electronics

MDPI

Article

Design Methodology and Metrics for Robust and Highly Qualified Security Modules in Trusted Environments

Luca Crocetti *, Pietro Nannipieri, Stefano Di Matteo and Sergio Saponara

Department of Information Engineering, University of Pisa, Via G. Caruso, 16, 56122 Pisa, Italy; pietro.nannipieri@unipi.it (P.N.); stefano.dimatteo@dii.unipi.it (S.D.M.); sergio.saponara@unipi.it (S.S.)
* Correspondence: luca.crocetti@unipi.it

Abstract: Cyberattacks and cybercriminal activities constitute one of the biggest threats in the modern digital era, and the frequency, efficiency, and severity of attacks have grown over the years. Designers and producers of digital systems try to counteract such issues by exploiting increasingly robust and advanced security mechanisms to provide secure execution environments aimed at preventing cyberattacks or, in the worst case, at containing intrusions by isolation. One of the most significative examples comes from General Purpose Processor (GPP) manufacturers such as Intel, AMD, and ARM, which in the last years adopted the integration of dedicated resources to provide Trusted Execution Environments (TEEs) or secure zones. TEEs are built layer by layer on top of an implicitly trusted component, the Root-of-Trust (RoT). Since each security chain is only as strong as its weakest link, each element involved in the construction of a TEE starting from the RoT must be bulletproof as much as possible. In this work, we revise and propose a design methodology to implement in both hardware (HW) and software (SW) highly featured and robust security blocks by highlighting the key points that designers should take care of, and the key metrics that should be used to evaluate the security level of the developed modules. We also include an analysis of the state of the art concerning RoT-based TEEs, and we illustrate a case study that documents the implementation of a cryptographic coprocessor for the secure subsystem of the Rhea GPP from the European Processor Initiative (EPI) project, according to the presented methodology. This work can be used by HW/SW security module designers as a cutting-edge guideline.

Keywords: computer security; Trusted Execution Environment; Root-of-Trust; secure boot; Side-Channel Attack; Crypto-Tile; European Processor Initiative; Rhea general-purpose processor

Citation: Crocetti, L.; Nannipieri, P.; Di Matteo, S.; Saponara, S. Design Methodology and Metrics for Robust and Highly Qualified Security Modules in Trusted Environments. *Electronics* **2023**, *12*, 4843. https://doi.org/10.3390/electronics12234843

Academic Editor: Andreas Mauthe

Received: 14 October 2023
Revised: 27 November 2023
Accepted: 28 November 2023
Published: 30 November 2023

1. Introduction

In the modern age, due to the pervasive adoption of technology in even more contexts and aspects of human life, digital systems are becoming more complex and sophisticated day by day, aiming at ensuring more advanced and efficient services. On the one side, this trend is bringing unprecedented benefits; on the other hand, the increasing amount of processed and exchanged data, interconnections, and infrastructures is extending the attack surfaces for cyberattacks. Indeed, hand in hand with the evolution of technology, in the last decades the attacks aimed at violating the security of data have become more elaborate, more frequent, and their consequences more severe [1–3]. The main approach to counteract this phenomenon relies on equipping digital systems with dedicated and adequate security mechanisms to prevent attacks by reducing the vulnerabilities and to contain them or limit their effects with tempest reactions. In this sense, the state of the art promoted by the main General Purpose Processor (GPP) manufacturers lies in the creation of Trusted Execution Environments (TEEs). A TEE is a secure and isolated area within a processor or microcontroller that provides a high level of security for running specific applications and processing sensitive data. TEEs are designed to protect against various forms of attacks, including software-based attacks, hardware attacks, and even attacks

from the operating system or other running applications. Indeed, they provide a safe and isolated environment where code and data can be executed and processed securely. TEEs are often built on top of a Root-of-Trust, i.e., a foundational component or process that is inherently trusted to perform critical security functions [4–6]. This can include tasks like hardware-based secure boot, key management, and attestation. TEEs typically leverage the hardware-based security features provided by the RoT to establish and maintain their security. For example, some TEEs, like ARM TrustZone or Intel Software Guard Extensions (SGX), rely on a hardware-based RoT to establish a secure boot process and to ensure the integrity and confidentiality of the TEE's operation. The construction of a TEE starting from an RoT consists of several steps. It involves several elements, each of which must be carefully designed to not introduce security vulnerabilities that might compromise the overall security of the TEE. This is because TEEs built on top of RoT are essentially security chains that are only as strong as their weakest link. Indeed, several examples of security flaws in TEEs of GPP producers can be found in the literature. For instance, in the case of the ARM TrustZone, cache SCAs have been exploited in [7–10], EMA attacks in [11,12], and fault attacks in [13].

In this work, we present a comprehensive design methodology and the security design metrics for the elements (both hardware and software) that are involved in RoT-based TEEs. The application focus is on embedded security: we aim to provide the hardware–software infrastructure that can be used by system developers to enforce the security of their embedded systems. We aim to exploit the lesson learned from the EPI project (see [14–23] as references to the implementations of the specific HW accelerators) to make available to the security community the methodology and strategy to design such an advanced system. The presented methodology builds on the experience acquired during the design of a cryptographic coprocessor to assist the secure boot process in the Rhea GPP, which represents the first family of processors born from the European Processor Initiative (EPI). Our goal is to extract the lessons learned from the design activity and develop a generic methodology that can be proposed to the security community. The EPI project is a collaborative effort between various stakeholders, including research institutions, universities, and industry partners, to drive innovation in processor technologies, and it is aimed at developing and promoting European-designed and European-manufactured High-Performance Computing (HPC) processors and accelerators. It represents a strategic effort that seeks to reduce Europe's reliance on non-European technologies in the field of supercomputing and data processing by ensuring that Europe has access to cutting-edge processor technologies including state-of-the-art security mechanisms.

The remainder of this work is organized as follows: Section 2 provides the main definitions, principles, and outline of Roots-of-Trust and Trusted Environments, by highlighting the mechanisms that involve each other. Section 3 gives a review of the state of the art concerning RoT-based TEEs from the main GPP manufacturers such as Intel, AMD, and ARM. Section 4 presents the proposed methodology and the most qualifying security metrics that should be carefully addressed when designing any of the elements involved in the construction of a Trusted Environment on top a Root-of-Trust. In Section 5, we present a case study concerning the development of a cryptographic coprocessor that has been included in the Security Subsystem of Rhea GPP, according to the proposed methodology and metrics. Finally, Section 6 summarizes the conclusions of this work.

