Secure Processor Architecture: An Overview and Exploration of ARM TrustZone

Haotian Yang

May 1, 2024

1 Introduction

Security — As computational technologies evolve, individuals increasingly benefit from the conveniences of information exchange, rapid online transactions, and simplified electronic payments. However, these processes involve the handling of critical data, such as bank accounts, passwords, private documents, and personal information, underscoring the importance of cybersecurity. Protection of this key information can be achieved through various methods, including human intervention and technological implementation. In addressing the enhancement of security, both software and hardware components are pivotal. This paper will introduce aspects of security starting from the processor, or central processing unit (CPU), which is the core computational unit in the server, responsible for executing instructions and processing data. Secure processor architectures are primarily designed as extensions of commodity processors based on architectures like x86 or RISC[1]. Governments and companies have an increased need to explore effective security processor architectures due to two factors.

1. **Increased Software Complexity:** Modern operating systems and softwares, comprising millions of lines of code, present substantial security challenges. Their complexity not only increases the risk of vulnerabilities but also undermines the reliability of systems in safeguarding applications. In closed-source software, numerous zero-day vulnerabilities remain unknown until exploited. Conversely, in open-source frameworks, vulnerabilities might be intentionally introduced by anonymous entities, emphasizing the necessity of hardware-level security.

2. **Growth of Embedded and IoT Devices:** The proliferation of embedded and IoT devices, ranging from autonomous vehicles to smart wearable technology, marks a boosting trend. These devices, store personal information and connect to the Internet, raising concerns about data breaches and device vulnerabilities. Hacked devices can lead to consequences ranging from fraud to physical harm, necessitating the need of secure processor architectures to mitigate these risks[2].

2 Threats of Attacks

Attacks — The processor, in conjunction with the hardware and software in a computer system, is exposed to various security vulnerabilities that can be exploited by attackers. Attack methods are classified according to the attack pattern, which includes the location of the attack (local or remote) and the medium used (hardware or software)[1]. Several prevalent attack methodologies will be introduced:

1. **Physical Attacks**
    - **Shack Hacks:** Attackers utilize rudimentary tools to gain physical access to the processor through interfaces such as JTAG debug, boundary scan I/O, or built-in self-test facilities. They monitor and actively intervene with the processor by manipulating pins, altering bus lines, and employing other methods[3].
    - **Lab Attacks:** Attackers use high-precision laboratory equipment, such as electron microscopes, for detailed reverse engineering, such as monitoring analog signals to analyze cryptographic keys. These attacks are more costly than shack attacks.
2. **Hack Attacks**
    - **General:** Attackers employ software tools, such as viruses and malware, to infiltrate devices through physical or wireless connections. These attacks include side-channel attacks that exploit vulnerabilities by monitoring and probing cache memory to detect and extract data.
    - **Spectre and Meltdown Exploits:** These vulnerabilities illustrate the implementation of side-channel attacks. Spectre and Meltdown fundamentally manipulate design flaws and prediction features in modern processors to bypass system protections. Attackers exploit these vulnerabilities to conduct side-channel attacks, aiming to extract sensitive cache data, thereby compromising the confidentiality and integrity of computer systems[4].

3 Root of Trust

3.1 Foundational Concepts: TCB and TEE

The Trusted Computing Base (TCB) is composed of both hardware and software components that collaborate to uphold security guarantees as outlined by the system architecture. The TCB encompasses all components that are essential to maintaining security and supports the containment of security breaches.

The Trusted Execution Environment (TEE) includes all components within the TCB, designed to provide a secure execution environment for sensitive tasks.

The TEE ensures the protection and processing of sensitive data within a secure area of a processor, isolated from the regular operating system.

3.2 Introduction

In computer systems, security is implemented through multiple layers, each dependent on the reliability of the preceding layers[5]. The Root of Trust (RoT) forms the foundational core of this hierarchy, establishing the initial security anchor from which all subsequent security measures are derived.

The RoT comprises cryptographic keys crucial for securing the initial boot and system verification processes. This is one of two pillars of trust in secure processor architecture; the other being the reliability of the hardware's manufacturer and components, which is beyond the scope of this discussion[1]. Adhering to the principle that systems must boot securely or not at all, the implementation of RoT is vital, particularly in Systems on a Chip (SoC) and embedded devices[6]. By initiating a secure boot and anchoring the security strategy of the entire system, the RoT ensures that all operations within the TCB and TEE are performed in a secure environment, making it an indispensable element in the architecture of secure processors.

