International Journal of Emerging Research in Engineering and Technology
Pearl Blue Research Group| 9th ICAILLMBA (ALLAM) 2025-Conference Proceedings
ISSN: 3050-922X | https://doi.org/10.63282/3050-922X.ICAILLMBA-117

Original Article

Leveraging Trusted Platform Modules for Hardware-Rooted Security and Robust Device Encryption

Dr. K V S R P Varma[1], Rama Kiran Kumar Indrakanti[2], Nitin Vishnoi[3], Vamsi Chitta[4]
[1]Senior Technical Architect, HCL America, NY, USA 14580.
[2]ROSS UofM, Michigan, USA 48109.
[3]DBA Candidate: ESGCI, USA 48307.
[4]R&D Engineer, Broadcom, India 411047.

Abstract *- The TPM (Trusted Platform Module) has become a core building block in modern hardware-rooted security architectures, ensuring cryptography and device integrity through its tamper-resistant protection. As an independent microcontroller, it securely stores encryption keys, credentials, and platform measurements, which allow for strong authentication and attestation mechanisms. Its presence in modern operating systems and networked devices fortifies defences against unauthorized access, manipulation of firmware, and physical attacks. In disk encryption, TPMs are used to store the disk encryption key in hardware to prevent key extraction, even when the adversary gains physical possession of the storage media. BitLocker, Secure Boot, and virtual smart cards are just a few examples of technologies whose security relies on the TPM as a hardware root of trust for ensuring that only signed software executes during the boot process and sensitive data is protected throughout the life of the device. This paper revisits the architectural considerations of TPMs, assesses its contributions towards hardware security and encrypted storage, and underlines their increased applicability in heterogeneous computing environments.*

Keywords *- Trusted Platform Module (Tpm), Hardware-Rooted Security, Secure Boot, Device Encryption, Cryptographic Key Protection, Platform Attestation, Firmware Integrity, Bitlocker, Hardware Trust Anchor, Secure Storage.*

1. Introduction

Advances in digital ecosystems, cloud interconnected infrastructure, and heterogeneous computing architectures have led to dramatic changes in the existing security paradigm. Contemporary device-level computing infrastructure, ranging from business class laptops and industrial control units, through IoT gateways and embedded processing solutions, is characterized by an operating environment where attackers have more sophisticated capabilities. They currently take advantage of firmware bugs, engage with the boot process, leverage physical access-based crypto service secrets, and maintain invisible rootkits that bypass regular software-level security tools. Consequently, the traditional, software-focused security model has been ineffective regarding providing system integrity and securing confidential data.

To address these issues, the computing community has seen a move towards hardware-based security, where trust is placed in trust anchor silicon that is integrated into the device itself. Of these, Trusted Platform Module (TPM) is perhaps one of the most widely used and successful technologies in this domain. Developed by the Trusted Computing Group (TCG) [1], Trusted Platform Module is a special purpose microcontroller-based architecture that is designed for secure generation, storage, and management of cryptographic keys, integrity measurements, and authorization policies [2]. Unlike traditional software-based key management systems, which can be susceptible to memory scraping, privilege escalation attacks, as well as privilege escalation attacks, Trusted Platform Module is well insulated from these attacks based on its use of trust anchor silicon-based security.

Its role far exceeds that of simple key storage: It enables a wide spectrum of security capabilities, including secure boot, measured boot, remote attestation, sealed storage, and hardware-backed device encryption. These capabilities together create a chain of trust starting from the earliest phases of system initialization and continuing through the device's operational lifecycle. With such systems, the execution of only authenticated firmware and software components reduces persistent threats and unauthorized modifications by orders of magnitude.

The relevance of TPM has been further reinforced by regulatory and compliance needs in the global market. Regulations such as GDPR, HIPAA, PCI DSS, and government security guidelines continue to emphasize hardware-level protection for trusted data. Additionally, operating systems have realized the need for TPM as an essential component; for example, Windows 11 requires TPM 2.0 compliance for achieving basic security on personal and business computers. Cloud computing and virtualization solutions continue to adapt virtual TPM (vTPM), enabling trusted data within virtual machines and container-based applications [3]. While the TPM technology is widely adopted, it is not without limitations. Supply chain vulnerabilities, weak firmware TPMs, complex policies of PCRs, and integration in a heterogeneous environment are some of the open issues that are still under active research. More lately, the recent awareness of confidential computing, the appearing need for post-

quantum cryptography, and the use of distributed trust models raise important questions for the future development of TPMs. In this paper, a thorough review is given of the role of the Trusted Platform Module as a foundation technology of hardware-based security. Issues such as the architecture of TPMs, the use of TPMs in secure booting of platforms as well as platform integrity, the role of TPMs in developing secure device encryption schemes, as well as the increasing applicability of TPMs in the areas of IoT, clouds, and enterprise technology areas, will be covered.

2. Background and Related Work

Trusted Computing as a concept has undergone a paradigm shift in the last two decades in light of the growing requirement for trust verification in more complex systems. When computing platforms changed from being isolated to cloud and mobile systems, the trust verification methodology in the form of traditional software security models became insufficient. This section gives a historical background in relation to the emergence of the Trusted Platform Module (TPM), a hardware-based trust verification model, and its significance.