2. Definitions, Principles, and Outline of Roots-of-Trust and Trusted Environments

A Root-of-Trust (RoT), sometimes referred to as Trust Anchor, is a fundamental component in computer and information security. It is an element, typically implemented in hardware or firmware, that is considered implicitly trusted and forms the basis for establishing and verifying the authenticity and integrity of various system components, software, and data. The primary purpose of an RoT is to provide the secure foundations for building and maintaining trust within a computing environment. For instance, it is used for the following:

- Secure Boot and Initialization. The RoT ensures that a computer system starts up securely. It verifies the integrity of the bootloader and firmware during the boot process, ensuring no malicious code has been injected into these critical components. This helps prevent firmware-level attacks and rootkits from compromising the system.
- Code and Data Authentication. The RoT mechanisms can be used to authenticate and verify the integrity of software applications and data. This ensures that only trusted and authorized code and data are executed or accessed on a system.
- Cryptographic Operations. The RoT often includes cryptographic capabilities and can generate and manage cryptographic keys. This is crucial for secure communication, data encryption, and authentication between different system components and devices.
- Secure Storage. Some RoT implementations provide secure storage for sensitive data, such as cryptographic keys or biometric templates, ensuring that this data cannot be easily tampered with or stolen.
- Remote Attestation. The RoT can be used for remote attestation, allowing a system to prove its trustworthiness to remote entities. This is particularly important in scenarios like cloud computing, where a remote server needs assurance that a client device is secure before granting access.
- Hardware-Based Security. Many RoT implementations are designed to be tamper-resistant and are stored in secure hardware modules, making it difficult for attackers to compromise their integrity.
- Trust in Supply Chain. The RoT can establish trust in the supply chain by ensuring that hardware components and software are genuine and have not been tampered with during manufacturing, distribution, or deployment.
- Protecting Against Insider Threats. The RoT can also help protect against insider threats by limiting access to sensitive information and functions based on authentication and authorization policies.

The RoT is the key element for building Trusted Execution Environments, typically through secure boot routines. In this sense, a TEE can be seen as a layered structure in which each layer is enabled (and configured) upon verification of the previous layer. The resulting environment constitutes a secure zone where code and data can be executed and processed securely. Each layer provides specialized security mechanisms that rely on the ones of the lower layer and support the higher layer. In case of an attack on any of the elements of a layer, the corresponding layer can attempt to counter the attack with its security mechanisms, and, in case of failure, that layer can be isolated by leveraging the lower layer. The highest layer of the security chain is the one connected with the main operating system, other parts of the processor, and eventually external systems, which are insecure. In some cases, not only can the highest layer be interfaced with insecure elements or zones, but also lower layers can, for instance, to provide advanced specialized services. In the former case, the TEE can be modeled as a Chain-of-Trust (CoT), i.e., the organization of the layers is purely sequential; in the latter case, the TEE can be modeled as a Tree-of-Trust (ToT), i.e., some layers concur with each other. A representation of the two different topologies is given in Figure 1.

The separation between the secure zone and the non-secure zone can be physical or logical. In the first case, only the secure software can access the hardware security-critical components, and the insecure software can exchange data with the secure software to request security services. The most significant example relies on the usage of two distinct processing units. One of the processing units (e.g., a MicroController Unit, MCU) is dedicated to all and only the security-related functions and processes, and is eventually assisted by additional hardware accelerators or specialized circuits; for this reason, such a microcontroller is called a Secure MCU, and it is provided with its own RAM that is called Secure RAM (SRAM). The other processing unit (e.g., the main processor) is in charge of executing the operating system and general-purpose applications. The main processor can make a request for security services to the Secure MCU through dedicated interconnections

(e.g., a mailbox system). In the second case, both secure and non-secure software share the same hardware resources, but only the secure software has access to the resources dedicated to the security functions. An example can be a unique processor that executes both security-related software and non-security-related applications, hence exploiting the same RAM for both of them. Furthermore, some memory spaces are dedicated only to security-related data and instructions, and their access is forbidden to non-secure applications. Section 3 reports an overview of the state-of-the-art RoT-based Trusted Environments.

