are derived. The private portion of the EK is protected by the TPM's hardware security and is never exposed to the outside world. [16]

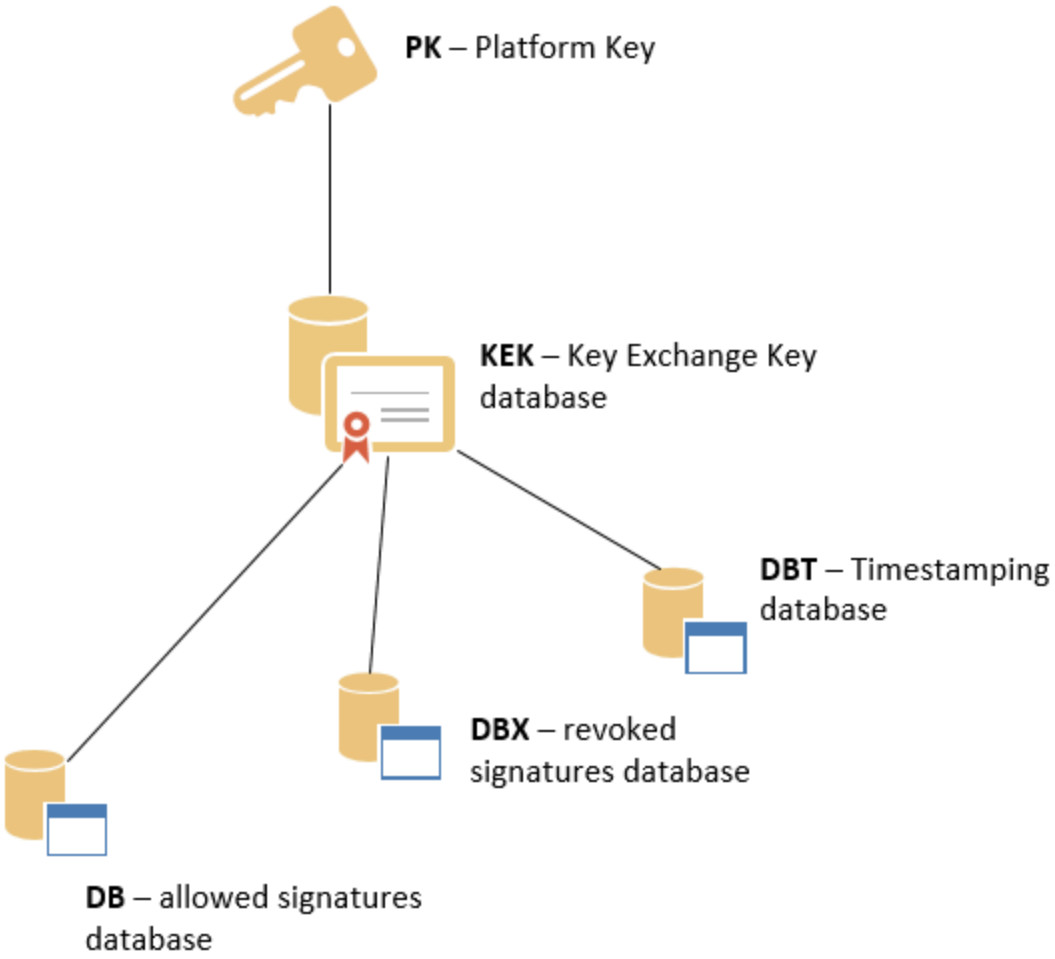

Figure 4 - Hierarchy of UEFI Secure Boot Variables [15].

Windows Boot Process

A typical secure boot process involves the following steps [16-17]:

1. Initialization of UEFI firmware when the system is powered on.
2. The UEFI will perform an integrity check on itself using the PK to establish a root of trust for the boot process.
3. The UEFI verifies the signature of the bootloader against the db and dbx.
4. If the signature matches on the db list, the firmware loads the bootloader into the system memory, control of the system is transferred to the bootloader. Unsigned, modified, or revoked bootloaders are refused. If the firmware is not trusted, the UEFI firmware must initiate OEM-specific recovery to restore trusted firmware.
5. Chain of trust continues, the signed bootloader then verifies the OS kernel or next stage binaries, which in turn verify the next stage, creating an ordered chain of signature checks.

6. If there is a problem with Windows Boot Manager, the firmware will attempt to boot a backup copy of Windows Boot Manager. If this also fails, the firmware must initiate OEM-specific remediation.
7. After Windows Boot Manager has started running, if there is a problem with the drivers or kernel, Windows Recovery Environment (Windows RE) is loaded so that these drivers or the kernel image can be recovered.
8. Windows loads other kernel drivers and initializes the user mode processes, including security software.

