

Hardware Based Secure Boot and Trusted Execution Environments

Tej Pratap Sinha¹, Sukhdev Vashu², Neeraj Chopra³

Associate Professor, Assistant Professor

Department of Computer Engineering

Horizon College of Engineering, Jaipur, India

Tejpratap18s@rediffmail.com¹, sukhdev0vashu@gmail.com², neerachopradk@yahoo.com³

Abstract

The growing reliance on embedded and IoT devices in critical applications has amplified the need for hardware-based security mechanisms. Hardware-Based Secure Boot (HBSB) and Trusted Execution Environments (TEEs) are essential technologies that ensure the integrity and confidentiality of software execution from system initialization to runtime. Secure Boot verifies the authenticity of firmware and operating systems during startup, preventing unauthorized code execution, while TEEs provide an isolated environment for sensitive computations and cryptographic operations. This paper provides a comprehensive review of hardware-based secure boot mechanisms and TEEs, exploring their architecture, implementation, advantages, challenges, and applications. The discussion includes comparative analyses, current research trends, and future directions for integrating these technologies in embedded systems, IoT devices, and edge computing platforms.

Keywords: *Hardware Security, Secure Boot, Trusted Execution Environment, Embedded Systems, IoT Security, TPM, ARM TrustZone*

1. INTRODUCTION

In the modern era, embedded systems and IoT devices are increasingly deployed in critical applications such as healthcare, automotive, industrial automation, and financial systems. These devices are often exposed to potential security threats including firmware attacks, malware injections, and unauthorized software modifications. Traditional software-based security measures are insufficient for protecting sensitive operations because they can be

bypassed by low-level attacks. Hardware-based mechanisms, such as Secure Boot and Trusted Execution Environments, offer enhanced security by anchoring trust at the hardware level.

Hardware-Based Secure Boot ensures that the system boots only trusted firmware and operating systems. This process involves verifying digital signatures and cryptographic hashes during initialization. TEEs, on the other hand, provide isolated execution environments that safeguard sensitive data and computation from potentially compromised operating systems or applications. Together, these mechanisms form a foundation for end-to-end hardware security in modern computing platforms.

2. HARDWARE-BASED SECURE BOOT

Hardware-Based Secure Boot is the foundation of trust in modern embedded and computing platforms. It ensures that from the very first instruction executed by the processor, only authenticated and untampered software is allowed to run. Unlike software security mechanisms that operate after the operating system is loaded, secure boot begins at the hardware level, making it extremely difficult for attackers to bypass.

Secure boot is widely used in microcontrollers, SoCs, automotive ECUs, IoT gateways, smartphones, and industrial controllers where firmware attacks can cause serious damage or safety issues.

2.1 Definition and Objectives

Secure Boot can be defined as a **hardware-enforced chain of trust** that validates every piece of code during the boot sequence using cryptographic techniques before execution is allowed.

The important idea is: **“Nothing runs unless it is verified.”**

The primary objectives are explained below in more depth:

Integrity Verification

Integrity verification ensures that the firmware, bootloader, and OS image stored in flash memory have not been modified after manufacturing or deployment.

- Cryptographic hash functions (SHA-256/SHA-3) are used to compute a digest of the firmware.
- This digest is compared with a trusted reference value stored securely in hardware.
- Even a single-bit change in the firmware will produce a completely different hash, allowing detection of tampering.

This prevents:

- Firmware modification by attackers
- Malware injection into boot code
- Corruption during updates

Authentication

Authentication ensures that the firmware originates from a **trusted vendor or developer**.

- Digital signatures based on RSA/ECC are attached to firmware images.
- Public keys are permanently stored in hardware (ROM, eFuse, OTP memory).
- During boot, the signature is verified using the stored public key.

This ensures:

- Only vendor-approved firmware runs
- Unauthorized third-party code cannot be executed
- Secure firmware updates in the field

Early Detection of Attacks

Boot-level malware and rootkits are extremely dangerous because they load before the OS and remain hidden.

Secure boot stops such attacks at the earliest stage by:

- Halting the boot process if verification fails
- Triggering recovery mode or safe firmware
- Logging failure events for forensic analysis

This is why secure boot is sometimes called **first line of defense** in hardware security.