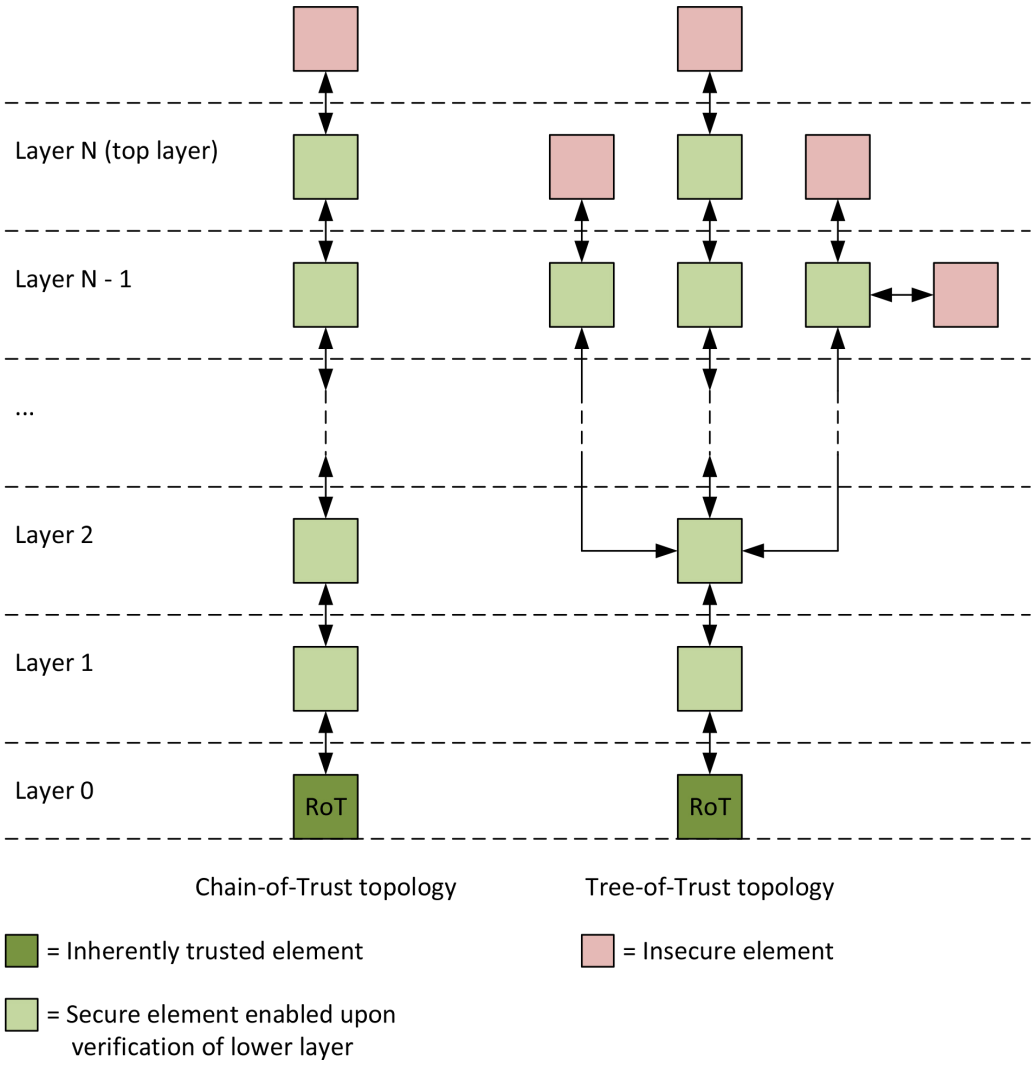

Figure 1. Layered organization of TEEs as Chain-of-Trust or Tree-of-Trust.

3. State of the Art of RoT-Based Trusted Environments

Several implementations of hardware security solutions are released by the main GPP vendors like ARM, Intel, and AMD targeting different uses such as isolated execution and TEE, acceleration of security functions, random number generation, and memory protection.

ARM proposes the so-called TrustZone [24,25], which is a system-wide technology integrating security features into the ARM processors, bus fabric, and peripherals. Instead of protecting assets in a dedicated hardware block, the TrustZone architecture adds security functionalities to any part of the System-on-Chip (SoC). The TrustZone strategy is based on the partitioning of all the SoC resources (both hardware and software) in two worlds: the Secure world for the Security Subsystem and the Normal world for everything else. At the hardware level, the division of the world is implemented through different hardware features: the bus fabric can ensure that no Secure world resources can be accessed by

the Normal world components. This is accomplished through the Advanced eXtensible Interface (AXI) bus, which contains a dedicated bit to determine secure and non-secure read/write operations; hardware extensions are included into the processor in order to enable it to execute code from both the Normal world and the Secure world in a time-sliced fashion. This feature removes the need for a dedicated processor for the security part, so the single processor runs the secure software when its state is secure, and the normal software when it is in the non-secure state; introduction of a hardware classification of the memory-mapped devices as secure or non-secure through the TrustZone Address Space Controller (TZASC) and memory division into secure and non-secure through the TrustZone Memory Adapter (TZMA); and distinction between secure and non-secure prioritized processes interrupts through the interrupt controller. Each world manages the resources for applications belonging to its world's space. The two different software stacks (i.e., secure software and non-secure software) are executed one at a time on the same processor. The context switching between Secure and Normal world is handled by monitor mode, which is the highest privilege level of the Secure world and can access both worlds' system's resources. ARM TrustZone is integrated inside Cortex-A [26], as well as in Cortex-M [27] with some differences related mainly to privilege levels of the processor.

The Intel Software Guard eXtensions (SGX) is an isolation technology included in almost all the 6th generation of Intel processors, which virtually creates a secure zone for secure code. This technology, unlike the ARM TrustZone, does not divide the execution process into two worlds (i.e., Secure and Normal world) but creates a secure computing space within the untrusted application: the Enclave. The Enclave is a protected container the applications can instantiate, which ensures confidentiality and integrity by using memory access checks and by encrypting data and code that goes outside the CPU. At the hardware level, the architectures supporting SGX are equipped with a set of security-related instructions and dedicated hardware components like the Enclave Page Cache (EPC) to support the multitasking of different Enclaves and the Enclave Page Cache Map (EPCM) to track page space of each Enclave. As discussed in [28], a user application can instantiate an Enclave using a trusted process. Such a process communicates with the EPC to assign an enclave page to the application, and the page information is checked by the EPCM. The target process is executed thanks to the Enclave and after its completion, the Enclave is destroyed and the page space released. Further information about the Intel SGX technology can be found in [29–33].

AMD proposes the Platform Security Processor (PSP) [34], which is a standalone coprocessor embedded inside the main AMD CPU. It is responsible for creating, monitoring, and maintaining the security environment, including managing the boot process, initializing security-related mechanisms, monitoring the system for any suspicious activity, and implementing an appropriate response. The PSP consists of an ARM microcontroller, cryptographic coprocessors, local memory, local registers, and interfaces to interact with the system memory, IOs, and configuration registers. The primary function of the PSP is to protect the main AMD core and to provide the hardware RoT; the PSP is in charge of the boot sequence using its own ROM and SRAM using the Unified Extensible Firmware Interface (UEFI) Secure Boot process. The communication mechanism between the PSP and the main CPU is implemented by interrupts: the PSP processor can generate interrupts to the CPU using PCI-compliant messages and the main CPU can raise interrupts to the PSP as well. This is accomplished via a PSP–CPU mailbox mechanism.

The CryptoManager Root-of-Trust (CMRT) [35,36] is a family of hardware-based RoT IPs developed by Rambus. It includes a 32-bit RISC-V processor, a ROM unit, and hardware resources dedicated to accelerating the security algorithms and managing the security assets. It also features private buses and interfaces for the integration of One-Time Programmable (OTP) memories and SRAMs within the Trusted Environment. It was developed to assist general-purpose processing units for Internet-of-Things (IoT), automotive, connectivity, and sensor applications.

Another commercial solution for hardware RoT is the Synopsys tRoot Hardware Security Module (HSM) [37]. The tRoot HSMs are designed to be integrated into SoCs, and the architecture can be customized depending on the application. The tRoot includes a CPU with secure instructions and data controllers for external storage, hardware key management and protection, hardware accelerators for cryptographic functions, and different interfaces (e.g., UART, GPIO, etc). The main goal of the tRoot is to provide a TEE to protect sensitive information and processing and implement security-critical functions such as secure boot, storage, debugging, anti-tampering, and key management.

For what concerns academic solutions, the work in [38], named Bastion architecture, is a hardware–software architecture for protecting security software modules in an untrusted software stack. At the hardware level, Bastion features dedicated instructions and registers inside the CPU, dedicated registers external to the CPU for storing hash values and keys, and dedicated engines for encryption, hash functions, and random number generation. The strategy in the Bastion architecture is to provide direct hardware protection of the hypervisor before employing it to protect the operating system and application modules. A similar solution has been proposed in [39], where the authors present an architecture named HyperWall to provide hypervisor-secure virtualization through hardware support. Also in this case, the solution of [39] proposes a hardware extension applied to a standard processor rather than dedicated and isolated hardware for security. The hardware extension includes new instructions, dedicated registers for the virtualization and key materials, and crypto-engines for random number generation, encryption, and hash functions. Other academic solutions can be found in [40–42].

