Hardware Root-of-Trust implementations in Trusted Execution Environments

Usman Ali[1], Hamza Omar[1], Chujiao Ma[2], Vaibhav Garg[2] and Omer Khan[1]

[1] University of Connecticut, CT
[2] Comcast, PA

Abstract. Hardware-based Root of Trust (HRT) is considered the gold standard for bootstrapping trust in secure computing. This paper analyzes HRT implementations across state-of-the-art TEEs and differentiates HRT implementation across two dimensions: 1) Security Properties & Threats and 2) Hardware Capabilities. Later, this work analyzes and compares 1) Intel SGX, 2) ARM TrustZone, 3) NXP Trust Architecture, 4) AMD SEV, 5) Microsoft Pluton, and 6) Apple T2 HRTs in terms of threats, security properties, and capabilities.

Keywords: Hardware Root-of-Trust · Trusted Computing · Secure Hardware

1 Introduction

Trustworthy computing is recognized as an open challenge in cybersecurity. Trusted Execution Environments (TEEs) are secure execution environments where applications can be executed securely irrespective of the rest of the system. However, TEEs are complex designs consisting of software and hardware component(s) and provide an enriched execution environment. A core technical constraint of TEE is to guarantee trust, i.e., to make computing behave as per user expectations. Significant support has emerged for hardware-based root-of-trust (HRT) implementation to anchor trust in secure computing. HRT is perceived to be more secure compared to the software-only counterpart. Individual TEEs are designed to cater a distinct threat models and consequently have differing implementations of HRTs and software component(s). For example, Apple T2 [6] guarantees trust in mobile and laptops, whereas Microsoft Pluton[5] protects gaming consoles against physical attacks. The appropriate capabilities of an HRT are determined by the applicable attack scenarios and corresponding mitigation strategies. A detailed study on understanding of HRT implementation is missing in literature.

A hardware root-of-trust may be implemented in a variety of ways, even for the the same attack model. For example, a system may implement the hardware root-of-trust as a separate hardware module or fuse it inside the processor chip itself. The core metrics for analyzing these secure systems vary from one design to the other, hence HRT implementations. One design may consider a large set of attacks and introduce high hardware intrusiveness compare to software components. In comparison, a different design may consider a high software intrusiveness against a larger subset of attacks and have much lower hardware complexity.

This paper analyze hardware-based root-of-trust implementation across state-of-the-art trusted execution environments and compares distinct HRT implementations regarding security guarantees, and threat-prevention assurances. We illustrate the usefulness of this work by analyzing and comparing HRT implementations in 1) Intel SGX, 2) ARM TrustZone, 3) NXP Trust Architecture, 4) AMD SEV, 5) Microsoft Pluton, and 6) Apple T2.

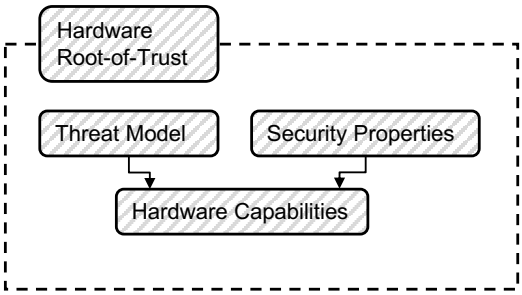

Figure 1: The design process of HRTs (1) identification of threats, (2) Security properties that formally guarantee protection against threats, and (3) hardware capabilities that implement security properties.

We begin by defining HRTs based on the understanding of industry and security groups. The next section describes a detailed background about threats to secure systems & formal security properties that guarantee protection against them, then the capabilities that implement security guarantees to protect against those threats. Later, we present a case-study analysis and comparison of six state-of-the-art HRT implementations.