2.2 Architecture of Secure Boot

The architecture of hardware-based secure boot follows a **Chain of Trust model**, where each stage verifies the next stage before passing control.

Stage 1: Root of Trust (RoT)

The Root of Trust is the most critical part of the system.

- Stored in immutable memory (ROM, eFuse, OTP)
- Cannot be modified after manufacturing
- Contains:
 - Public cryptographic keys
 - Hash of the first bootloader
 - Minimal verification code

When power is applied, the processor always begins execution from RoT.

Why RoT is important:

If RoT is compromised, the entire security collapses. Therefore, it is physically protected and immutable.

Stage 2: Bootloader Verification

The RoT verifies the **first-stage bootloader** stored in external flash.

Process:

1. RoT computes hash of bootloader.
2. Verifies digital signature using stored public key.
3. If valid → control passed to bootloader.
4. If invalid → boot stops or recovery mode.

The bootloader is responsible for initializing memory, clocks, and peripherals.

Stage 3: Firmware / OS Verification

The verified bootloader now continues the chain:

- Verifies second-stage bootloader or firmware
- Verifies operating system kernel
- Verifies device drivers and configuration

Each stage is signed and verified before execution. This creates a **continuous trust chain**.

Stage 4: Execution Authorization

Only after all verifications succeed:

- The operating system is allowed to load
- Applications begin execution
- Security policies are enforced

If any stage fails:

- System may enter safe mode
- Load a backup firmware
- Prevent further execution