Note*- Steps 1 to 4 → Secure Boot, Steps 5 to 8 → Trusted Boot

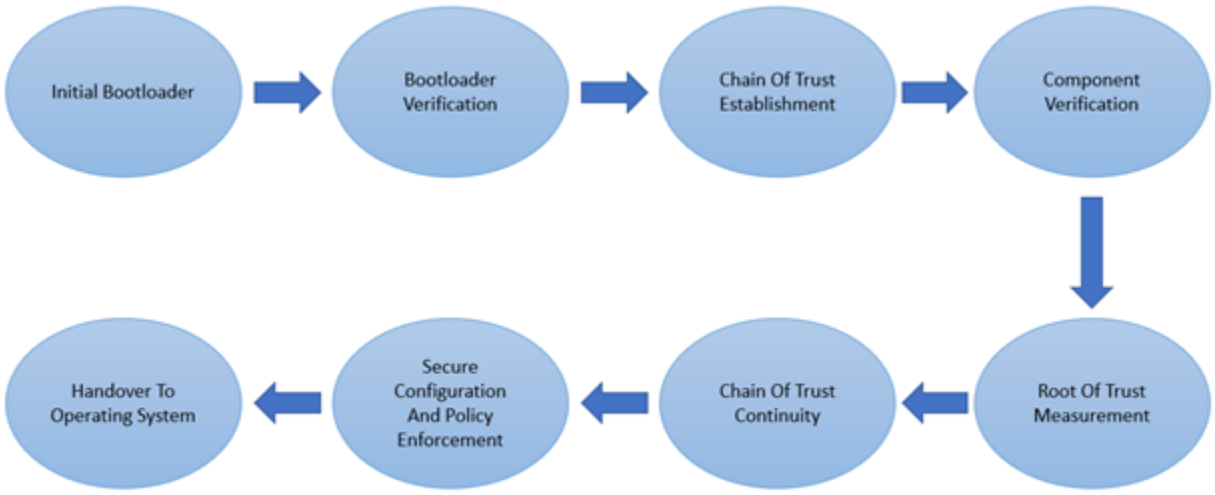

Figure 5 - Overview of Secure Boot [18].

Note- For a step-by-step demo with screenshots on verifying secure boot status in your windows machine and enabling Secure Boot, see Demo PPT slides 1–18.*