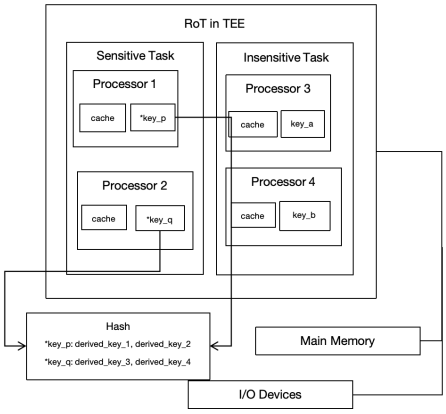

Figure 1: A diagram of the Root of Trust implementation of a simplified Trusted Execution Environment with four processors. Secure processors for the sensitive task obtain secret keys, 'ket_p', and 'key_q', which are stored within the processor boundary. The secret keys are not allowed to leave the boundary and can only be accessible to Trusted Computing Base components. Other keys are derived from the RoT secret keys.

3.3 Principles

The cryptographic keys representing the RoT must follow three fundamental principles: authentication, confidentiality, and integrity[1]. Authentication ne-

cessitates that each processor has a unique secret key, which pairs with a public key to facilitate authentication. Confidentiality is achieved through specialized encryption keys that safeguard data as it exits the processor. Integrity is maintained by ensuring that the processor's root key cannot be deduced from any derived keys, due to the one-way nature of the encryption employed. As Figure 1 illustrates, keys going out of the processor boundary are derived from the root keys and encrypted.

4 Implementation of TEE

TEE — A Trusted Execution Environment (TEE) is a tamper-resistant processing environment operating on a separation kernel that protects its runtime states and stored assets[7]. The TEE prioritizes various security goals, protecting Trusted Software Modules (TSMs), virtual machines (VMs), and containers. It ensures the authenticity of the execution code and the confidentiality and integrity of runtime states stored in persistent memory. Several exemplary TEE architectures will be introduced:

1. **eXecute-Only Memory:** eXecute-Only Memory (XOM) aims to segregate program execution into mutually exclusive memory 'compartments'. As Figure 2 illustrates, the functional API of each compartment is helpful for components segregation. The system prohibits programs from accessing data in compartments other than their own. Each compartment is associated with a unique session key for the encryption and decryption of instruction and data streams. Only one compartment can be active at any time and its key will be loaded into the hardware and utilized to provide cryptographic protection[1].

2. **Intel's Secure Guard Extensions:** Intel's Secure Guard Extensions (SGX) aims to protect trusted software modules (enclaves). It creates a clear division between trusted and untrusted code segments, ensuring sensitive operations are securely executed within the enclaves. SGX can filter unauthorized access by creating encrypted containers, even from privileged software, to shield enclaves from untrusted operating systems and external entities[8].

3. **ARM TrustZone:** ARM TrustZone aims to create two separate 'worlds' for executing a trusted and secure operating system to run in parallel with an untrusted and normal operating system[1]. It ensures that the secure world is safe even if the normal operating system or its applications are attacked. Trust Zone incorporates tag memory and system buses with identifiers to show the current execution world and allow the secure world to obtain the resource exclusively.

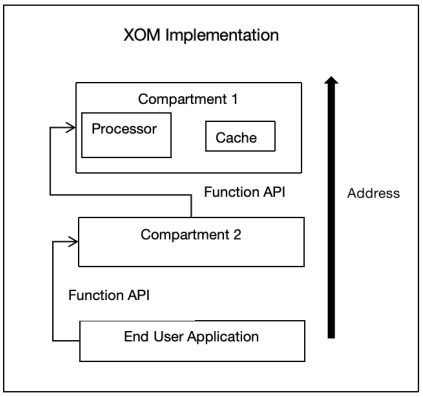

Figure 2: A diagram of the eXecute-Only Memory implementation workflow. The end-user application needs to utilize the function API to operate each compartment.

5 Evolution of Arm TrustZone

TrustZone — In the development path of ARM TrustZone, optimization of this security technology has been enhanced in the ARM architecture series from ARMv6 to ARMv8. Initially introduced in 2007 with ARMv6, TrustZone created a dual-world security structure—consisting of a secure and a non-secure world. When the processor operates in the secure world, it accesses both secure and non-secure world resources. However, in non-secure mode, it is restricted from accessing secure resources.

In ARMv7-A, TrustZone was updated to include a secure monitor mode, which manages the transition between the two worlds via a Secure Monitor Call (SMC) and preserves the processor state during switches[9]. As Figure 3 illustrates, the secure monitor is the base layer of the architecture that is shared with both worlds. It also introduced the 'Hypervisor' as a part of ARM's virtualization extensions to operate in the non-secure world at a higher privilege level and activate Stage-2 page tables to manage and translate the memory—first from Virtual Address (VA) to Intermediate Physical Address (IPA), and then from IPA to Physical Address (PA)[9]. This setup allowed the hypervisor to monitor and control OS memory access attributes finely.