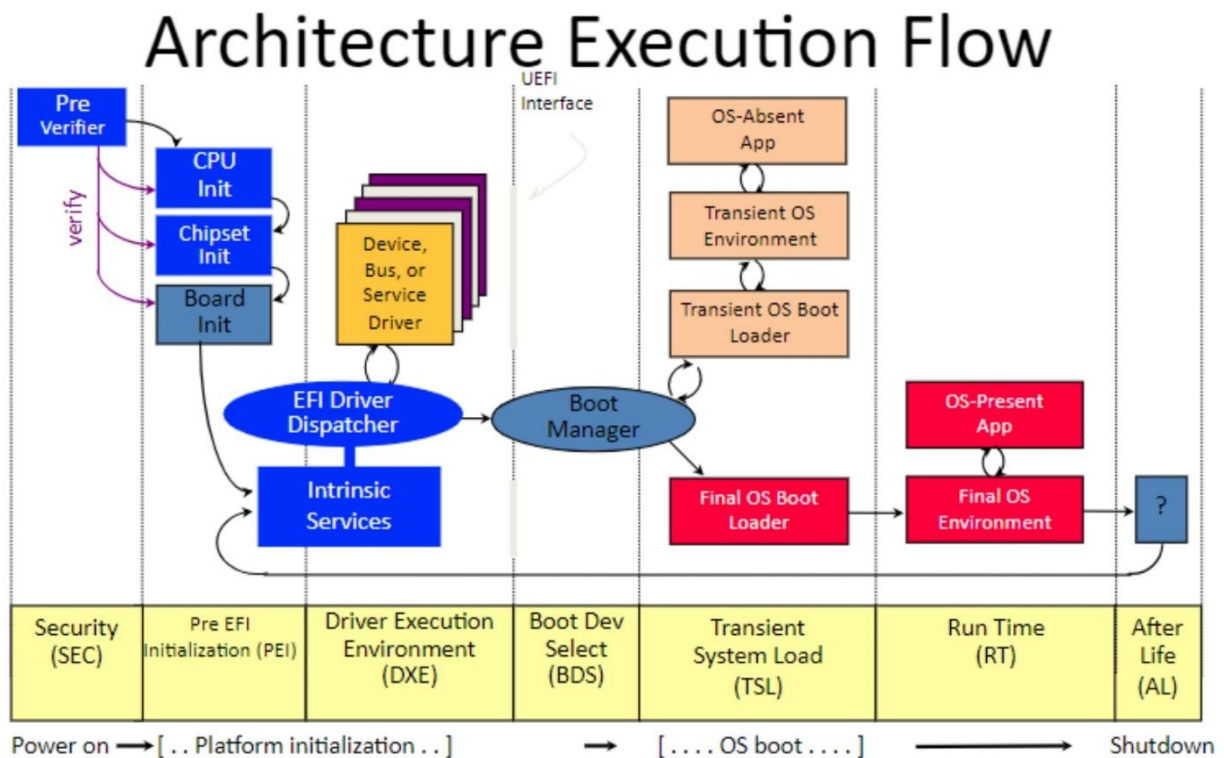


Figure 1: Secure Boot Process Flow

2.3 Cryptographic Techniques Used in Secure Boot

Cryptography is the core element that makes secure boot reliable. Without strong cryptographic primitives, the chain of trust cannot be enforced. Secure boot mainly depends on three major cryptographic components: **PKI for authentication**, **hashing for integrity**, and **secure hardware modules for key protection**.

Public Key Infrastructure (PKI)

PKI is used to verify that the firmware or bootloader is coming from a trusted source.

- Firmware images are digitally signed by the device manufacturer using a **private key**.
- The corresponding **public key** is permanently stored inside the hardware (ROM, eFuse, OTP).
- During boot, the digital signature attached to the firmware is verified using this public key.

If the signature matches:

→ Firmware is authentic and allowed to run.

If the signature fails:

→ Boot process is stopped.

Common algorithms used:

- RSA-2048 / RSA-4096

- ECC (Elliptic Curve Cryptography) such as ECDSA (preferred in embedded systems due to smaller key size and faster computation)

Why PKI is important:

Even if an attacker copies the firmware, they cannot modify it because they do not possess the private signing key.

Hash Functions (SHA-256, SHA-3)

Hash functions are used to check whether the content of firmware has been altered.

- A hash (digest) of the firmware is calculated during signing.
- During boot, the device recalculates the hash and compares it with the stored or signed value.
- Any change in firmware, even a single bit, changes the hash completely.

Properties of cryptographic hash functions used:

- One-way (cannot retrieve original data)
- Collision resistant
- Fast to compute in hardware

SHA-256 is most widely used, while SHA-3 is being adopted in newer secure systems.

Role in secure boot:

Hashing ensures **integrity**, while PKI ensures **authentication**.

Hardware Security Modules (HSMs) and Secure Key Storage

Keys are the most sensitive part of secure boot. If keys are exposed, the whole system security fails.

Hardware protection methods include:

- eFuse or OTP memory for storing public keys
- On-chip cryptographic accelerators
- Trusted Platform Module (TPM) or Secure Elements
- Physically unclonable functions (PUFs) in advanced chips

These modules:

- Prevent keys from being read or modified
- Perform cryptographic operations internally
- Resist physical attacks like probing or glitching

2.4 Implementation Examples in Real Platforms

Secure boot is not just theoretical; it is implemented in many commercial processors and microcontrollers.

Intel Boot Guard

- Provides hardware root of trust in Intel chipsets.
- Verifies BIOS/UEFI firmware before CPU executes it.
- Uses fused keys inside chipset for signature verification.
- Prevents unauthorized BIOS modification (common malware target).

Used in: PCs, servers, industrial computers.

ARM Trusted Boot with TrustZone

- ARM processors include secure ROM code that performs signature verification.
- TrustZone separates secure and non-secure worlds.
- Secure boot runs in the secure world before handing control to normal OS.
- Widely used in smartphones, IoT SoCs, and embedded boards.

Used in: STM32, NXP i.MX, Raspberry Pi (newer versions), mobile chipsets.

Trusted Platform Module (TPM)

- Dedicated chip for cryptographic measurements.
- Stores hashes of each boot stage in Platform Configuration Registers (PCRs).
- Allows remote attestation (proving system booted securely).
- Works along with secure boot to log and verify boot sequence.

Used in: Laptops, desktops, industrial gateways, automotive ECUs.

2.5 Advantages of Hardware-Based Secure Boot

Secure boot provides several important benefits for embedded and computing systems.

- **Prevents execution of untrusted software:** Only signed firmware can run.
- **Stops firmware-level malware:** Rootkits and bootkits are blocked before OS loads.
- **Enables secure firmware updates (OTA):** Devices can verify updates remotely.
- **Establishes hardware root of trust:** Forms base for TEEs and other security layers.
- **Regulatory compliance:** Required in automotive, medical, and industrial standards.