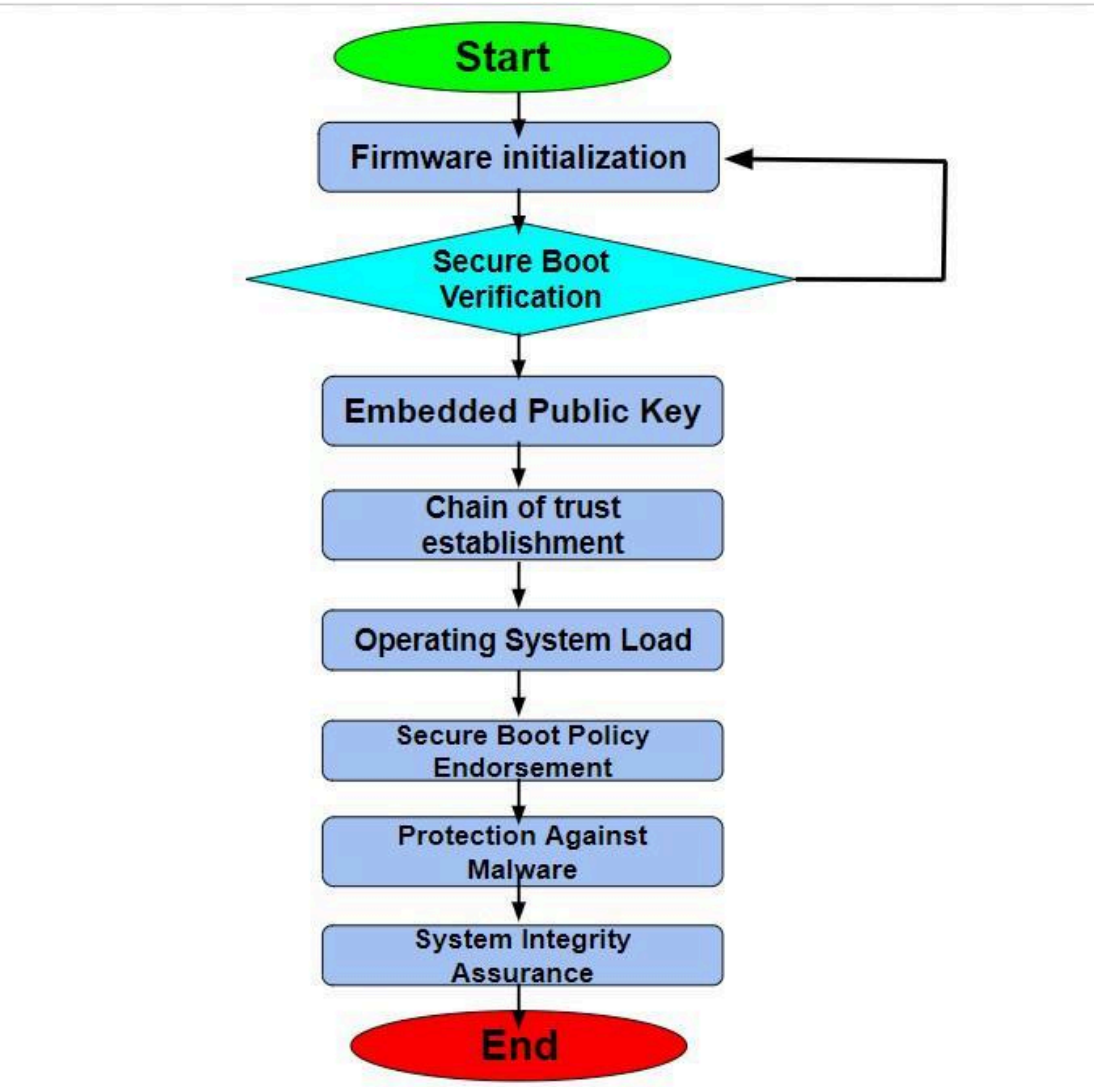

Figure 6 - Secure Boot Data Flow Diagram [19].

What Secure Boot Does and Does Not

Does

- Blocks execution of untrusted code.
- Authenticates every piece of software loaded during the boot process (UEFI, bootloader, OS Kernel, Drivers) by verifying the digital signature is from a trusted authority.
- Prevents malware like bootkits and rootkits from taking control before the OS loads.

Does not

- Record the outcome of boot processes.
- Record measurements for each stage in the boot process.
- Protect against all malware.

Measured Boot

Measured Boot is a security process that cryptographically records each component of a system's boot sequence from firmware to early drivers into the PCR's of a TPM. This creates a verifiable, tamper-evident log of what ran during boot. The client can send the measurements to a remote machine, this process is called attestation. The remote machine can evaluate the PCR measurement values and determine if the values represent a trustworthy OS state including the launch of trustworthy anti-malware software. If the measurements listed in the log do not match the TPM PCR values, the client is untrustworthy [20].

Unlike Secure Boot, it does not prevent untrusted code from running but provides a verifiable record of what was executed during startup. By giving anti-malware software a trusted record of pre-boot components, Measured Boot helps detect compromises and initiate remediation, ensuring that the system and its software can be trusted even before traditional protections are active [20]. Measured boot is about verification and logging, working alongside Secure Boot.

Building Blocks of Measured Boot

TPM Extend Operation – A function that modifies a PCR's value. This function takes the current PCR value and a new measurement (a hash of a system component like a bootloader, UEFI settings, or an operating system file and hash [20]. The extend function is order dependent. This makes the final PCR value depend on the exact sequence of measurements.

The function is defined as:

New_PCR_N = HASH (Current PCR_N ‖ New_Measurement_Digest)

PCR – Recall that a PCR is a memory location within a TPM chip that stores cryptographic information in a trusted manner. They're reset with every system boot and can't be set directly; they can only be "extended." The final value of this consecutive hash is a unique fingerprint of the entire boot process. TPM 2.0 implementations have at least 24 PCR's, where the first 24 usually measure:

- **PCR's 0-7** – Firmware and Core Root of Trust Measurement (CRTM).
- **PCR's 8-15** – OS and Application.
- **PCR's 16-24** – Often used for Dynamic Root of Trust for Measurement (DRTM).

Note- For step-by-step instructions to enable TPM 2.0 in UEFI (Intel PTT and AMD fTPM), see Demo PPT slides 19–31.*

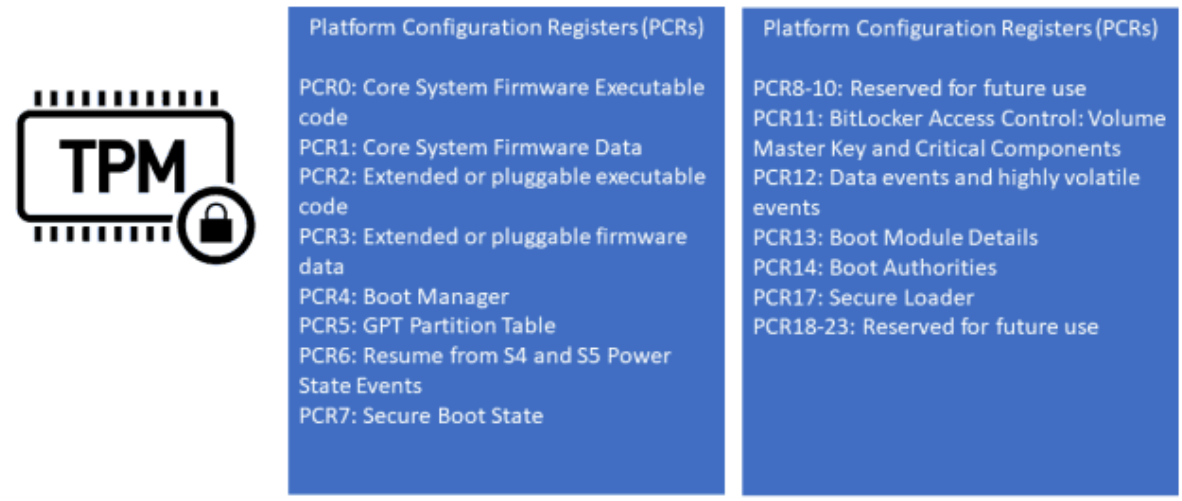

Figure 7 - PCR indexes [21].