2.1. Evolution of Trusted Computing

As the need to address the issue that software cannot be relied upon to enforce the policy in the presence of an advanced attacker grew, the field of trusted computing evolved. Malware, rootkits, and attacks at the boot level demonstrated that an attacker could compromise the system before the operating system had even initialized its defences. Therefore, trust anchors were integrated into the physical layer. The Trusted Computing Group, established in 2003, encapsulated such vision through standards for the hardware-based trust mechanisms [1]. TPM technology emerged as the face of such vision to facilitate a secure execution environment for the execution of cryptographic operations and measurements related to platform integrity. Later, TPM technology started to see adoption from server computers to consumer computers and then to cloud systems.

2.2. Early Hardware Security Modules (HSMs)

Before TPMs became mainstream, organizations relied on HSMs for protection of cryptographic keys and carrying out secure operations. They offered a strong resistance to tampering and hence had been widely used in banking, certificate authorities, and government systems. However, they were:

- expensive
- large in physical terms
- difficult to be integrated in consumer devices
- not designed for platform integrity measurement

TPMs were conceived as a lightweight, cost-effective alternative that could be embedded into everyday computing devices while still providing essential hardware-rooted security guarantees.

2.3. TPM Standards Emerge (TPM 1.2 → TPM 2.0)

The TPM standard has evolved remarkably as follows:
TPM 1.

- Implemented RSA Based cryptography
- Supported SHA 1 hashing
- Offered fixed authorization models
- Commonly found in corporate PCs and servers

TPM 1.2, although the starting point of the trusted computing era, became dated owing to the cryptographic weaknesses and rigidity of this technology.

TPM 2.
TPM 2.0 which was released in 2014 major improvements

- Support for contemporary cryptography algorithms (ECC, SHA 256)
- Flexible authorization rules
- Improving the management
- Agility
- Improved virtualization and embedded systems support

TPM 2.0 has now become the industry standard and has become a prerequisite for Windows 11 operating systems.

2.4. Comparison with Other Hardware Trust Technologies

While TPM is the most-widely-adopted hardware trust anchor, there are several related technologies that complement- or compete with-its capabilities.

- Intel Boot Guard: Boot Guard enforces the cryptographic verification of the firmware components involved in the very early stages of system boot. Unlike TPM, it does not store keys or perform attestation. It makes sure that the initial boot block is authentic [4].
- ARM Trust Zone: It provides, through Trust Zone, a hardware-enforced split between secure and non-secure execution environments, allowing for secure key storage and trusted execution [5] [6] [7]. However, it does not have standardized attestation mechanisms as found in TPM.
- Secure Enclaves, Intel SGX and AMD SEV SNP: Enclaves protect sensitive computations from privileged attackers, including compromised operating systems [8] [9] [10]. However, not providing platform-wide integrity measurement does not provide disk encryption key protection.
- Physically Unclonable Functions (PUFs): PUFs generate device unique keys based on silicon manufacturing variations. They offer strong identity guarantees, but they typically lack the full feature set of TPMs, including PCRs, sealed storage, and standardized command sets [11] [12] [13].
- Firmware TPMs: Implemented in system firmware rather than discrete chips, fTPMs provide TPM functionality but may be more vulnerable to firmware-level attacks [14] [15] [16].

2.5. Trusted Computing in Operating Systems

Modern operating systems typically integrate TPM functionality deeply into their security frameworks. They also provide a TPM runtime environment with

- TPM is utilized by Windows in BitLocker, Secure Boot, Credential Guard, and virtual smart cards.
- There are Linux components named tpm2 tools, systemd cryptenroll, IMA (Integrity Measurement Architecture), and LUKS
- TPM is employed in Chrome OS for boot verification and the protection of a user's personal
- Endpoint security solutions for the enterprise depend on TPM for device identity and attestation purposes.

This level of OS integration has made TPM an essential element that currently pervades all security infrastructures.

3. TPM Architecture and Design Principles

The TPM is a microcontroller with a unique design that provides hardware-rooted trust for computing platforms. Its architecture is a combination of cryptographic engines, secure storage, authorization logic, and measurement registers into an integrated security subsystem to operate independently of the host operating system [17]. This section examines in detail TPM architecture, internal components, principles of operation, and the design philosophies enabling its utilization as a robust trust anchor.

3.1. Architectural Overview

At its core, a TPM is a tamper-resistant hardware component that performs sensitive security operations in isolation from the main CPU. This isolation is fundamental: even if the operating system or firmware is compromised, the TPM remains secure and continues to enforce trust policies [18] [19].

A typical TPM includes the following subsystems:

- Cryptographic processors for RSA, ECC, SHA-1, SHA-256
- True Random Number Generator (TRNG) for entropy generation
- Non-volatile memory (NVRAM) for persistent key storage
- Volatile memory for temporary data
- Platform Configuration Registers (PCRs) for integrity measurements
- Authorization and access control logic
- Command processing engine implementing the TCG command set

These components collectively enable the TPM to securely generate, store, and use cryptographic keys without exposing them to the host system.

3.2. TPM 1.2 vs TPM 2.0: Evolution of Capabilities

The move to TPM 2.0 from TPM 1.2 involved an important architectural change.