- **Improves device reliability in field deployments.**

2.6 Challenges in Secure Boot Implementation

Despite strong security, implementing secure boot has practical challenges.

Key Management and Revocation

- Devices in field may need key updates if keys are compromised.
- Revoking old keys and installing new keys securely is difficult.
- Requires secure provisioning during manufacturing.

Performance Overhead

- Cryptographic signature verification takes time during boot.
- Resource-constrained microcontrollers may experience longer boot times.
- Hardware accelerators are often required to reduce delay.

Legacy Hardware Limitations

- Older devices do not have ROM-based RoT or crypto engines.
- Software-only secure boot is weaker and easier to bypass.
- Retrofitting secure boot into legacy systems is costly.

3. TRUSTED EXECUTION ENVIRONMENTS (TEES)

3.1 Definition and Purpose

A Trusted Execution Environment (TEE) is an isolated execution environment that runs alongside a main operating system (REE – Rich Execution Environment). TEEs ensure confidentiality and integrity of sensitive code and data, protecting them from potentially compromised applications or malware.

3.2 Architecture

TEEs generally include the following components:

- **Secure OS/Monitor:** Manages secure applications and communication between REE and TEE.
- **Trusted Applications (TAs):** Sensitive software running within the TEE.

- **Hardware Isolation:** Memory and processor resources are protected using features like ARM TrustZone or Intel SGX.

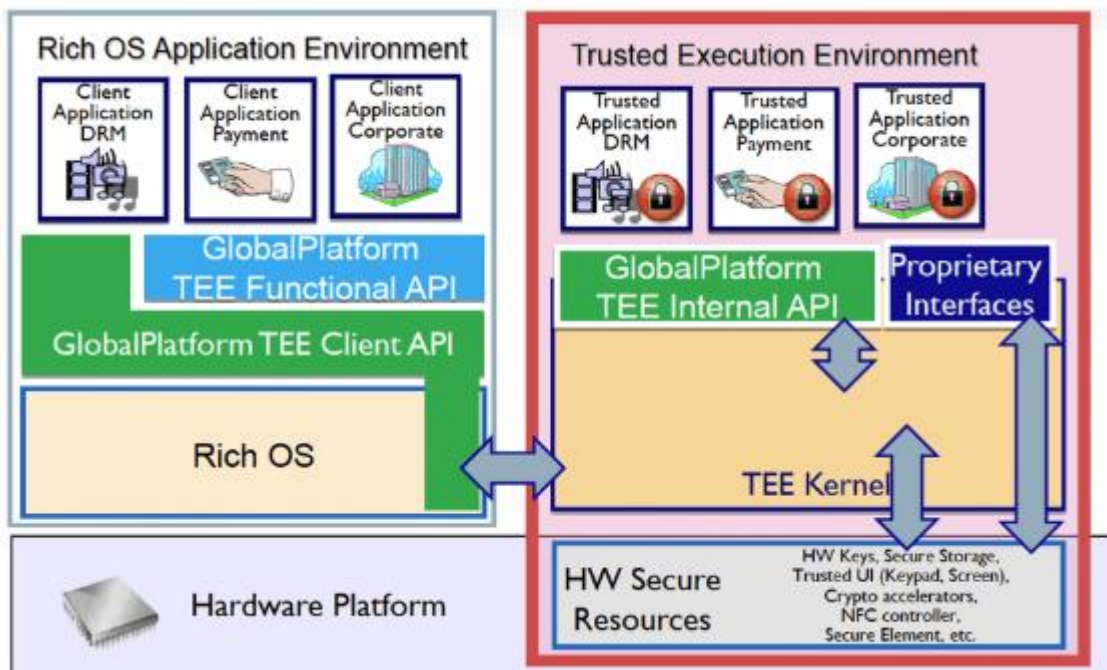


Figure 2: TEE Architecture

3.3 Key Technologies

1. **ARM TrustZone:** Divides CPU resources into secure and non-secure worlds.
2. **Intel Software Guard Extensions (SGX):** Creates protected enclaves for sensitive computation.
3. **RISC-V Keystone:** Open-source TEE implementation for RISC-V platforms.