2 Hardware Root-of-Trust

The terminology *Hardware Root-of-Trust* (HRT) is often used as the foundational block by vendors of different security schemes, protocols, products, or services within secure computing, but every designer refers to a different object using this terminology. For example, Intel [3] defines HRT as cryptographic keys protected by hardware and inseparable ID from hardware. Dell [2] defines it as an immutable hardware chip that is used to attest the integrity of the BIOS code. Once BIOS is verified, this trusted BIOS is used to start chain-of-trust in the Dell systems. NXP [4] refers to it's SoC solution *QorIQ Layerscape Processing Platforms Trust Architecture 3.0* as a HRT. In contrast to industry, NIST [9] refers to Trusted Computing Group (TCG) defined Trusted Platform Modules (TPM) as HRT in one of their Special Publications 800-164. TCG [10] defines TPM as a computer chip (microcontroller) that can securely store artifacts used to authenticate the system. Considering these various definitions of HRTs, we unify a HRT definition in this paper as:

Definition 1. Hardware-based Root-of-Trust (HRT) is an immutable hardware component or a set of hardware components (e.g., an encryption engine and/or a dedicated secure processor) considered unconditionally trusted against a well-defined threat model.

HRT is sometimes confused with Trusted Execution Environment (TEE). TEE is a comprehensive, secure execution environment that consists of software and hardware component(s) and protects against a well-defined threat model. In contrast, HRT is a hardware component(s) that used in secure computing to anchor trust against a specific threat model. An HRT is often used within TEE to enforce authentication, encryption, and isolation.

3 Background

To understand and compare distinct HRTs, a breakdown of HRT across two dimensions: 1) Security Properties & Threats, and 2) Capabilities. This breakdown is used in the second part of framework to compare distinct HRTs in terms of security guarantees against threat vectors, performance implications of different capabilities in HRTs, and implementation cost of these capabilities in HRTs. Figure 1 presents the abstract view of the framework.

3.1 Framework Components

The framework has two components. The first comprises the threats to HRTs as well as the corresponding security properties they target. The second consists of the capabilities that provide defenses against those threats and thus assure those same security properties. In this section we outline the key security properties, threats, and capabilities.

3.1.1 Threats & Security Properties

HRTs are used to bootstrap trust in secure computing and protect against various threats to critical assets. In this section, the framework explains the different types of available threats to HRT. After explaining threats to HRTs, this section discusses security properties that guarantee protection against these threats. This framework broadly categorizes threats into physical threats and software threats. The framework discusses traditional physical threats, software threats, and side-channel threats within these categories. The majority of existing HRTs fail to protect against both physical and software side-channel threats.

Physical threats are a set of attacks in which the attacker has physical access to hardware resources of the critical assets. This framework further breaks down physical attacks into probe attacks and physical side-channel attacks. Probe attacks are passive attacks that allow attackers to access critical assets' secrets by physically probing exposed wires or pins on interconnects or DRAM / Processor chips, respectively. Physical side-channel threats are passive attacks where the attackers observe power, heat, or electromagnetic leakages of the HRT. They then infer information on critical assets using these observations.

Software threats are a set of attacks in which the attacker co-exists alongside critical assets on the hardware resources. This framework categorizes software threats into application-level attacks, system-level attacks, and software side-channel attacks. Application-level malware (e.g.,. Mirai malware) usually exploits zero-day bugs to compromise critical assets. Contrary, system-level threats involve malicious operating systems or firmware (e.g., root-kits). A malicious OS or firmware disables all enforced security capabilities. Software side-channel threats are emerging practical class of attacks (e.g., Spectre & Meltdown) that exploit underlying hardware to break security guarantees.

Security Properties HRTs protect against threats by enforcing fundamental security properties (i.e., confidentiality and integrity). TEEs enforce these properties with the help of hardware/software capabilities. Confidentiality provides strong guarantees for critical assets from getting accessed by unauthorized malicious entities. An HRT usually satisfies confidentiality by enforcing isolation or sealing (e.g., encryption) for critical assets. Integrity guarantees an HRT's originality, and an HRT can guarantee integrity by attesting (a proof of authenticity) critical assets on any given instance.