TPM 1.2 Characteristics:

- Fixed cryptographic algorithms such as RSA, SHA 1
- Limited flexibility in authorization policies
- Rigid structure in PCR
- Mainly for PC class devices

- While TPM 1.2 provided foundational functionalities, issues arose in its use of SHA 1 and general lack of algorithm agility as the cryptographic standards evolved

TPM 2.0 Enhancements
TPM 2.0 provides a more modular and extensible architecture:

- Algorithm agility: support for RSA, ECC, SHA 256, SHA 384, and more
- Enhanced Authorization: policy-based access control, PCR-based conditions
- Improved PCR management: dynamic PCR banks
- More platforms supported: Desktops, Servers, IOT devices and more, embedded systems
- Better virtualization support: enabling virtual TPMs

This allows TPM 2.0 to be better suited for today's security needs, which involve cloud workloads, mobile devices, and regulated environments.

3.3. Cryptographic Engines and Key Management

TPM also has specific hardware engines that support cryptographic computations. These engines run in a secure environment that is separate from that of the host CPU.

Important Key Types Handled by TPM

- Endorsement Key (EK): an identity key provisioned by the manufacturer
- Storage Root Key (SRK): root of the storage hierarchy
- Attestation Identity Keys (AIKs): for Remote Attest
- Sealing keys: bind data to specific PCR states
- Signing keys: used for digital signatures
- Binding keys: utilized for encrypting data to TPM

Key Hierarchy
TPM uses a hierarchical key structure:

```
Root of Trust
  ├── Endorsement Hierarchy
  ├── Storage Hierarchy
  ├── Platform Hierarchy
  └── Null Hierarchy
```

This hierarchy ensures that compromise of one key does not affect others and that sensitive keys never leave the TPM in plaintext.

3.4. Platform Configuration Registers (PCRs)

PCRs are one of the most critical components of TPM architecture [21]. They store cryptographic hashes representing the system's boot state.

PCR Characteristics

- Each PCR holds a 160-bit (TPM 1.2) or 256-bit (TPM 2.0) hash
- PCRs can only be extended, not overwritten
- Extending a PCR uses the formula:
  [PCR_{new} = H(PCR_{old} \parallel measurement)]
- This ensures that even small changes in boot components produce different PCR values

PCR Usage
PCRs are used for:

- Measured boot
- Secure boot validation
- Remote attestation
- Key sealing

By binding cryptographic keys to PCR values, TPM ensures that keys are only released when the system is in a trusted state.

3.5. Sealing and Binding Operations

Sealing and binding are fundamental TPM operations that protect data based on system integrity.
Binding:

- Encrypts data using a TPM-resident public key
- Only the TPM with the corresponding private key can decrypt it
- Sealing:
- Encrypts data and binds it to specific PCR values
- Data can only be unsealed when PCRs match the expected state
- This mechanism is widely used in disk encryption, credential protection, and secure configuration storage

3.6. Authorization and Access Control

TPM 2.0 introduces a flexible authorization model that supports:

- Password-based authorization
- HMAC-based authorization
- Policy-based authorization
- PCR-based conditions
- Time-based and locality-based restrictions

Policies can be combined to create sophisticated access control rules, enabling fine-grained security for sensitive operations.

3.7. TPM Command Processing and State Machine

The TPM operates as a command-driven device. The host system communicates with the TPM through a well-defined command set standardized by TCG.
Command Flow

- Host constructs a TPM command
- TPM validates authorization
- TPM executes the command internally
- TPM returns a response

The TPM state machine ensures that commands are executed atomically and securely, preventing partial execution or unauthorized access.

3.8. Hardware vs Firmware TPMs

TPMs can be implemented in two primary forms:
Discrete TPM (dTPM)

- Separate physical chip
- Strongest tamper resistance
- Preferred for high-security environments

Firmware TPM (fTPM)

- Implemented in system firmware (e.g., Intel PTT, AMD fTPM)
- Lower cost and easier integration
- More vulnerable to firmware-level attacks

Virtual TPM (vTPM)

- Software-based TPM for virtual machines
- Relies on hardware trust anchors for isolation

Each implementation has trade-offs in performance, security, and deployment complexity.

3.9. Design Principles of TPM

The TPM is built on several core design principles:

- Isolation: cryptographic operations occur in a protected environment
- Non-exportability: private keys never leave the TPM
- Measurement: system integrity is continuously recorded
- Attestation: platforms can prove their integrity to remote parties
- Tamper resistance: hardware is designed to resist physical attacks
- Standardization: consistent behaviour across vendors and platforms

These principles collectively enable TPM to serve as a reliable hardware root of trust.

4. TPM in Secure Boot and Platform Integrity

The integrity of the boot process is one of the most important aspects of secure systems in today's world. High value targets in the early stages of the system boot process, prior to the operating system being in service with its protection mechanisms, are attractive targets for malicious actors. Malware like bootkits, rootkits, or firmware-based implants can compromise the system long before the antivirus software or the operating system kernel is in place." "The Trusted Platform Module is key to protecting against these kinds of attacks, providing support for secure boot, measured boot, and remote attestation that together form the chain of trust from power on through runtime. In this section, we analyse how TPM integrates with the firmware and operating system of the platform in enforcing platform integrity and prevent unauthorized modifications, as well as in providing cryptographic proof of system trustiness.

4.1. Foundations of Platform Integrity

Platform integrity refers to the integrity of the absence of modification or interference of the firmware, boot components, and operating system of the system by an unauthorized third party. TPM makes the integrity of the platform possible through two ways:

- Secure Boot: This ensures that only verified components run.
- Measured Boot: It records cryptographic measurements of all components, whether trusted or not.