In ARMv8-M, TrustZone was optimized to reach faster context switching and low-power applications[9]. Unlike ARMv7-A, It eliminates the need for a secure monitor mode to reduce world-switching latency by memory mapping and implementing transitions automatically by programs. As Figure 4 illustrates, the transition between both worlds can be initiated directly instead of asking for a secure monitor.

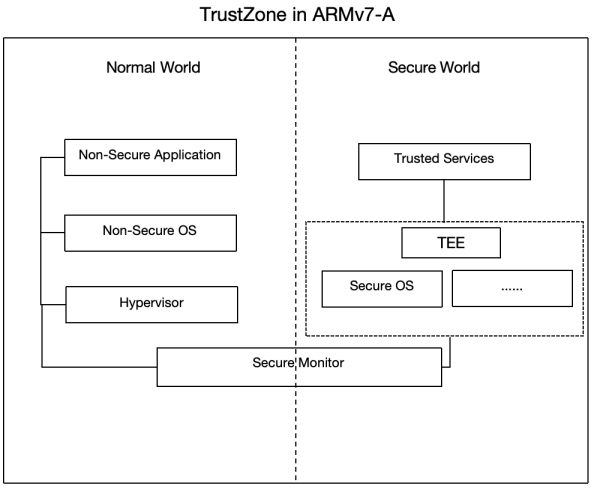

Figure 3: A diagram of Arm TrustZone in ARMv7-A

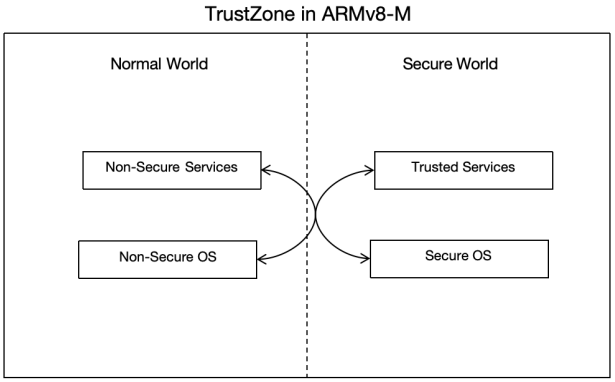

Figure 4: A diagram of Arm TrustZone in ARMv8-M

6 Applications of TrustZone

Applications — ARM TrustZone has become a popular security processor architecture in recent years, with applications spanning various fields to enhance security. Two exemplary applications will be introduced:

1. **Online Payment:** E-commerce, happening everywhere in the world, is inherently more vulnerable to security threats such as scams and transaction failures than traditional trades. Particularly for online payments, where security and stability are greatly needed. A research by Prof. Kamble, P.A., and Miss. 'Neha Patil has shown a model to secure online payments baked on ARM TrustZone[10]. They utilize TrustZone to provide a secure execution environment for processing transactions.

   As Figure 5 illustrates, a designed online payment stream system, 'Darkroom,' initiates from the customer application, which sends encrypted image data to a cloud server. The server protects this data throughout its lifecycle within the secure confines of a TrustZone-enabled processor. The data undergoes secure modifications requested by the user and is then transferred back to the normal operating environment after processing. This implementation of TrustZone shows potential for mitigating common e-commerce security risks and providing a secure payment environment.

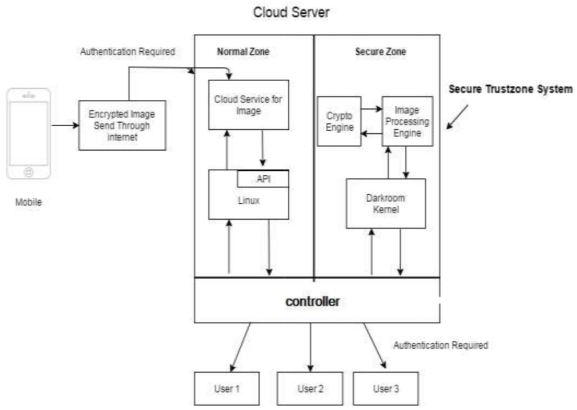

Figure 5: A diagram of execution stream for the 'Darkroom' system[10]

2. **Vehicle Engine Control Units:** Embedded systems, particularly in the automotive aspect, with a strong need for security due to the nature of slow iteration and isolation of the platform, are dependent on secure processor architecture such as Arm Trustzone. In a survey published by ProvenRun, Vehicle Engine Control Units (ECUs), which are crucial for managing various vehicle functions, the implementation of TrustZone to secure ECUs by isolating critical computing tasks in a secure environment is highlighted[11]. This isolation helps prevent unauthorized access

to vehicle operations and secure sensitive tasks such as signal processing. Categorized by the tasks performed by vehicle ECUs, including the Electric Vehicle Charger Controller (EVCC), central gateways, and others, these units are segregated within a TEE and communicate with each other following the TrustZone model. This structure ensures secure interactions across different ECUs and the overall security framework of the vehicle.