3.1.2 Capabilities

TEEs enforce security properties with the help of HRTs or a combination of HRT/software capabilities that protect against threats.

Hardware Isolation A TEE usually includes an execution engine that operates on critical assets. The HRT execution capabilities (e.g., processor core, cache, TLB's, etc.) are either spatially isolated or temporally isolated to enforce confidentiality.

Memory A TEE requires a secure storage area to store critical data, and this capability is implemented using a dedicated or shared memory component. The dedicated memory is used for critical storage. In the case of shared memory, HRT encrypts critical assets using an encryption engine to avoid information leakage from shared memory.

Encryption Engine A TEE includes a dedicated hardware based encryption engine (i.e., HRT) that converts exposed critical assets into ciphered critical assets and vice-versa.

An encryption engine requires secret keys to perform operations.

Secret Keys There are two types of secret keys (private keys and derived keys) within HRTs. Private keys are permanently fused cryptographic keys within the chip (e.g., secure storage, co-processor, or encryption engine). These keys are used within the encryption engine to encrypt critical assets or used to generate derived keys. Derived keys are unique cryptographic keys that are generated at runtime using permanently fused private keys. These derived keys have various purposes, including encryption for secure applications.

Secure Storage Secure storage is a multi-purpose dedicated memory region/module used to store derived keys and critical information (e.g., execution status of critical assets). Another important usage of secure storage is to store hashes for on-chip attestation of software packages, including firmware, OS, and hypervisor. Secure storage is only accessible to hardware-specific elements, e.g., memory encryption engine or a secure co-processor.

Secure Boot Secure boot is a hardware/software capability that allows on-chip attestation of firmware, operating systems, and applications using hashes stored in the secure storage.

3.2 Framework Application

The classification of threats and capabilities allows to compare distinctive HRTs with each other.

3.2.1 Security Analysis Comparison

HRTs consist of a set of capabilities that protect against threats and thereby assure certain security properties. A comparative security analysis of two different HRTs allows a user to identify: 1) where a particular threat is addressed by a HRT, 2) what capability defends against specific threats, and 3) what security property is thus guaranteed. For example, an HRT 'A' can protect against software threats using spatial isolation of hardware resources, while HRT 'B' protects software threats using temporal hardware resource isolation; both implementations will satisfy the security property of confidentiality but incur different performance and implementation costs.

4 Analysis of HRTs

The purpose of the framework is to allow users to differentiate between different HRT offerings. We aim to illustrate its use with a case study of ten state-of-the-art HRTs. We begin by breaking down these HRTs in terms of threats, security properties, and capabilities. Our results are summarized in Table 1.

4.1 Intel SGX

Intel SGX is an HRT with a trusted execution environment (TEE) [15] to protect critical assets against systen software level threats in cloud computing. Intel assumes that its processor chips are protected against sophisticated physical attacks due to its sensitive fabrication technology, the processor chip will malfunction or crash if any such attacks are conducted [11]. Therefore, Intel considers physical attacks on processor chips beyond the scope of SGX's threat model as shown in Table 1.

To protect against software attacks, Intel SGX executes critical assets (e.g., software libraries) in isolated containers called *enclaves*. Enclaves consist of *isolated hardware* using temporal isolation of execution resources and spatial isolation of memory regions using *enclave page cache (EPC)*. This memory area is not accessible to any user or privileged software (e.g., OS and firmware). Enclaves effectively enforce confidentiality to protect against software attacks. Intel SGX also features a memory encryption engine (MEE),

Table 1: Analysis & comparisons of HRTs in secure computing in terms of threats, security properties, and capabilities. "X" represents that specific threat(s) are not considered in the threat model; whereas the "gray box" depicts that some other capability is dealing with that specific threat within an HRT. An empty box represents no identifiable or documented description regarding that specific attack. C is confidentiality, and I is integrity.