The review of the state-of-the-art commercial solutions for TEEs is summarized in Table 1.

Table 1. Comparison of state-of-the-art TEEs.

	Rambus CMRT	ARM TrustZone	Intel SGX	AMD PSP	Synopsis tRooT
Implementation Strategy	Discrete	Integrated	Integrated	Integrated	Discrete
Isolation approach	Physical	Logical	Logical	Physical	Physical
Platform(s)	SoC IP	Cortex CPUs	from 6th generation of Intel CPUs	Ryzen CPUs	SoC IP

4. Design Methodologies and Key Aspects for Security Modules in Secure Zones

According to how much was highlighted by the review of the state of the art (Section 3), each of the proposed Trusted Environments relies on different and multiple components such as microcontrollers, OTPs, and coprocessors. As each of these elements is involved in the construction and maintenance of the TEE, each of them must offer an adequate level of security and adequate characteristics to carry out such purposes. Any flaw or vulnerability in any of the elements involved in these processes can severely compromise the security of the whole TEE. From a security point of view, any system integrating or adopting a Trusted Environment can be modeled as illustrated in Figure 2, by separating the security-related functions resources (Secure zone, green box) and non-secure-related resources (Non-secure zone, red box). The block that connects the zones is the representation of all the mechanisms through which the two zones can interact. It must integrate robust security policies that define how the assets of the secure zone may be accessed. For instance, it must regulate the content and the transfer methods of data from one zone to the other one and vice versa.

According to the model in Figure 2, three key aspects must be addressed in the implementation of the secure zone to guarantee such an environment is trusted. The three aspects are:

1. the Security Services and Functions;
2. the Interface Security Policy;
3. the Physical Implementation.

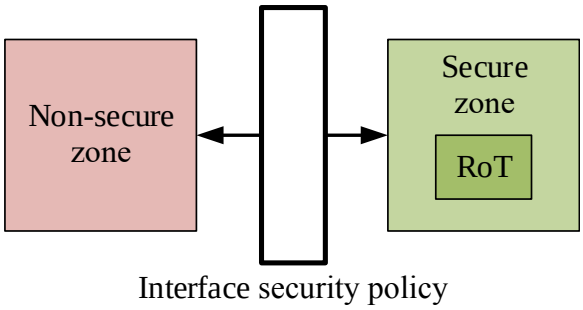

Figure 2. Security partitioning model of a generic system integrating a TEE.

The first aspect concerns the definition of which security function, service, or mechanism is included (e.g., confidentiality, integrity, authentication), how it is implemented (e.g., symmetric-key cryptography, hash functions, public-key cryptography, or other solutions), and the level of security required. In particular, this last point regards the usage of cryptographic algorithms. Indeed, the National Institute of Standards and Technology (NIST), which is one of the most important organizations in the matter of cybersecurity and standardized most of the cryptographic algorithms used, defined different levels of security. The security level (or strength) is expressed in bits, and the minimum accepted level for long-term protection is 128 bits. The quantification of the security strengths offered by the most diffused cryptographic algorithm is reported by the NIST in [43]. In addition, the used cryptographic algorithm(s) must also be verified at the functional level, ensuring its compliance with the corresponding standard. For this purpose, the NIST releases test vectors for each of its standardized algorithms through the Cryptographic Algorithm Validation Program (CAVP).

The second aspect focuses on the mechanisms aimed at managing and regulating the access and the life cycle of the security-critical assets. Once the security service and the cryptographic functions are defined at the first point, it is possible to individuate the security-related material and sensitive data that will be processed such as the cryptographic keys. In this case, the interface security policy may integrate dedicated mechanisms for the establishment or installation, the secure storage, and the maintenance of keys. Seal/unseal mechanisms can be used to protect keys from unauthorized access (by supporting different privilege levels), to bind the key for a specific usage (e.g., a specific cryptographic operation), and/or to limit its usage (e.g., for one, tens, or hundreds of times). Also, data access must be regulated. In this sense, an interface security policy mechanism may rely on a dedicated Finite State Machine (FSM) that strictly handles the data processing phases and allows access only once the cryptographic operation is completed. Any access attempt before the conclusion of the cryptographic operation may be logged and reported as a warning or as an alert.

The third and last aspect, which is equally important, concerns the physical implementation. Indeed, also the physical implementation of a secure system or module can introduce vulnerabilities, even if the chosen security algorithm is robust and cannot be violated from a numerical or mathematical point of view. When the module executes the chosen algorithm, the physical level may emit sensitive information through physical quantities, such as time, power consumption, electromagnetic radiation, or sound. This constitutes an unintentional leakage of information through a secondary channel (or side channel), i.e., the physical one, and it can be exploited to perform an attack bypassing the interface security policy level. To give a more intuitive explanation, for example, it is enough to consider the physical processes involved in the usage of transistors implemented in CMOS technology. When these hardware elements treat digital information formed by 0s and 1s, they can be assimilated into a capacitor which is charged or discharged, according to the digit value. Therefore, by measuring the power consumption of transistors, it could be possible to recover the value of the processed data. Due to their nature, such attacks take the name Side-Channel Attacks (SCAs), and Figure 3 depicts their scheme of operation.

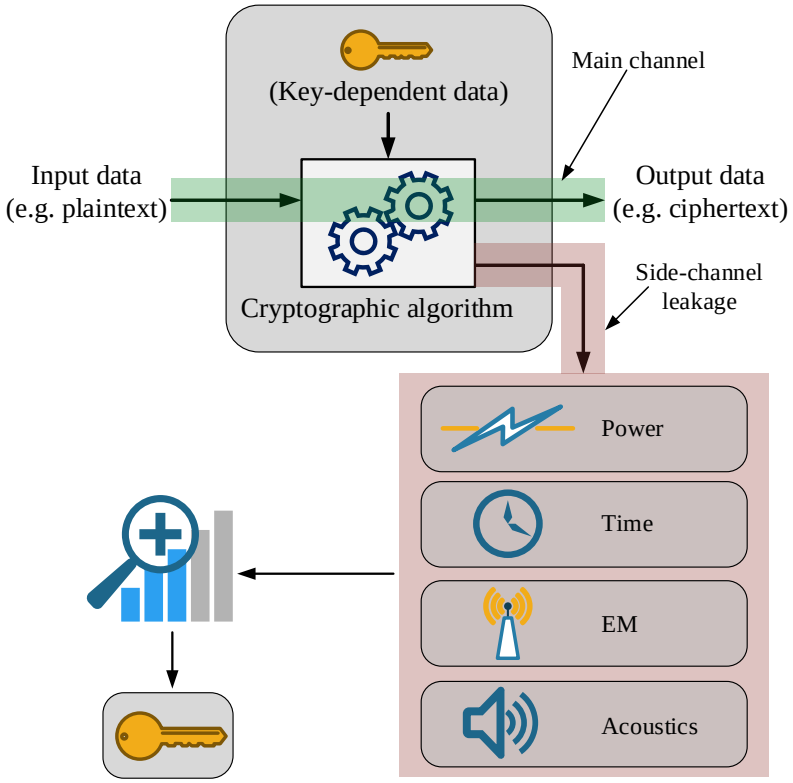

Figure 3. Scheme of operations of SCAs. By analyzing the leakage of information emitted through a side channel such as power, time, Electromagnetic Emissions (EM), or acoustics, an adversary is able to retrieve the secret key.