3.4 Security Functions

- Secure key storage and cryptographic operations.
- Protection against memory and side-channel attacks.
- Isolation from the main OS to prevent privilege escalation.

3.5 Applications

- Mobile payment systems and banking apps.
- DRM (Digital Rights Management) for media content.
- IoT devices requiring sensitive data processing (e.g., medical sensors).

3.6 Challenges

- Vulnerability to side-channel attacks if hardware is poorly designed.

- Complex software development due to restricted APIs.
- Performance overheads in resource-constrained devices.

4. INTEGRATION OF SECURE BOOT AND TEEs

Hardware-based secure boot and TEEs complement each other by forming a layered security framework:

Feature	Secure Boot	TEE	Integration Benefit
Primary Role	Firmware validation	Runtime isolation	End-to-end protection from boot to runtime
Root of Trust	Hardware	Hardware-based	Unified key management
Security Target	Boot-time malware	Runtime attacks	Comprehensive attack surface coverage
Cryptography	Signature verification	Enclave keys, encryption	Mutual reinforcement for confidentiality and integrity

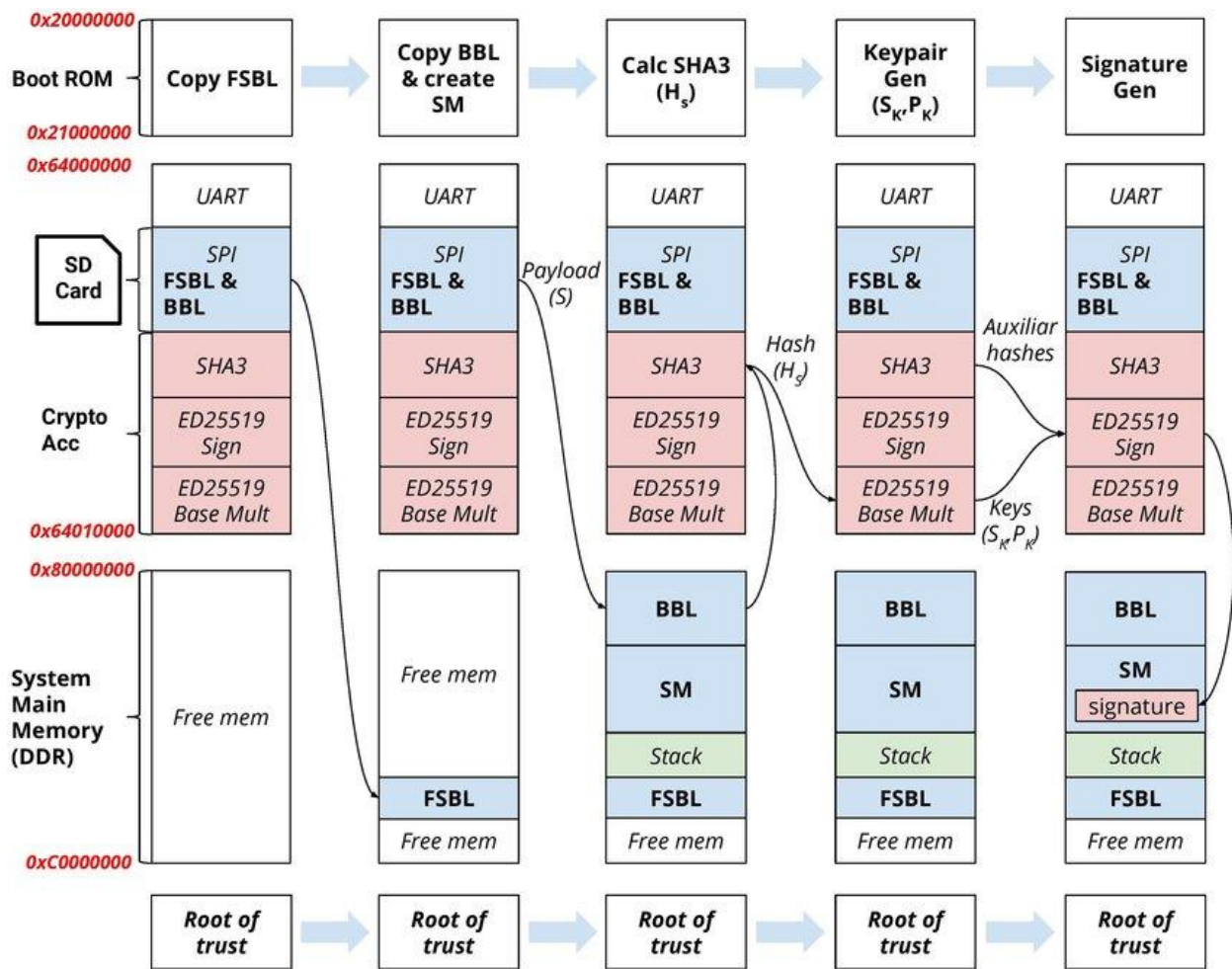


Figure 3: Combined Secure Boot and TEE Model

Integration ensures that only trusted firmware is loaded and that sensitive operations remain secure during runtime. This approach is increasingly important in automotive ECUs, industrial IoT devices, and edge AI platforms.

5. CASE STUDIES

5.1 IoT Devices

Secure boot and TEE mechanisms in IoT devices prevent firmware tampering, protecting connected sensors and actuators from remote attacks. For example, ARM TrustZone-based IoT gateways can securely manage cryptographic keys for multiple devices.

5.2 Automotive Systems

Modern vehicles contain multiple ECUs that control safety-critical operations. Secure boot prevents malicious firmware injection, and TEEs isolate sensitive operations such as encryption of vehicle-to-everything (V2X) communication data.

5.3 Mobile Devices

Smartphones rely heavily on TEEs for secure authentication (biometrics, payment processing). Devices implement secure boot to ensure that operating system updates are legitimate, reducing exposure to malware.

6. FUTURE DIRECTIONS

- **Lightweight TEEs for Edge Devices:** Development of energy-efficient TEEs for IoT devices with limited resources.