Threats / Properties & Capabilities			Physical Threats			Software Threats			
			Probe Attacks			User Software		System Software	
			Bus	DRAM	Core	RAM	Core	Firmware	OS
Intel SGX	C	Hardware Isolation			X	Enclave	Enclave	Enclave	Enclave
		EPC							
	I / C	Primary Keys	MEE	MEE		MEE			
		Derived Keys							
		Secure Storage							
ARM TrustZone	C	Hardware Isolation	X	X	X	Secure World	Secure World		
	I	Secure Storage						Secure Boot	Secure Boot
		Chain of Trust							
NXP TrustLayer	C	Hardware Isolation				Secure World	Secure World		
	I / C	Fused Keys	SEC	SEC		SEC		Secure Boot	Secure Boot
		Derived Keys							
		Secure Storage							
	I	Temper detector			Temper Monitor				
AMD SEV	C	Keys							Secure Co-Processor
		Secure Storage							
		Crypto Engine	AES	AES					
Microsoft Pluton	C	Hardware Isolation			Side Channel Monitor	XVD	XVD		
	I / C	Private Keys	HW Crypto	HW Crypto		HW Crypto		Secure Boot	Secure Boot
		Derived Keys							
		Secure Storage							
		Crypto Engine							
Apple T2	C	Hardware Isolation				Secure Enclave	Secure Enclave		
	I / C	Private Keys	AES	AES		AES		Secure Boot	Secure Boot
		Secure Storage							
		Crypto Engine							

which is used to enforce integrity for critical assets. The MEE is made up of a *cipher engine*, permanently fused *primary keys*, dynamically generated *derived keys* and *secure storage* for derived keys. It is used to encrypt/decrypt critical assets and protect against attacks on interconnects and DRAM.

4.2 ARM TrustZone

ARM TrustZone is an HRT with a TEE [12] with applications in mobile and embedded security. It offers a system-wide approach to protect critical assets against software threats. ARM TrustZone provides *hardware isolation* built into the CPU by running two domains side-by-side, and sharing resources per set configuration to guarantee confidentiality. The two execution environments are: a "secure world" hosting TEE for running trusted code and a "normal world" for running general code. The partition of the two worlds is achieved by hardware logic in the AMBA bus fabric, peripherals, and processors. TrustZone can secure a software library or an entire OS to run in a secure area. It is based on the principle of least privilege, so the system modules (like drivers and applications) do not have access to a resource unless necessary. TrustZone includes a *secure boot* sequence that verifies secure boot images (i.e., firmware, OS, and secure applications), authenticated using private keys, stored in *secure storage*. Once booted, the communication between secure and normal world takes place via software called "core logic" in Cortex-M processors

or "secure monitor" in Cortex-A processors. It uses specific areas of memory that cannot be accessed by non-secure OS regardless of privilege level, and the operations executed in the secure world are not accessible by users with kernel access and rights. It considers physical threats as out-of-scope of their threat model and is silent about software side-channel attacks.

4.3 NXP TrustLayer Architecture

NXP Trust Architecture [4] is a system-on-chip (SoC) based HRT built using an ARM core with TrustZone technology and have applications in embedded systems. NXP's TrustLayer Architecture extends the threat vector of ARM TrustZone to include protection against both software and physical threats. It is silent about software side-channel threats to the best of our knowledge. NXP TrustLayer guarantees confidentiality using *hardware-enforced isolated* environment at the processor core, interconnects, and memory level critical assets using TrustZone Secure World (including *secure boot* to create chain-of-trust) and protects against software attacks. NXP TrustLayer enforces integrity property and protects against physical attacks on interconnects and DRAM with its security engine called SEC. SEC consists of a cipher engine, one-time programmable *primary keys* stored in *secure storage* called security fuses, and dynamically generated *derived keys*. NXP TrustLayer also includes a temper monitor detector that protects against physical threats on the SoC chip.