Event logs record each measured boot component, capturing the type and data of each event in a TCG defined format. These logs, combined with TPM PCR values, allow verifiers to reconstruct and validate the boot sequence. Attestation enables a client to prove its state to a remote system by sending a signed Quote of its PCRs, including a nonce to prevent replay attacks. The Quote is signed with a unique Attestation Identity Key (AIK), which the server verifies to ensure it originates from the correct client.

Event Log – The event log is what remote or local attesters use to reconstruct what was extended and in what order. Every measured item should produce an event log entry describing the event type and data. The log and the PCR values together let a verifier recompute PCR's and check them against the quoted PCR's. The logs should be recorded in a format described by the TCG [22].

Attestation – When a client machine provides evidence about its state to a remote computer. The Measured Boot feature allows a client to send TPM PCR values to a remote computer in a trusted way. This allows that party to check remotely whether the system booted in a trustworthy state by comparing the reported values against known good ones.[22].

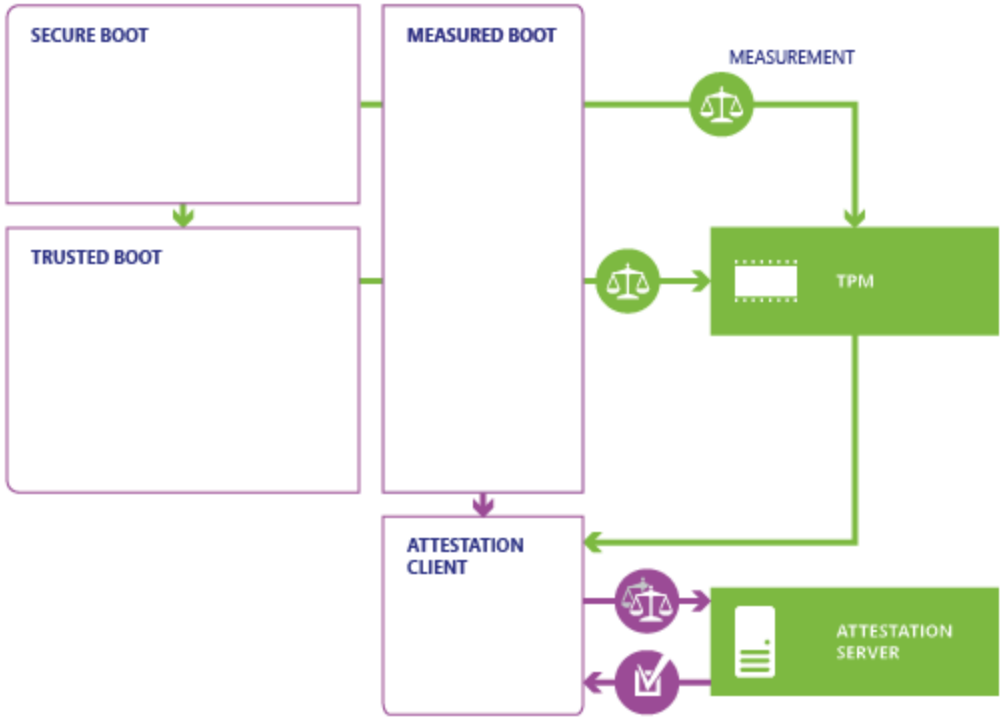

Figure 8 - Measured Boot and remote attestation process [14].

Measured Boot Process

1. **Core Root of Trust Measurement (CRTM)** – A small, immutable piece of code that is the very thing to run on the platform, often part of the CPU's firmware. The CRTM's job is to hash the UEFI firmware and extend that hash to PCR0.
2. **UEFI Firmware Measurement** – UEFI takes over and measures the bootloader, extending its hash to the next available PCR. It also measures other relevant components like the UEFI variables and configuration settings, adding their hashes to other PCR's as specified by the system's policy.
3. **Bootloader and Kernel Measurement** – The bootloader is then responsible for measuring the OS kernel and any early-load drivers before running them. It computes a hash of the kernel file and extends this hash to a designated PCR.
4. **Driver and Application Measurement** – Once the OS starts, it can continue the measurement process. Early-launch anti-malware (ELAM) drivers can measure other drivers and add their hashes to the PCR's, providing a log of the system's state before all services are fully operational.
5. **Create the Event Log** – As each measurement is taken and extended to a PCR, a corresponding entry is created in a TCG Event Log. This log provides a detailed,

human-readable record of which components were measured and what their hashes were. The PCR's only store the final chained hash, but the Event Log stores the individual measurements, providing the raw data for later verification.