These two mechanisms, working together, serve both purposes: prevention, which prevents unauthorized code from passing, and detection, which detects code that has malicious code.

4.2. Chain of Trust in Modern Boot Architectures

A chain of trust means this follow-through process, in which each stage of the boot must validate the integrity of the subsequent one before relinquishing control. TPM reinforces this chain through the storage of measurements inside of the Platform Configuration Registers or PCRs [20].

A typical chain of trust includes:

1. Root of Trust for Measurement (RTM)
   - Usually implemented in immutable firmware (e.g., CPU microcode or ROM).
   - Performs the first measurement.
2. Firmware (UEFI/BIOS)
   - Measures its own components and extensions.
   - Extends measurements into PCRs.
3. Bootloader
   - Validates kernel images and initial RAM disks.
   - Extends measurements into PCRs.
4. Operating System Loader
   - Measures kernel modules, drivers, and critical services.
5. Operating System Runtime
   - May continue measuring components (e.g., Linux IMA).

At each stage, the TPM records measurements using the extend operation, creating a cryptographic log of the system's boot state.

4.3. Secure Boot: Enforcing Authenticity of Boot Components

Secure Boot is a policy-driven mechanism that ensures only digitally signed and authorized firmware and software components are executed during startup.

How Secure Boot Works

- Firmware contains a database of trusted public keys stored in some secure location.
- Each boot component (bootloader, kernel, drivers) must be signed.
- If a signature is invalid or missing, the component is blocked.

TPM's Role in Secure Boot

While Secure Boot can operate without TPM, TPM enhances it by:

- Recording measurements of validated components
- Binding encryption keys to Secure Boot states
- Providing attestation evidence that Secure Boot was enforced

This integration ensures that even if an attacker bypasses Secure Boot, the TPM will detect the deviation through PCR mismatches.

4.4. Measured Boot: Recording Integrity for Verification

Measured Boot complements Secure Boot by recording the hash of every boot component, regardless of trust status.

Key Characteristics

- Does not block execution
- Provides a complete audit trail
- Enables detection of unauthorized modifications

PCR Extensions
Each measurement is extended into a PCR using:
[PCR_{new} = H(PCR_{old} \parallel measurement)]
This creates a tamper-evident log of the boot sequence.

Use Cases

- Enterprise compliance checks
- Forensic analysis
- Remote attestation
- Key sealing for disk encryption

Measured Boot is widely used in cloud and enterprise environments where trust must be verified remotely.

4.5. Remote Attestation: Proving Platform Integrity to External Parties

Remote attestation allows a device to prove its integrity to a remote verifier (e.g., enterprise server, cloud controller).
Attestation Workflow

1. TPM signs PCR values using an Attestation Identity Key (AIK).
2. The device sends signed PCR values and measurement logs to the verifier.
3. The verifier checks:
    - AIK certificate validity
    - PCR values against expected baselines
    - Integrity of the measurement log

Applications:

- Zero-trust architectures
- Secure enterprise onboarding
- Cloud VM integrity verification
- Industrial control system monitoring

Remote attestation is a cornerstone of modern endpoint security frameworks.

4.6. TPM and Operating System Integrity Frameworks

Operating systems integrate TPM-based integrity mechanisms to extend trust beyond the boot process.
Windows

- Measured Boot integrated with BitLocker
- Device Health Attestation (DHA) for enterprise compliance
- Credential Guard uses TPM to protect secrets
- IMA (Integrity Measurement Architecture) measures runtime files and binaries
- EVM (Extended Verification Module) protects metadata
- systemd-cryptenroll binds disk encryption keys to PCRs
- ChromeOS
- Verified Boot uses TPM to protect user data and enforce firmware integrity

These integrations demonstrate TPM's role as a foundational component of OS-level security.

4.7. Protection Against Boot-Level Attacks

TPM-backed secure boot and measured boot mitigate several classes of attacks:

- Bootkits and Rootkits:

Malicious code injected into bootloaders or kernels is detected through PCR mismatches.

- Firmware Tampering:

Unsigned or modified firmware components fail Secure Boot validation.

- Evil-Maid Attacks:

Attackers attempting to modify boot components while the device is unattended cannot bypass TPM-sealed keys.

- Rollback Attacks:

TPM can detect attempts to load older, vulnerable firmware versions.

- Cold-Boot and DMA Attacks:

Even if memory is compromised, TPM-sealed keys remain protected.

5. TPM for Device Encryption

The vulnerability of data leakage arising from device loss, non-approved access, or direct physical attacks has drastically heightened. The legacy software-oriented encryption solution may be highly secure for data stored in a static environment, but these methods can be susceptible to key extraction attacks, memory scraping, as well as boot-level attacks. The Trusted Platform Module (TPM) removes these issues by utilizing its so-called hardware-rooted key protection, along with maintaining rigorous integrity requirements prior to its release. This topic continues below for a detailed discussion about TPM, device-level encryption, how it functions, and its efficiency in countering direct as well as logical attacks.

5.1. Overview of TPM-Backed Device Encryption

TPM-backed encryption combines the trust inherent in the hardware roots to protect storage media through encryption. Compared to encrypting the keys in either the memory or the storage, keys are generated, stored, and released when the system is in a trusted state.

Benefits:

- Hardware isolation: Keys always remain in TPM memory and never leave it in plaintext form
- Integrity binding: The keys are bound to PCR values.
- Resistant to Physical Attacks: The keys cannot be compromised even if the device is stolen.
- Automatic unlocking: The encrypted volumes can be automatically unlocked without user interaction if integrity checks are passed.