SCAs are widely documented in the literature and constitute one of the hottest topics in the cybersecurity field. An exhaustive and systematic review of SCAs can be found in [44–46], while [47–63] report several examples of the most diffused categories of attacks that exploit the physical implementation of a device, which are listed below:

- Cache attack [47–49]: This class of attacks is based on monitoring cache accesses in a shared physical system and, as the name suggests, it is mostly applied to software contexts. The protection strategies usually consist of replacing the LUTs with a series of equivalent logical operations, using alternative forms of LUTs, or creating an obvious memory access pattern (i.e., reading all LUT values in a fixed order and using only the one needed) [47,48].
- Timing attack [50–53]: This category of attacks concentrates on measuring the execution time of routines, targeting software implementations in which different instructions, branches, RAM cache hits, etc., cause time variation in the process. The protection mechanisms typically involve techniques for time equalization and reduction of unpredictable events (such as cache misses, etc.), or insertion of random delays [51–53].
- Power analysis [53–61,64,65]: This family of attacks is one of the most effective on both hardware and software implementations, and it exploits the power consumption of underlying physical circuits. Concerning the software case, if each opcode of the instruction set architecture can be associated with a different and specific power trace shape, then the series of operations performed by a routine is revealed; on the hardware side, data dependencies over different power traces can be analyzed to recover secret information. The most notorious attack classes of this group are Single Power Analysis (SPA), Differential Power Analysis (DPA), and Correlation Power Analysis (CPA). The first one consists of recovering data values by just observing the shape of a power trace; for instance, in a branched architecture where a certain value of processed information activates a certain branch rather than another one, isolating

the branch-specific power profile within a power trace makes it possible to recover the sequence of 0s and 1s. DPA and CPA attacks instead are more sophisticated, and they employ statistical methods to disclose secret information. As can be intuited from what has been expressed so far, each of these attacks requires a certain knowledge of architecture, especially in the case of DPA and CPA, which typically make use of a power consumption model to make a guess of the value of processed data: the further the model from the real architecture, the weaker the guess. For power analysis attacks, the countermeasures usually are implemented by adopting two techniques, masking or hiding, or a mix of both. The former is aimed to reduce or eliminate the dependencies between data and power consumption; the latter instead is aimed to jam the Side-Channel with noise in order to reduce the signal-to-noise ratio of leaked information. Masking typically introduces significant overhead in terms of area and critical path, by requiring up to 5 times the amount of logic resources employed by the unprotected module, and lowering the maximum frequency of the circuit up to 2/3 times [53,56,59,61,65]. On the other hand, hiding usually is realized by coupling the unprotected module in parallel to a block dedicated to noise generation, thus the critical path of the circuit is not affected and the overhead of resources is limited. In that case, the most significant cost concerns are the power consumption, which can introduce an overhead of up to 5 times larger [54,55,57].

- Electromagnetic analysis [62]: This category of attacks relies on leaked electromagnetic radiation. It is usually assimilated to the class of power analysis attack, under the rough assumption that electromagnetic radiation is highly correlated (or proportional) to power consumption. Therefore, attacks and defense mechanisms are developed accordingly to the one of power analysis, using electromagnetic probes, instead of an instrument to measure the current.
- Differential fault analysis [63]: This type of attack has the purpose of discovering secrets by introducing faults while the system is running normally. The corresponding protection mechanisms typically involve architectural redundancy approaches and techniques for concurrent error detection.

Confirmations of the severity of SCAs can be found, for example, in [66]. It analyzes the resistance of ARM TrustZone with respect to Side-Channel leakages and reports several violations of security due to fault attacks [67–69], cache attacks, [7–10], and EMA [11–13].

According to the key aspects to be addressed for the development of a secure module, the security partitioning model illustrated in Figure 2 should be applied recursively to each layer and to each element of the chain involved in the construction of the TEE by dividing it into both trusted and non-trusted components. The results of this approach converge in the definition of a more accurate partitioning model that is represented in Figure 4. Such a model counts four abstraction layers that are, respectively from the top to the bottom, the protocol level, the software architecture level, the hardware micro-architecture level, and the circuit level. Each of these levels is characterized by the separation between security-related and non-security-related resources and a specific interface security policy between them. In addition, the interconnection between each level and the higher one is characterized by a specific class of SCA. Respectively, from the top to the bottom, they are interpreter-level attacks (such as fault attacks), timing analysis and cache attacks, and power analysis attacks (such as SPA, DPA, and CPA).

In conclusion, when designing a secure module to be integrated into a Trusted Environment (for its construction and maintenance), the three highlighted security aspects must be carefully addressed. According to the implementation methodology (only hardware, only software, or mixed hardware–software), the security module must be modeled according to the scheme presented in Figure 4, and the corresponding security mechanisms must be developed including the interface security policy and resistance measures against SCAs.