6. **Attestation** – The real power of Measured Boot comes after the system is running. A remote server or a local application can request a copy of the TPM's PCR values and the TCG Event Log. The server can then compare these hashes to a known good set of values. If the values match, it's considered proof that the system has booted with a known, uncompromised configuration. If they don't match, it indicates that a component has been tampered with, even if Secure Boot didn't block it. This allows for a security audit and can be used to prevent a compromised system from accessing sensitive data or networks.

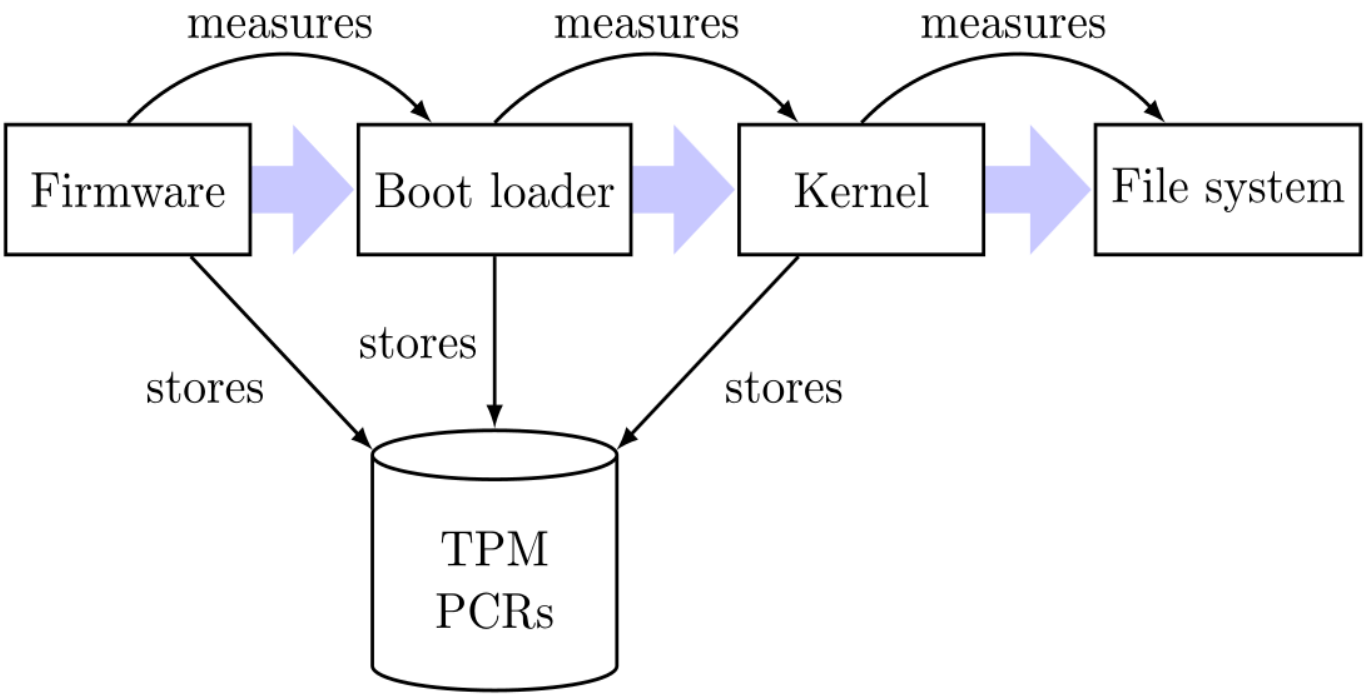

Figure 9 - Measurements taken at each stage in the boot sequence [23].

Difference in Approach

Secure Boot takes a preventative approach with a "verify then execute" model, stopping execution if verification fails. In contrast, Measured Boot takes a detection and attestation approach with a "measure then report" model, recording the components even if they were loaded and allowing the boot to continue, with the log used for later verification. Measured Boot does not prevent the system from booting if a component is found to be untrusted.