Because of this balance of security and usability, TPM-protected encryption has become a ubiquitous capability of enterprise and consumer electronics.

5.2. TPM-Backed Key Protection Mechanisms

TPM enhances encryption security through several mechanisms:

5.2.1. Key Generation

The TPM generates cryptographic keys using its internal True Random Number Generator (TRNG). These keys are created within the TPM boundary and are never exposed externally.

5.2.2. Key Sealing

Keys are sealed to specific PCR values, ensuring they can only be unsealed when the system's boot state matches expected measurements.

5.2.3. Key Hierarchy Enforcement

The TPM's hierarchical key model ensures that:

- Compromise of one key does not affect others
- Keys cannot be exported or cloned
- Keys remain bound to the physical device

5.2.4. Authorization Policies

TPM 2.0 supports flexible policies that combine:

- PCR conditions
- Passwords
- HMAC authorization
- Locality restrictions
- Time-based constraints

These policies allow fine-grained control over when and how encryption keys are released.

5.3. Disk Encryption Architectures Using TPM

Several widely used disk encryption technologies like dm-crypt and Truecrypt integrate TPM for enhanced security.

5.3.1. Microsoft BitLocker

BitLocker is one of the most prominent examples of TPM-backed encryption. It uses TPM to:

- Protect the Volume Master Key (VMK)
- Validate boot integrity before unlocking the drive
- Enforce recovery mechanisms when PCR values mismatch

BitLocker Operational Flow:

1. TPM measures boot components and extends PCRs.
2. If PCR values match expected baselines, TPM releases the VMK.
3. The VMK decrypts the Full Volume Encryption Key (FVEK).
4. The FVEK decrypts the disk sectors.

If any boot component is modified, BitLocker enters recovery mode, requiring a recovery key.

5.3.2. Linux LUKS2 with TPM2 Integration

Modern Linux distributions support TPM-based unlocking through:

- systemd-cryptenroll
- tpm2-tools
- Clevis + Tang frameworks

LUKS2 TPM Workflow

- The disk encryption key is sealed to TPM PCRs
- During boot, TPM unseals the key only if integrity checks pass
- The OS automatically decrypts the root file system

This eliminates the need for manual passphrase entry while maintaining strong security guarantees.

5.4 Protection against Physical Attacks

TPM-backed encryption significantly strengthens defenses against physical attacks that target encryption keys or attempt to bypass authentication mechanisms.

5.4.1. Cold-Boot Attacks

Cold-boot attacks exploit residual data in RAM to extract encryption keys. TPM mitigates this by:

- Never storing keys in RAM in plaintext
- Releasing keys only after integrity checks
- Binding keys to PCR values

Even if RAM is cloned or frozen, TPM-sealed keys remain inaccessible.

5.4.2. DMA Attacks

Direct Memory Access (DMA) attacks—using interfaces like Thunderbolt or PCIe—can extract keys from memory. TPM prevents this by ensuring:

- Keys are not stored in system memory
- Keys are unsealed only after secure boot validation

Combined with IOMMU protections, TPM significantly reduces DMA attack surfaces.

5.4.3. Evil-Maid Attacks

In an evil-maid attack, an adversary with temporary physical access attempts to modify boot components to capture passwords or keys.

TPM mitigates this by:

- Detecting bootloader or firmware modifications
- Preventing key release when PCR values mismatch
- Triggering recovery workflows

This makes TPM-backed encryption particularly valuable for laptops and mobile devices.

5.4.4. Disk Cloning and Offline Attacks

If an attacker clones the storage device:

- TPM-sealed keys cannot be transferred
- The cloned disk cannot be decrypted
- PCR-bound keys prevent unlocking on another system

This provides strong protection against offline brute-force attempts.

5.5. TPM and Enterprise Data Protection

Enterprises rely heavily on TPM for endpoint security:

- Automatic encryption enforcement
- Remote attestation for compliance
- Credential protection
- Secure key escrow and recovery

TPM-backed encryption is now a baseline requirement in many regulatory frameworks, including government security baselines and corporate compliance policies.

5.6. Limitations of TPM-Backed Encryption

While TPM significantly enhances encryption security, it is not without limitations:

- PCR brittleness: Minor system changes can trigger recovery modes
- Firmware TPM vulnerabilities: fTPMs may be susceptible to firmware attacks.
- Supply-chain risks: Compromised hardware can undermine TPM trust.
- Usability challenges: Recovery workflows can be complex for end users.

These limitations highlight the need for careful policy design and robust implementation practices.

6. TPM in Identity, Authentication, and Credential Protection

Although the TPM is well-acknowledged regarding its contributions in secure booting as well as device encryption, its contributions in the context of identity management and protection of credentials cannot be ignored. In the current scenario, the reliance on cryptography identities, which include certificates, private keys, authentication tokens, as well as passwords, is extensive in establishing trust among various stakeholders, including users, devices, as well as services. In this case, these high-value assets will often become the prime target of an attack, through memory weaknesses, phishing, malware, as well as direct physical access. TPM creates a hardware root of trust in protecting such high-value assets.

In this section, we'll discuss how TPM improves the assurance of identity, the authentication process, and the safeguarding of credentials in an operating system environment and cloud-connected platforms.