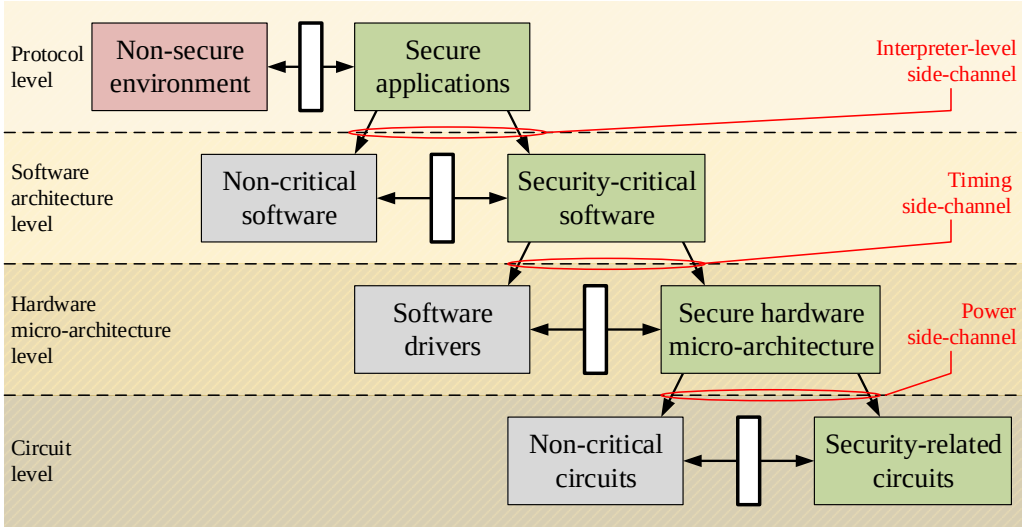

Figure 4. Recursive application of security partitioning model to the chain of elements forming the TEE.

5. Case Study: Cryptographic Hardware Coprocessor for the Secure Subsystem of Rhea GPP

Similarly to the AMD PSP, the decision for the isolation strategy of the TEE inside the Rhea GPP was to isolate it from the main processing unit(s) at the physical level by implementing a Security Subsystem dedicated to all and only the security services. The outline of the Rhea GPP developed in the framework of the EPI project is illustrated in Figure 5, while Figure 6 focuses on the internal architecture of the Trusted Environment (i.e., the Security Subsystem).

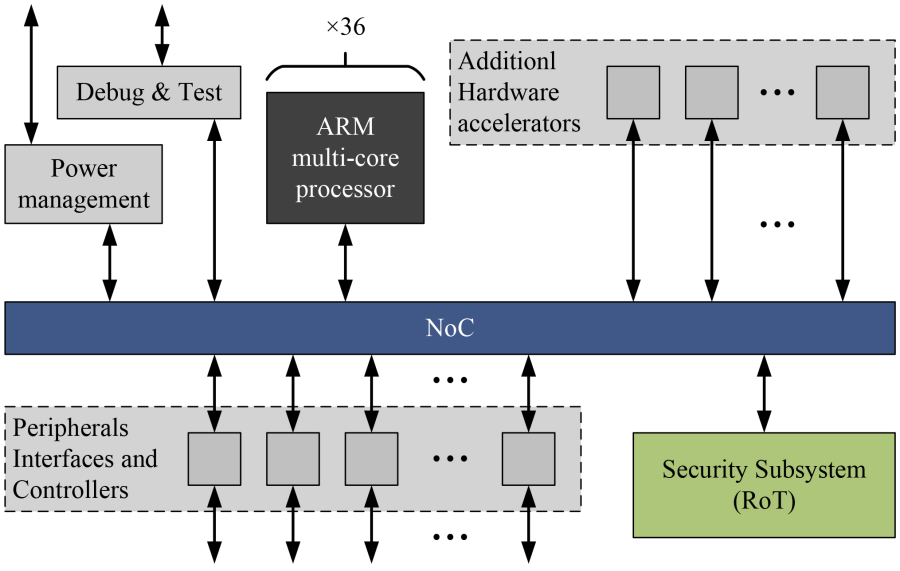

Figure 5. Simplified outline of Rhea GPP [14].

Our research group was involved in the development of the cryptographic hardware coprocessor (the blue box in Figure 6) that has the purpose of assisting the secure boot routine for enabling the Security Subsystem and providing high-performance security mechanisms [20]. The implemented module was named Crypto-Tile and offers cutting-edge security services [14]. According to the presented design methodology in Sections 5.1–5.3, we described the features integrated within the developed security module, respectively, at the security service level, interface security policy level, and physical implementation level.

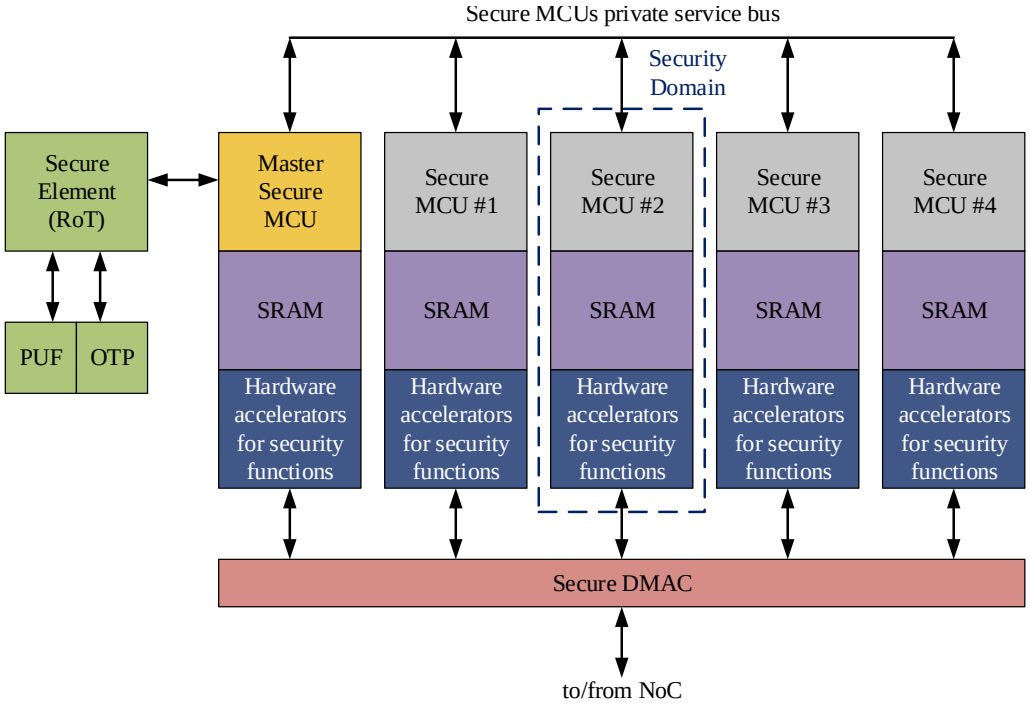

Figure 6. Outline of Security Subsystem within Rhea GPP [14].