- **Post-Quantum Secure Boot:** Adapting secure boot mechanisms to resist quantum cryptographic attacks.
- **Hardware-Software Co-Design:** Unified frameworks for integrating secure boot, TEEs, and runtime monitoring in embedded systems.
- **Standardization Efforts:** Expanding open standards like GlobalPlatform TEE API for interoperability across devices.

7. CONCLUSION

Hardware-Based Secure Boot and Trusted Execution Environments are critical technologies for establishing trust in modern computing systems. Secure Boot protects the system from boot-time malware by verifying the authenticity of firmware, while TEEs provide isolated environments for sensitive computations, ensuring runtime security. The integration of both technologies delivers a robust hardware-rooted security framework suitable for IoT, embedded systems, automotive platforms, and mobile devices. Despite implementation challenges, advancements in hardware design, cryptography, and standardization are expanding the applicability of these mechanisms. Future research will focus on lightweight, interoperable, and post-quantum-secure implementations to address emerging threats in an increasingly connected world.

REFERENCES

1. ARM Ltd., *ARM Security Technology – Building a Secure System using TrustZone*, 2022.

2. Intel Corporation, *Intel Boot Guard and Trusted Execution Technology Overview*, 2021.
3. GlobalPlatform, *TEE System Architecture Specification*, 2023.
4. R. Mehra, S. Kaur, *Embedded System Security with Hardware-Based Trusted Execution*, Journal of Embedded Computing, 2022.
5. D. Lie et al., *Architectural Support for Secure Execution in Commodity Processors*, ACM SIGOPS, 2006.
6. Y. Yarom, K. Falkner, *Cache Attacks and Side-Channel Exploits in TEEs*, IEEE Security & Privacy, 2018.
7. S. Ravi et al., *Security in Embedded Systems: Design and Implementation*, Springer, 2020.
8. NIST, *Recommendation for Hardware Root of Trust in Embedded Systems*, Special Publication 800-193, 2021.
9. M. Ambrosin et al., *Post-Quantum Secure Boot: Challenges and Opportunities*, IEEE Transactions on Computers, 2023.
10. A. Roy, R. Mehra, *Hardware-Software Co-Design for Trusted Execution in IoT Devices*, International Journal of Computer Applications, 2021.