6.1. TPM as a Hardware Root of Identity

The core component in TPM identity protection is the Endorsement Key (EK), which is an asymmetric key pair uniquely provided by the device's manufacturer and integrated in the TPM during production. The EK is recognized as the identity that binds the device and allows:

- Device attestation
- Secure provisioning
- Certificate enrolment
- Trusted device onboarding

The EK is usually packed together with an Endorsement Certificate that has been issued by the TPM vendor, linking the EK to the authorized hardware component. This certificate enables the remote services to ensure that they connect with an actual TPM and not an emulator.

6.2 .TPM-Backed Virtual Smart Cards

One of the most impactful applications of TPM in authentication is the Virtual Smart Card (VSC). Traditional smart cards require dedicated hardware tokens, card readers, and administrative overhead. TPM-based VSCs emulate the functionality of physical smart cards using TPM-protected keys.

Benefits of Virtual Smart Cards:

- No external hardware required
- Private keys stored securely inside TPM
- Resistant to key extraction and cloning
- Seamless integration with enterprise PKI
- Supports multi-factor authentication

Use Cases:

- Certificate-based login

- VPN authentication
- Secure email (S/MIME)
- Code signing

VSCs have become a standard feature in enterprise Windows environments, significantly reducing the cost and complexity of smart card deployments.

6.3. Credential Guard and Secure Authentication on Windows

Microsoft Windows integrates TPM deeply into its authentication architecture through features such as:

Credential Guard:

- Uses virtualization-based security (VBS) and TPM to isolate authentication secrets
- Protects NTLM hashes, Kerberos tickets, and domain credentials
- Prevents credential theft attacks such as Pass-the-Hash and Pass-the-Ticket

Windows Hello for Business:

- Uses TPM to store private keys for biometric authentication
- Eliminates password-based login
- Supports asymmetric key-based authentication to Azure AD and Active Directory

Secure Boot + TPM:

- Ensures that authentication secrets are only released when the system is in a trusted state

These mechanisms collectively strengthen endpoint authentication and reduce reliance on vulnerable password-based systems.

6.4. TPM for SSH Key Protection

SSH keys are widely used for secure remote access in enterprise, cloud, and DevOps environments. However, private keys stored in files are vulnerable to theft through malware, misconfiguration, or insider attacks.

TPM mitigates these risks by:

- Generating SSH private keys inside the TPM
- Preventing export of private keys
- Requiring TPM authorization policies for key usage
- Binding key usage to PCR values

Linux distributions increasingly support TPM-backed SSH keys through tools such as:

- tpm2-pkcs11
- tpm2-tss-engine
- OpenSSH TPM integration patches

This approach significantly reduces the attack surface for credential theft.

6.5. TPM for Password and Secret Protection

Beyond asymmetric keys, TPM can protect a variety of sensitive secrets:

- Password vault master keys
- OAuth tokens
- API keys
- Disk encryption passphrases
- Application secrets

Applications can use TPM to seal secrets to specific system states, ensuring that:

- Secrets are only accessible when the system is trusted
- Malware cannot extract secrets even with elevated privileges
- Secrets remain protected during offline attacks

This makes TPM a valuable component in secure application design.

6.6. TPM in Enterprise Identity and Access Management (IAM)

Enterprises increasingly rely on TPM for device identity and authentication workflows.

Device Identity

TPM provides a hardware-anchored identity for:

- Zero-trust network access
- Device compliance checks
- Secure enrollment into MDM/EMM platforms
- Certificate-based device authentication

Multi-Factor Authentication (MFA)

TPM strengthens MFA by:

- Protecting private keys used in authentication
- Enforcing biometric or PIN-based unlock
- Preventing credential replay attacks

Remote Attestation

Enterprises use TPM attestation to verify:

- Device health
- Secure boot enforcement
- Patch and configuration compliance

This is particularly important in regulated industries such as finance, healthcare, and government.

6.7. TPM in Cloud and Federated Identity Systems

Cloud platforms increasingly integrate TPM for secure identity management.

Azure AD and TPM

- TPM-backed keys used for device registration
- Hardware-rooted authentication for cloud services
- Conditional access policies based on TPM attestation

Google Cloud and vTPM

- Virtual TPMs provide identity for cloud VMs
- Used for secure boot and attestation in cloud workloads

Federated Identity

TPM strengthens federated identity protocols such as:

- SAML
- OAuth 2.0
- OpenID Connect

By protecting tokens and private keys from theft.

7. Security Analysis

A discussion on Trusted Platform Modules would be incomplete without mentioning the TrustZone, which is considered the foundation of hardware root-of-trust security and provides high assurance in key protection, platform integrity, and platform attestation. Nevertheless, as with all other security technologies, the TrustZone in the TPM module is not free from the challenges faced by all security technologies, i.e., it is not invulnerable to its own limitation and implementation challenges. This section would present an in-depth discussion on the security posture and resilience of the TrustZone module in the TPM.

7.1. Strengths of TPM as a Hardware Root of Trust

TPM's design incorporates several architectural principles that collectively provide strong security guarantees.

7.1.1. Hardware-Based Isolation

TPM isolates cryptographic operations from the host CPU and memory. Private keys never leave the TPM boundary, significantly reducing exposure to:

- Memory scraping
- Kernel-level malware
- Privilege escalation attacks

- DMA-based key extraction

This isolation is one of TPM's most powerful defenses.