4.4 AMD SEV

AMD Secure Encrypted Virtualization (SEV) [1] is a security extension to AMD EPYC processors, with the primary purpose of protecting virtual machines (VM) against malicious system-level software (i.e., hypervisors). As AMD SEV has a specific threat model, it does not discuss other threats, including user-level attacks within VMs, firmware level attacks, and physical attacks. It consists of two modules, a secure co-processor, and a cipher module. The secure co-processor has an ARM core, fused *private keys*, and *secure storage*. The co-processor is used to generate and store derived keys (public/private), and configurations for VMs. Each VM's contents are encrypted with the private key using the cipher engine to isolate the VMs and the hypervisor from each other. This guarantees confidentiality for critical assets (i.e., VMs). SEV requires configurations for VM and hypervisor. The configuration allows the VM to indicate which memory pages should be encrypted. The hypervisor uses hardware virtualization instructions and communication with the AMD Secure processor to manage the memory controller's appropriate keys.

4.5 Microsoft Pluton

Microsoft Pluton [5] is an HRT introduced in Xbox products to prevent piracy of games and cheating on the Xbox platform. Pluton threat model protects against physical threats on exposed pins and wires and software threats, including bugs in the application and system-level software that leads to leakage of critical assets (i.e., primary keys). Pluton secure processor is a dedicated chip that is considered trustworthy. The secure pluton processor consists of a crypto engine, secure storage (for registers and derived keys), secure memory (RAM and ROM), primary fused keys, and a side-channel monitor. Xbox splits memory read-only OS memory and read-write register memory. OS is verified using a secure boot with fused keys. Xbox secure OS splits games into integrity-protected and encrypted Xbox Virtual Disks (XVD) that are verified using a hardware streaming crypto engine.

4.6 Apple T2

Apple T2 [6] is a multipurpose dedicated secure enclave co-processor that ensures security in Apple products. Apple threat model includes physical threats on DRAM modules and software threats involving application-level and system-level attacks. T2 includes secure enclaves that include encrypted memory storage to store private keys for secure boot, FileVault, and fingerprint data for Touch ID. Apple T2 includes a dedicated AES crypto engine that connects non-volatile storage with Apple general purpose processor. Secure boot capability verifies the integrity of firmware, boot loaders, and kernels to initiate a chain of trust for user-level applications.

5 Case Study Comparisons of HRTs

The framework is used to compare security analysis, performance implications, and hardware/software implementation costs for analyzed HRTs.

5.1 Security Analysis Comparison

Table 1 includes the details for the security analysis comparison of the HRTs in terms of their threats. HRTs are designed to protect against a wide range of threats. This section compares HRTs based on physical, software, and side-channel threats.

5.1.1 Physical Threats

involve probing exposed wires, pins and using sophisticated hardware (i.e., lasers) to leak secrets. All HRTs except ARM TrustZone include a crypto engine that encrypts all data leaving the HRT chip. These encrypted secrets travel on wires and are stored in secondary storage. An attacker can probe the wires and exposed pins but requires keys to decrypt secrets. All HRTs assume that physical chip that consists of an HRT are immutable to probing except NXP TrustLayer and Microsoft Pluton. NXP TrustLayer and Microsoft Pluton includes a dedicated temper monitor / side channel monitor capability to detect physical attacks on the chip.

5.1.2 Software Threats

are categorized into application-level attacks (i.e., bugs in software), and system-level attacks (i.e., malicious OS or firmware). All HRTs except AMD SEV offer a spatial or temporal isolated environment to protect against application-level attacks. AMD SEV threat model doesn't consider software level attacks, instead protects virtual machines against system-level attacks (i.e., malicious hypervisor). Intel SGX, ARM TrustZone, and NXP TrustLayer provide temporal isolation to critical assets for processing. Whereas Microsoft Pluton, and Apple T2 offer dedicated chip (spatial isolation) for critical assets processing. All HRTs except Intel SGX offer secure boot capability to protect against system-level attacks, and trust either firmware or OS. Intel SGX enclaves guarantee protection against system-level attacks.