7 Conclusion

In this survey, a comprehensive overview of Secure Processor Architecture with an exploration of ARM TrustZone and its implementation was discussed. Processor security is fundamentally important and originates from the most basic layer, such as the Root of Trust (RoT) previously discussed. The sources of threats and attack methods are constantly evolving; hence, a security architecture cannot be safe at all times. Therefore, cooperation among all components is crucial. There is no absolutely safe architecture; as long as the probability of being hacked is sufficiently low or the attacker decides to give up due to the high cost of the attack, the architecture is considered successful. TrustZone is a highly useful tool, and various institutions and people continuously update and work on it. However, vulnerabilities in this architecture have been discovered and exploited. Thus, there is a strong need to continue implementing and researching secure processor architecture with stricter requirements.

8 Bibliography

References

[1] Jakub Szefer. *Principles of Secure Processor Architecture Design*. Springer Nature Switzerland AG, 2022.

[2] Sara Sun Beale and Peter Berris. *Hacking the Internet of Things: Vulnerabilities, Dangers, and Legal Responses*. In Eric Hilgendorf & Jochen Feldle (Eds.), *Digitalization and the Law* (forthcoming). Presented at the conference on Technology and the Law, University of Würzburg, 2018. Charles L.B. Lowndes Professor, Duke Law School; J.D., Duke Law School, 2017.

[3] ARM Limited. *ARM Security Technology: Building a Secure System using TrustZone Technology*. Copyright 2005-2009 ARM Limited. Revisions:

- Revision A - December 2008: First release.
- Revision B - January 2009: Minor language clarifications, fixed monitor latency calculation on page 5-12.
- Revision C - April 2009: Added information related to multiprocessor systems (Accelerator Coherency Port on page 3-10; Multiprocessor

systems with the Security Extensions on page 3-13; Multiprocessor debug control on page 3-18; Secure software and multiprocessor systems on page 5-13; Hardware design checklist on page 7-3; Software design checklist on page 7-5).

[4] Hill, M. D., Masters, J., Ranganathan, P., Turner, P., & Hennessy, J. L. (2019). *On the Spectre and Meltdown Processor Security Vulnerabilities*. Institute of Electrical and Electronics Engineers (IEEE). doi:10.1109/mm.2019.2897677

[5] Andrew Regenscheid. *Roots of Trust in Mobile Devices*. Presented at the ISPAB, February 2012. Computer Security Division, Cryptographic Technology Group.

[6] Vincent Zimmer, Senior Principal Engineer, Intel, vincent.zimmer@intel.com; Michael Krau, Systems Engineer at Intel, michael.p.krau@intel.com. *Establishing the Root of Trust*, August 2016.

[7] Mohamed Sabt, Mohammed Achemlal, Abdelmadjid Bouabdallah. *Trusted Execution Environment: What It Is, and What It Is Not*. 14th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, August 2015, Helsinki, Finland. doi:10.1109/Trustcom.2015.357. HAL Id: hal-01246364.

[8] Victor Costan and Srinivas Devadas. *Intel SGX Explained*. Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology. Emails: victor@costan.us, devadas@mit.edu.

[9] Sandro Pinto, Centro Algoritmi, Universidade do Minho; Nuno Santos, INESC-ID, Instituto Superior Técnico, Universidade de Lisboa. *Demystifying Arm TrustZone: A Comprehensive Survey*. ACM Computing Surveys (CSUR), Vol. 51, No. 6, Article 130, January 2019, Pages: 1-36. DOI: https://doi.org/10.1145/3291047.

[10] Prof. Kamble, P.A. and Miss 'Neha Patil. *Secure Online Payment with ARM TrustZone*. International Journal of Development Research, Vol. 07, Issue 07, pp. 13872-13875, July 2017. ENTC Department, SIT Lonavala, Pune, Maharashtra, India.

[11] ProvenRun. *Trusted Execution Environment in Automotive Critical ECUs*.