7.1.2. Tamper Resistance
Discrete TPMs incorporate physical protections such as:
- Shielded enclosures
- Tamper-evident circuitry
- Voltage, temperature, and glitch detection

These features make physical extraction of secrets extremely difficult.

7.1.3. Integrity Measurement and Attestation
PCR-based measurements provide:
- A cryptographic record of the boot process
- Detection of unauthorized modifications
- Remote attestation capabilities

This enables systems to prove their integrity to external verifiers.

7.1.4. Strong Cryptographic Foundations
TPM 2.0 supports modern algorithms:
- RSA-2048/3072
- ECC (NIST P-256, P-384)
- SHA-256 and SHA-384
- HMAC-based authorization

Algorithm agility ensures long-term cryptographic resilience [22].

7.1.5. Secure Key Hierarchies
TPM enforces strict key hierarchies that prevent:
- Key export
- Unauthorized duplication
- Cross-device cloning

This is essential for device identity and encryption.

7.2. Threat Model
A realistic threat model for TPM considers adversaries with varying capabilities:

7.2.1. Software-Level Attackers
Attackers with malware, rootkits, or kernel-level access attempt to extract keys or bypass integrity checks.

7.2.2. Physical Attackers
Adversaries with physical access may attempt:
- Cold-boot attacks
- Bus probing
- Chip decapsulation
- Side-channel analysis

7.2.3. Supply-Chain Attackers
Compromised manufacturing or firmware updates can undermine TPM trust.

7.2.4. Insider Threats
Administrators or technicians with privileged access may attempt unauthorized key extraction or attestation manipulation. TPM is designed to resist most of these threats, but certain attack vectors remain challenging.

7.3. Known Vulnerabilities and Weaknesses

Despite its strengths, TPM is not invulnerable. Several categories of weaknesses have been identified in research and real-world deployments.

7.3.1. Firmware TPM (fTPM) Vulnerabilities

FTPMs implemented in system firmware (e.g., Intel PTT, AMD fTPM) lack the physical tamper resistance of discrete TPMs. Vulnerabilities include:

- Firmware rollback attacks
- Exploitable bugs in TPM firmware
- Side-channel leakage through shared CPU resources
- Susceptibility to DMA-based manipulation

For example, certain AMD fTPM implementations have exhibited latency and reliability issues due to firmware-level design flaws.

7.3.2. Side-Channel Attacks

Researchers have demonstrated that TPMs may be susceptible to:

- Power analysis
- Electromagnetic (EM) leakage
- Timing attacks

While discrete TPMs incorporate countermeasures, sophisticated adversaries may still extract information under laboratory conditions.

7.3.3. Supply-Chain Attacks

TPM security depends heavily on:

- Manufacturer integrity
- Secure provisioning of Endorsement Keys
- Authenticity of firmware updates

If the supply chain is compromised, attackers may introduce backdoors or malicious firmware.

7.3.4. PCR Brittleness and Policy Complexity

PCR-based authorization can be fragile:

- Minor system updates may change PCR values
- Misconfigured policies can lock users out
- Complex PCR dependencies increase operational risk

This brittleness can lead to usability issues and recovery challenges.

7.3.5. Vulnerabilities in TPM Software Stacks

The TPM 2.0 Software Stack (TSS) and OS-level integrations may contain vulnerabilities unrelated to TPM hardware. Examples include:

- Buffer overflows in TSS libraries
- Incorrect PCR handling in bootloaders
- Misconfigured authorization policies

These software-level issues can undermine TPM security if not properly managed.

8. Challenges and Limitations

The widespread use and strong security guarantees notwithstanding, there are indeed challenges relating to the TPM. While TPM provides a robust foundation for hardware rooted trust, its effectiveness heavily relies on correct implementation, secure supply chains, proper configuration, and careful integration with operating systems and applications. This section will explore practical, architectural, and operational limits of TPM deployment, pointing out those areas where TPM may fail or require complementary security mechanisms.

8.1. Hardware and Manufacturing Constraints

8.1.1. Cost and Form-Factor Limitations

Although TPM chips are relatively inexpensive, integrating them into low-cost IoT devices or ultra-compact embedded systems can be challenging. Manufacturers may opt for firmware TPMs (fTPMs) to reduce cost, sacrificing some tamper-resistance.

8.1.2. Supply-Chain Vulnerabilities

TPM security assumes a trustworthy manufacturing process. However, supply-chain attacks—such as malicious firmware injection, key extraction during production, or counterfeit chips—pose significant risks. If the Endorsement Key (EK) is compromised at the factory, the entire trust model collapses.

8.1.3. Hardware Aging and Reliability

Like any silicon component, TPMs are subject to hardware degradation. Failures can lead to:

- Loss of sealed keys
- Inability to boot encrypted systems
- Permanent device lockout

This is particularly problematic in long-lifecycle industrial or medical devices.

8.2. Firmware TPM (fTPM) Limitations

Firmware TPMs, implemented in system firmware rather than discrete chips, offer reduced physical security. They rely on the CPU and system memory, making them vulnerable to:

- Firmware rollback attacks
- Exploitable bugs in system firmware
- Side-channel leakage through shared resources
- DMA-based manipulation

While fTPMs are convenient and cost-effective, they do not provide the same level of tamper resistance as discrete TPMs.

8.3. PCR Brittleness and Policy Complexity

Platform Configuration Registers (PCRs) are powerful but fragile.