5.1. Security Services

In order to support both the secure boot routine and high-performance security functions at run-time, we embedded in the Crypto-Tile four independent and specialized cryptographic coprocessors, with each one for a different family of cryptographic operations. With reference to the architecture outline of the Crypto-Tile that is shown in Figure 7, our module embeds:

- A coprocessor for symmetric-key cryptography based on the Advanced Encryption Standard (AES) algorithm [17]. This coprocessor offers:
  - Confidentiality, through the AES modes operation Electronic CodeBook (ECB), Cipher Block Chaining (CBC), Cipher FeedBack (CFB), Output FeedBack (OFB), and CounTeR (CTR) [70];
  - Authenticated Integrity, through the AES mode of operation Cipher-based Message Authentication Code (CMAC) [71];
  - Authenticated Encryption, through the AES modes of operation Counter with CBC-MAC (CCM) [72] and Galois Counter Mode (GCM) [73,74];
  - Disk encryption, through the AES mode of operation XEX encryption mode with Tweak and ciphertext Stealing (XTS) [75].

  For each of the listed modes of operation, the AES coprocessor supports both 128-bit and 256-bit keys, respectively providing a corresponding security level of 128 and 256 bits [43].
- A coprocessor for public-key cryptography based on Elliptic Curve Cryptography (ECC) [16]. This coprocessor offers:
  - Acceleration of computationally intensive arithmetic operations over elliptic curves, such as Point Addition and Point Doubling;
  - Generation of private keys, key pairs (both private and public), and derivation of corresponding public keys starting from private keys;
  - Authentication and non-repudiation, through the Elliptic Curve Digital Signature Algorithm (ECDSA) [76].

For each of the listed functions, the ECC coprocessor supports both 256-bit and 521-bit elliptic curves, in particular the NIST P-256 and the NIST P-521, that provide, respectively, a corresponding security strength of 128 and 256 bits [43].

- A coprocessor for hash functions based on the Secure Hash Algorithm (SHA) [18]. This coprocessor offers integrity through the algorithms SHA2 [77] and SHA-3 [78]. For both the SHA functions, the supported digest sizes are 224, 256, 384, and 512 bits, providing a corresponding security level of 128 (256-bit digest) and 256 (384-bit digest) bits [43].
- A coprocessor for Random Number Generation (RNG). This coprocessor offers:
  - Generation of high-entropy key-related material and seeds, through a True Random Number Generator (TRNG) [19];
  - Generation of key-related material, non-repeating values, and initialization vectors through a Cryptographically Secure Pseudo-Random Number Generator (CSPRNG) [79], which is seeded thanks to the TRNG.

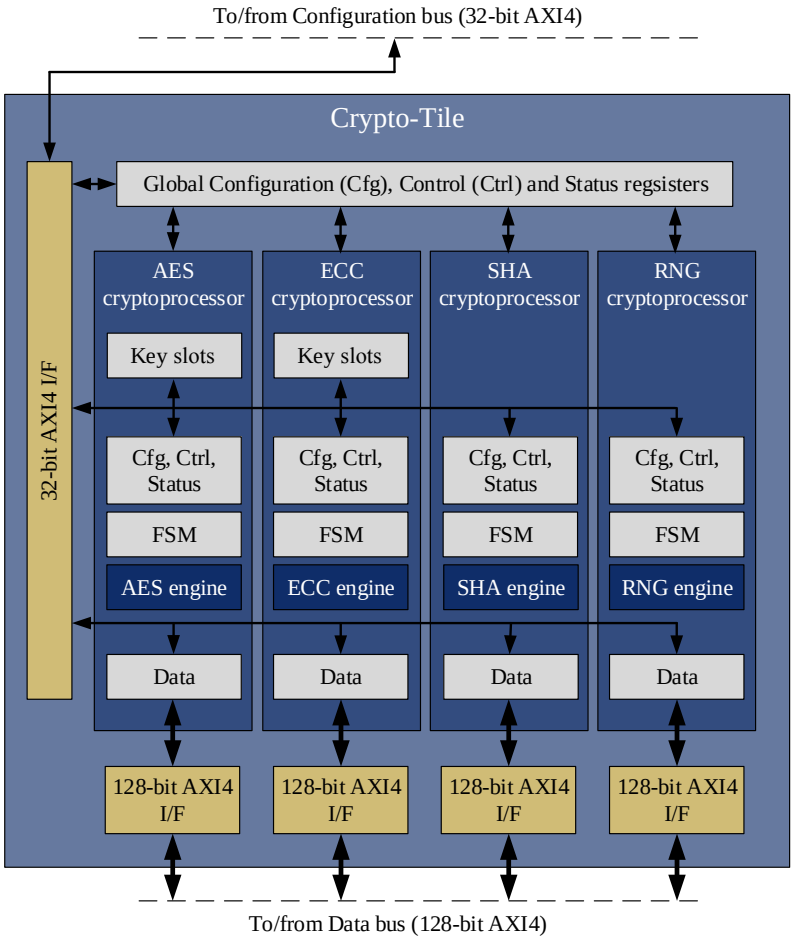

Figure 7. High-level architecture of Crypto-Tile [14].

The ECC and SHA cryptoprocessors support the secure boot routine to verify, respectively, the authenticity and the integrity of the first bootloader stage. Indeed, at the boot, the Secure Element in Figure 6 loads the software code from the OTP into the SRAM of the Master Secure MCU, enables the corresponding hardware acceleration unit (Crypto-Tile), and uses it to verify the integrity and the authenticity of the software code. Upon the success of this step, consequently, the Master Secure MCU performs similar operations by enabling the other Security Domains (triplets of Secure MCU, SRAM, and Crypto-Tile). After this process, the Secure Subsystem of Rhea GPP is fully enabled and can provide

run-time security services that are accelerated in the hardware by the instances of the Crypto-Tile and all the embedded cryptoprocessors.

All the implemented algorithms have been tested and verified at the functional level with the corresponding suite of test vectors provided by the NIST through the CAVP.

The implemented cryptographic coprocessors were developed to find the best solution in terms of the trade-off between performance, resource consumption, and security strength. Indeed, according to [43] the minimum level of security for long-term protection is 128 bits, and it corresponds to the usage of 128-bit keys for the AES algorithms, 256-bit elliptic curves for ECC functions and schemes, and 256-bit digests for the SHA2 and SHA-3 functions. In addition, we also integrated security levels for supporting protection in terms of Post-Quantum Cryptography (PQC) to enhance the security level of the Crypto-Tile. According to [80,81], this requires the usage of 256-bit keys for the AES algorithms and 384-bit digests in the hash functions. Since the logic resources to implement the hash functions for the generation of 224-bit and 512-bit digests are the same as the ones required for the generation of 256-bit and 384-bit digests, respectively, we integrated the generation mechanisms for all the digest sizes in the hash cryptographic coprocessors, improving its flexibility and without increasing the cost in terms of logic resources.