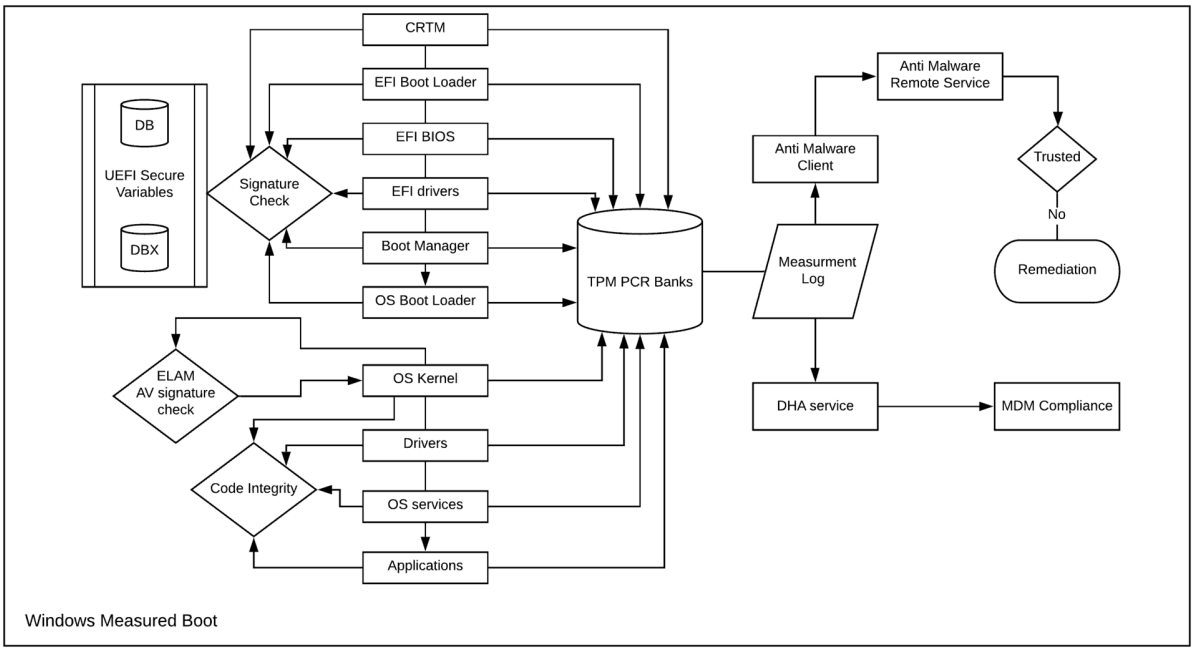

Figure 10 - Measured Boot flow diagram [24].

Note- See Demo PPT slides 32–42, for practical examples using windows powershell commands for parsing boot logs, inspecting TPM PCR registers in real-time, checking Secure Boot state and TPM status in windows 11.*

Security Analysis, Threats, and Attack Implications

TPM-FAIL and faulTPM Attacks

Introduction:

Windows 11 relies heavily on Secure Boot and the Trusted Platform Module (TPM) to ensure the integrity of the boot process. Secure Boot allows only trusted firmware and bootloaders to run, preventing malware from compromising the system at boot time. The TPM complements Secure Boot by recording cryptographic measurements of the boot sequence, storing keys, and enabling attestation and sealed secret functionalities. However, recent research has shown that vulnerabilities in TPM implementations can undermine the security guarantees of Secure Boot [25-27].

TPM-FAIL Attack

TPM-FAIL is a timing side-channel attack discovered in 2019, affecting certain hardware and firmware TPMs. The attack targets cryptographic operations, particularly RSA computations, which in some implementations were not executed in constant time. This means that subtle variations in execution time could leak information about secret keys [25].

For Secure Boot, the implications of TPM-FAIL are significant. Although the primary Secure Boot keys are stored in UEFI firmware, the TPM holds attestation keys and sealing keys that protect boot measurements and secrets. If an attacker recovers these keys, they could forge attestation responses or unseal secrets, potentially bypassing Secure Boot's measured-boot guarantees [25-26].

The attack generally involves running repeated TPM operations, precisely measuring response times, collecting thousands of samples, and using statistical analysis to reconstruct the private keys. Researchers demonstrated that BitLocker keys and TPM-protected secrets could be compromised in this manner [25].

Mitigation strategies include updating firmware and TPM software to constant-time implementations, applying security patches promptly, and ensuring multi-factor boot protection mechanisms are in place to reduce reliance on TPM-only operations [25-27].

faulTPM Attack

faulTPM is a hardware-level fault injection attack discovered in 2023, targeting firmware TPMs such as those implemented in AMD's Secure Processor. Unlike TPM-FAIL, faulTPM exploits physical faults, for example through voltage glitching, to disrupt internal TPM operations and extract cryptographic secrets [28-29].

For Secure Boot, faulTPM poses a threat because the TPM is responsible for attesting the integrity of the boot process and sealing keys. If an attacker can manipulate the fTPM to bypass these mechanisms or recover sealed keys, they can compromise measured boot integrity and access secrets that Secure Boot relies upon [28-29].

The attack typically requires physical access to the device. Specialized equipment delivers precisely timed voltage glitches during TPM operations, causing computational errors or bypassing security checks. Researchers were able to recover BitLocker keys and other sealed secrets within hours on affected systems [28-29].