8.3.1. Sensitivity to System Changes

Minor changes such as firmware updates, driver modifications, or bootloader patches—can alter PCR values. This may cause:

- Disk encryption recovery prompts
- Failed attestation
- Inability to unseal keys

9. Conclusion

This paper highlights the importance of the Trusted Platform Module (TPM) as a crucial technology for establishing hardware-based trust in modern computing systems. It improves security by providing effective key management, integrity checks, device identification, and protection of credentials, which helps make platforms more resistant to both physical and digital threats across different environments, such as cloud, mobile, IoT, and critical safety areas. The analysis of TPM's architecture, its relationship with secure and measured boot processes, and its role in device encryption emphasizes that it is more than just an extra security feature.

Current case studies proved that the TPM is secure, effective and more easily adaptable across various hardware security solutions, including enterprise endpoints, research, government control systems and in cloud platforms. While there are challenges such as PCR brittleness, firmware vulnerabilities, supply chain risks and problems with large-scale deployment, these problems can be managed through governance and constant monitoring. Looking ahead, TPM is likely to adapt as new technologies like post-quantum cryptography, confidential computing, and expansive IoT ecosystems emerge, altering security requirements. Future TPM designs are expected to incorporate post-quantum algorithms, enhanced virtualization features, improved developer tools, and more flexible attestation models. Overall, TPM is a well-established and forward-looking element of trusted computing, with its hardware-based trust model and expanding capabilities making it crucial for building secure, resilient, and trustworthy digital infrastructures.

References

[1] Trusted Computing Group. TCG Specification Architecture Overview Revision 1.4. 2007. Available online: [Online]. https://trustedcomputinggroup.org/wp-content/uploads/TCG_1_4_Architecture_Overview.pdf (accessed on 1 April 2024).

[2] Sailer, R.; Zhang, X.; Jaeger, T.; van Doorn, L. Design and Implementation of a TCG-based Integrity Measurement Architecture. In Proceedings of the 13th USENIX Security Symposium (USENIX Security 04), San Diego, CA, USA, 9–13 August 2004; pp. 223–238.
[3] Microsoft. Secure the Windows boot process- Windows security.
[Online].
https://docs.microsoft.com/en-us/windows/security/information-protection/
secure-the-windows-10-boot-process, December 2021.
[4] Intel Corporation. Hardening Intel® Trusted Execution Technology and Intel® Boot Guard. Security White Paper, Rev. 0.21, 2024. Document No. 824411.
[5] Arm Ltd., TrustZone Technology for the Armv8-A Architecture, Version 1.0, 2020.
[6] Arm Ltd., TEE Reference Documentation: TrustZone for Cortex-A, 2023.
[7] NXP Semiconductors, ARM® TrustZone®: How to Use It to Make Devices Secure and Safe, 2019.
[8] V. Costan and S. Devadas, "Intel SGX Explained," IACR Cryptology ePrint Archive, pp. 1–118, 2016.
[9] J. Lind, J. Kwon, D. Eyers, and R. Vitenberg, "EnclaveDB: A Secure Database Using SGX," in Proc. IEEE S&P, 2019, pp. 264–278.
[10] Intel Corporation, "Intel® Software Guard Extensions (Intel® SGX) Developer Guide," Intel White Paper, 2020.
[11] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Silicon Physical Random Functions," in Proc. ACM CCS, 2002, pp. 148–160.
[12] R. Maes, "Physically Unclonable Functions: Constructions, Properties and Applications," Proc. IEEE, vol. 103, no. 10, pp. 1868–1881, 2015.
[13] Sadeghi and D. Naccache (eds.), Towards Hardware-Intrinsic Security: Foundations and Practice, Springer, 2010.
[14] J. Butterworth, C. Kallenberg, and X. Kovah, "BIOS Chronomancy: Fixing the Core Root of Trust for Measurement," in Proc. IEEE Security & Privacy Workshops, 2014, pp. 1–12.
[15] AMD, "AMD-SB-4011: TPM 2.0 Reference Implementation Vulnerability," AMD Security Bulletin, 2023.
[16] Y. Zhang et al., "Firmware Attacks and Defenses: A Survey," ACM Computing Surveys, vol. 54, no. 5, pp. 1–36, 2021.
[17] ISO/IEC 11889-1:2015, Information Technology — Trusted Platform Module Library — Part 1: Architecture, 2nd ed., 2015.
[18] C. Shepherd and K. Markantonakis, "Building Execution Environments from the Trusted Platform Module," in Trusted Execution Environments, Springer, 2024, pp. 79–95.
[19] C. Ryu et al., "A Comprehensive Survey of TPM for Defense Systems," KSII Transactions on Internet and Information Systems, vol. 18, no. 7, pp. 1–20, 2024.
[20] R. Sailer, X. Zhang, T. Jaeger, and L. van Doorn, "Design and Implementation of a TCG-Based Integrity Measurement Architecture," in Proc. USENIX Security Symposium, 2004, pp. 223–238.
[21] S. Delaune, S. Kremer, M. D. Ryan, and G. Steel, "Formal Analysis of Protocols Based on TPM State Registers," Journal of Computer Security, vol. 19, no. 5, pp. 1029–1069, 2011.
[22] National Institute of Standards and Technology, "Security Requirements for Cryptographic Modules,"
[Online]. http://csrc.nist.gov/publications/fips/fips140-2/ fips1402.pdf, accessed on April 15, 2014.