5.1.3 Side Channel Threats

NXP TrustLayer and Microsoft Pluton include dedicated capabilities to protect against physical side-channel attacks (i.e., power or heat analysis). However, none of the HRTs guarantee protection against software side-channel attacks (i.e., timing attacks).

6 Conclusion

This paper developed an analytical framework for analyzing implementations of root-of-Trust in hardware. It allows a system developer or user to reason about distinct HRT implementations in terms of security, performance, and implementation cost. We use framework to evaluate ten HRTs, including Intel SGX, ARM TrustZone, NXP Trust Architecture, AMD SEV, Microsoft Pluton, and Apple T2 These HRTs have significantly different threat challenges. Even when systems consider the same threat, they often implement different capabilities to defend against it. This results in differences across performance and implementation cost. The appropriateness of an HRT depends on the operational exposure to threats, as well as practical constraints around overhead and resources. No single implementation will be a silver bullet. This research provides a way for both researchers and practitioners to examine and reach a common understanding of the relative merits of different HRTs.

References

[1] AMD, Secure Encrypted Virtualization (SEV) [Online]. Available: https://developer.amd.com/sev/ (URL)

[2] Dell, What Is Hardware Root of Trust? [Online]. Available: https://www.delltechnologies.com/en-us/blog/hardware-root-trust/ (URL)

[3] Intel, Intel Security Essential [Online]. Available: https://www.intel.com/content/dam/www/public/us/en/documents/solution-briefs/intel-security-essentials-solution-brief.pdf (URL)

[4] NXP, QorIQ Layerscape Processing Platforms Trust Architecture 3.0 [Online]. Available: https://www.nxp.com/pages/trusted-systems-technology:NETWORK_SECURITY_INT_SEC (URL)

[5] Guarding Against Physical Attacks: The Xbox One Story [Online]. Available: https://www.platformsecuritysummit.com/2019/speaker/chen/ (URL)

[6] Apple, T2 Security Chip [Online]. Available: https://www.apple.com/mideast/mac/docs/Apple_T2_Security_Chip_Overview.pdf (URL)

[7] Intel CSME [Online]. Available: https://www.intel.com/content/dam/www/public/us/en/security-advisory/documents/intel-csme-security-white-paper.pdf (URL)

[8] OpenTitan Security Model [Online]. Available: https://docs.opentitan.org/doc/security/specs/ (URL)

[9] NIST, Special Publication 800-164 [Online]. Available: https://csrc.nist.gov/CSRC/media/Publications/sp/800-164/draft/documents/sp800_164_draft.pdf (URL)

[10] TCG, Trusted Platform Module (TPM) [Online]. Available: https://trustedcomputinggroup.org/wp-content/uploads/Trusted-Platform-Module-Summary_04292008.pdf (URL)

[11] Intel Corporation, Intel Software Guard Extensions Presentation, *ISCA'15*, 2015.

[12] Julien Amacher and Valerio Schiavoni, 'On the Performance of ARM TrustZone', *Lecture Notes in Computer Science*, pp. 133–151, 2019.

[13] Saeid Mofrad, Fengwei Zhang, Shiyong Lu, and Weidong Shi, 'A Comparison Study of Intel SGX and AMD Memory Encryption Technology', *HASP'18*, article 9, 2018.

[14] Ofir Weisse, Valeria Bertacco, and Todd Austin, 'Regaining Lost Cycles with HotCalls: A Fast Interface for SGX Secure Enclaves', *ISCA'17*, 2017.

[15] Victor Costan and Srinivas Devadas, 'Intel SGX Explained', *Cryptology ePrint Archive*, report 2016/086, 2016.