Mitigation measures include keeping the fTPM firmware up to date, using TPM combined with PIN or multi-factor authentication, restricting physical access to sensitive devices, and implementing hardware protections such as tamper detection and voltage anomaly monitoring [28-29].

Real-world attacks

There have been **no public reports of real-world exploitation**. Both TPM-Fail and FaulTPM were demonstrated in research labs. Still, they highlighted a major weakness in how TPMs are implemented.

Key Takeaways:

Secure Boot relies on TPMs for measured boot and sealing operations, which are essential for maintaining system integrity at startup. Attacks such as TPM-FAIL and faulTPM illustrate that even trusted hardware can be exploited, either through subtle timing variations or physical fault injection. These vulnerabilities emphasize the need for defense in depth, including firmware updates, multi-factor boot protections, physical security, and continuous security testing [25-29].

Conclusion

When Microsoft announced TPM 2.0 as mandatory for Windows 11 in 2021, the decision sparked controversy and left many users discovering their devices could become obsolete overnight. However, analyzing TPM's integration with Secure Boot reveals strong justification for this requirement. As threats evolve, users need protections that robust cryptography provides - and foundational elements like quality random number generation and key management require hardware separation rather than vulnerable software implementations.

TPM 2.0 democratizes hardware security through accessible APIs and multiple implementation options, enabling all developers to integrate trusted cryptographic operations. The TPM story exemplifies how persistent commitment to security standards and industry collaboration can transform computing security from an afterthought into fundamental infrastructure, realizing the TCG's vision of broadly accessible trusted computing that now protects billions of users through essential hardware-backed security.

Bibliography

[1] Microsoft, “Trusted Platform Module Technology Overview,” *Microsoft Learn*, Aug. 15, 2025. [Online]. Available: https://learn.microsoft.com/en-us/windows/security/hardware-security/tpm/trusted-platform-module-overview [accessed Sep. 10, 2025].

[2] Trusted Computing Group, “Frequently Asked Questions: Design, Implementation and Usage Principles Version 3.0” *Trusted Computing Group*, 2011. [Online]. Available: https://trustedcomputinggroup.org/wp-content/uploads/Best-Practices-Document-FAQ_Final20110414.pdf [accessed Sep. 12, 2025].

[3] Trusted Computing Group, “TPM Main Part 1 Design Principles” *Trusted Computing Group*, Mar. 2011. [Online]. Available: https://trustedcomputinggroup.org/wp-content/uploads/TPM-Main-Part-1-Design-Principles_v1.2_rev116_01032011.pdf [accessed Sep. 14, 2025].

[4] Dell, “Trusted Platform Module Common Questions for Windows 11” *Dell Canada*, Jun. 16, 2025. [Online]. Available: https://www.dell.com/support/kbdoc/en-ca/000190999/trusted-platform-module-frequently-asked-questions-for-windows-11 [accessed Sep. 15, 2025].

[5] Trusted Computing Group et al., “Trusted Platform Module 2.0 Library Part 0: Introduction” *Trusted Computing Group*, Mar. 20, 2025. [Online]. Available: https://trustedcomputinggroup.org/wp-content/uploads/Trusted-Platform-Module-2.0-Library-Part-0-Version-184_pub.pdf [accessed Sep. 19, 2025].

[6] Microsoft, “Windows 11 requirements” *Microsoft Learn*, Jun. 17, 2025. [Online]. Available: https://learn.microsoft.com/en-us/windows/whats-new/windows-11-requirements [accessed Sep. 16, 2025].

[7] Trusted Computing Group, “TCG Certification Programs” *Trusted Computing Group*. [Online]. Available: https://trustedcomputinggroup.org/membership/certification/ [accessed Sep. 19, 2025].

[8] Trusted Computing Group et al., “Trusted Platform Module 2.0 Library Part 1: Architecture” Trusted Computing Group, Mar. 20, 2025. [Online]. Available: https://trustedcomputinggroup.org/wp-content/uploads/Trusted-Platform-Module-2.0-Library-Part-1-Version-184_pub.pdf [accessed Sep. 16, 2025].

[9] Trusted Computing Group, “TPM Software Stack (TSS),” *Trusted Computing Group*. [Online]. Available: https://trustedcomputinggroup.org/work-groups/software-stack/ [accessed Sep. 16, 2025].

[10] Trusted Computing Group, “Trusted Platform Module (TPM) 2.0: A Brief Introduction” *Trusted Computing Group*, 2015. [Online]. Available: https://trustedcomputinggroup.org/wp-content/uploads/TPM-2.0-A-Brief-Introduction.pdf [accessed Sep. 16, 2025].

[11] Trusted Computing Group, “What is a virtual Trusted Platform Module (vTPM)?” *Trusted Computing Group*. https://trustedcomputinggroup.org/about/what-is-a-virtual-trusted-platform-module-vtpm/ [accessed Sep. 16, 2025].

[12] Broadcom, “What Is a Virtual Trusted Platform Module” *Broadcom*. Sep. 03, 2025. [Online]. Available: https://techdocs.broadcom.com/us/en/vmware-cis/aria/aria-automation/8-16/vtpm-overview.html [accessed Sep. 16, 2025].

[13] Infineon Technologies AG, “OPTIGATM TPM SLB9672 TPM 2.0 FW15.Xx Datasheet” Infineon. Nov. 2024. [Online]. Available: https://www.infineon.com/assets/row/public/documents/30/49/infineon-slb9672-tpm20-spi-fw15xx-ds-rev1-5-2024-11-13-datasheet-en.pdf [accessed Sep. 16, 2025].

[14] Microsoft, “Secure the Windows boot process” *Microsoft Learn*, Aug. 18 2025, [Online]. Available: https://learn.microsoft.com/en-us/windows/security/operating-system-security/system-security/secure-the-windows-10-boot-process [accessed Sep. 2, 2025].

[15] J. B. Roy, “Understanding UEFI Secure Boot and how it helps to secure the Windows 10 boot process” *Linkedin*, Aug.10, 2019. [Online]. Available: https://www.linkedin.com/pulse/understanding-uefi-secure-boot-how-helps-windows-10-process-basu-roy [accessed Sep. 3, 2025].

[16] Microsoft, “Secure boot” *Microsoft Learn*, Feb. 8, 2023. [Online]. Available: https://learn.microsoft.com/en-us/windows-hardware/design/device-experiences/oem-secure-boot [accessed Sep. 4, 2025].

[17] K. Sood, S. O’Riordain, J. Geary, H. Zhu, and Intel Corporation, “Secure the network infrastructure – Secure Boot methodologies” 2019. [Online]. Available: https://builders.intel.com/docs/networkbuilders/secure-the-network-infrastructure-secure-boot-methodologies.pdf [accessed Sep. 7, 2025].

[18] eInfochips, “How Secure Boot help to Secure IoT Device,” *eInfochips*. [Online]. Available: https://www.einfochips.com/blog/how-secure-boot-help-to-secure-iot-device/ [accessed Sep. 9, 2025].

[19] GeeksforGeeks, “What is Secure Boot?” *GeeksforGeeks*, Jul. 23, 2025. [Online]. Available: https://media.geeksforgeeks.org/wp-content/uploads/20240319234526/Secure.jpg [accessed Sep. 12, 2025].

[20] Microsoft, “Secured Boot and Measured Boot: Hardening Early Boot Components Against Malware*”* *Microsoft Learn*, Sep. 7, 2012. https://learn.microsoft.com/en-us/previous-versions/windows/hardware/design/dn653311(v=vs.85)?redirectedfrom=MSDN [accessed Sep. 13, 2025].

[21] “Bitlocker using TPM,” *ITris Academy*, [Online]. Available: https://ladyitris.wordpress.com/bitlocker-using-tpm/ [accessed Sep. 16, 2025].

[22] Microsoft, “Understand PCR banks on TPM 2.0 devices” *Microsoft Learn*, Aug. 15, 2025. [Online]. Available: https://learn.microsoft.com/en-us/windows/security/hardware-security/tpm/switch-pcr-banks-on-tpm-2-0-devices [accessed Sep. 17, 2025].

[23] “Firmware-based TPMs for embedded systems” [Online]. Available: https://stefan-gloor.ch/ftpm [accessed Sep. 18, 2025].

[24] J. B. Roy, “Windows Measured Boot - How it helps to secure Windows OS platform” *How to Manage Devices Community Blog*, Jul. 11, 2024. [Online]. Available: https://www.anoopcnair.com/understanding-windows-measured-boot/ [accessed Sep. 19, 2025].

[25] D. Moghimi, B. Sunar, T. Eisenbarth, N. Heninger, “TPM-FAIL: TPM meets Timing and Lattice Attacks” arXiv, Nov. 13, 2019. [Online]. Available: https://arxiv.org/abs/1911.05673 [accessed Sep. 16, 2025]

[26] “TPM Fail – Attacking Trusted Platform Modules,” [Online]. Available: https://tpm.fail/ [accessed Sep. 18, 2025]

[27]Microsoft, “Secure boot (OEMs),” Microsoft Learn, [Online]. Available: https://learn.microsoft.com/en-us/windows-hardware/design/device-experiences/oem-secure-boot [accessed Sep. 18, 2025]

[28] M. Weiser, P. M. Teixeira, and B. Morin, “On the Insecurity of the Trusted Computing Ecosystem,” arXiv, Apr. 28, 2023. [Online]. Available: https://arxiv.org/abs/2304.14717 [accessed Sep. 16, 2025]

[29] AMD, “AMD-SB-4005: AMD Platform Secure Bootloader Advisory,” AMD Security Bulletin, Mar. 2023. [Online]. Available: https://www.amd.com/en/resources/product-security/bulletin/amd-sb-4005.html [accessed Sep. 16, 2025].