The synthesis results on a 7 nm standard-cell technology for the Crypto-Tile and the embedded cryptographic engines are illustrated in [14,16–19], in which we reported the area complexity expressed in Gate Equivalent (GE), the throughput, and the power consumption. In general, all the proposed engines outperform the existing solutions documented in the corresponding literature in terms of efficiency expressed as throughput per area and in terms of energy consumption. In particular, the AES engine, refs. [14,17], requires an area complexity of 56.01 kGE and supports a maximum frequency of 2.425 GHz, offering a throughput higher than 30 Gbps at the cost of 10.9 mW of dynamic power consumption. Concerning the engine for the public-key cryptographic functions, refs. [14,16], we opted for the ECC-based cryptography instead of the Rivest–Shadir–Adleman (RSA) cryptographic scheme, because it offers the same security level while requiring much lower resources [43]. Hence, the developed ECC engine occupies an area of 658.9 kGE, supports a maximum frequency of 1.525 GHz, and shows an average dynamic power consumption of 170 mW for signatures generation and 130 mW for signatures verification. It is interesting to signal that the ECC engine itself occupies about 50% of the area of the whole Crypto-Tile: this confirms that the choice of implementing public-key cryptographic functions based on the ECC algorithm was the best solution because the usage of the RSA-based counterpart would have required a much higher amount of resources, making the public-key coprocessor consume almost all the logic resource required to implement the Crypto-Tile. In addition, the ECC coprocessor has been designed with a specific interface to support higher-level ECC-based schemes such as the Elliptic-Curve Diffie–Hellman (ECDH) [82], the Elliptic-Curve Menezes–Qu–Vanstone (ECMQV) [82,83], and the Elliptic-Curve Integrated Encryption Scheme (ECIES) [83,84], which can be implemented in software. Thanks to the specialized mechanisms for the interface security policy (Section 5.2), the ECC coprocessor can accelerate in hardware the most computationally intensive parts of such schemes by assisting the dedicated secure software routines implementing them. Similarly, the SHA coprocessor, refs. [14,18], which occupies an area of 128.32 kGE, supports a maximum frequency of 3.725 GHz and shows average dynamic power consumption of 100 mW and 70 mW for the SHA2 and the SHA-3 functions, respectively, and is able to assist in hardware secure software routines dedicated to the HMAC scheme [85]. Finally, the RNG engine, refs. [14,19], requires an area complexity of 127.16 kGE and shows an average dynamic power consumption of 130 mW at the frequency of 4.325 GHz. This engine was characterized not only in terms of performance but it was also validated by using dedicated statistical test suites to evaluate the levels of entropy and randomness offered by the generated bitstreams [86]. All the tests were successful, satisfying the security requirements defined by the NIST and the Bundesamt für Sicherheit in der Informationstechnik (BSI) [19].

To evaluate the performance of the Crypto-Tile and its cryptographic engines, we also implemented a demoboard on a VCU128 board by Xilinx/AMD. The implemented system was aimed at emulating a Security Domain (Figure 6) by integrating the Crypto-Tile together with a 64-bit RISC-V processor CVA6 [87] as Secure MCU, a DDR controller to exploit the DDR4 memory onboard and implement the SRAM, and an AXI4 bus as system bus. The results showed that the Crypto-Tile is able to accelerate the security services and functions up to 400 times [14].

5.2. Interface Security Policy

According to the security functionalities included in the Crypto-Tile and described in Section 5.1, it is possible to individuate the main security-critical assets that require additional protection mechanisms when they are installed, used, and erased. We targeted the security-critical assets at three different levels:

- Global Crypto-Tile level: this level concerns the global configuration of the Crypto-Tile;
- Cryptographic Operation level: this level concerns data that are processed during the execution of the cryptographic operations;
- Key material level: this level concerns the keys, their installation in dedicated registers inside the Crypto-Tile, and their storage, access, and utilization statistics.

At the Global Crypto-Tile level, we integrated seal/unseal mechanisms that rely on a unique identifier that is configured during the secure boot process and the support of two authorization levels for access to the Crypto-Tile: a privileged level and an unprivileged level. For the cryptographic operation level, we integrated dedicated resources that strictly regulate the execution of each cryptographic operation and the access to the related data. For instance, Figure 8 shows the state diagram of the FSM that handles the cryptographic operations for the AES cryptoprocessor and manages access to the processed data. The auxiliary data such as initialization vectors can be provided in input only in the CONFIGURED state, while the input data to be encrypted can be written only in the RUNNING state, only once and only after the encryption of the previous data block is completed. In addition, the corresponding output data can be read only once and only in the RUNNING or ENDED state depending on the executed cryptographic operation. Similarly, the commands that can be provided to the AES cryptoprocessor strictly rely on the current state of the FSM.

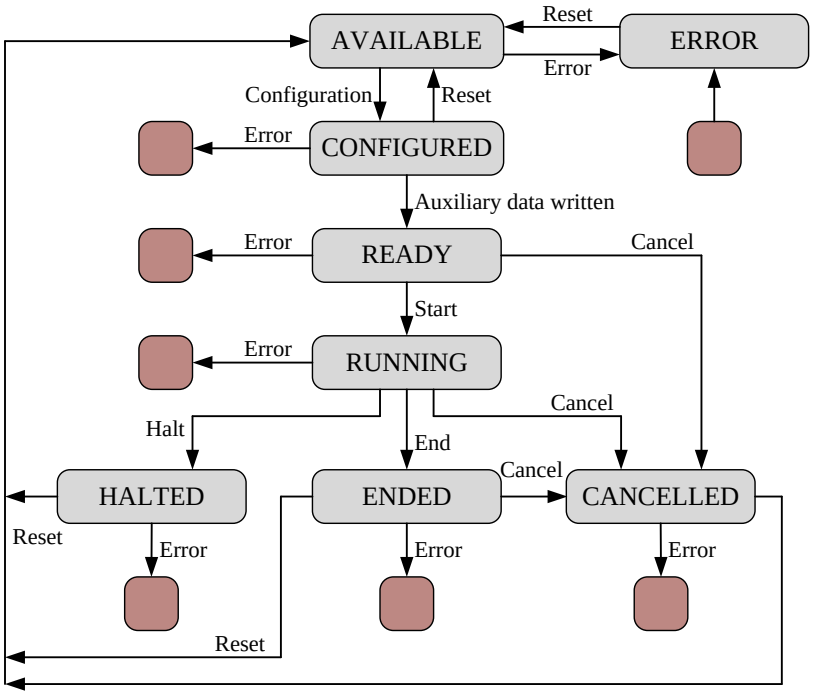

Figure 8. FSM regulating the data access during the execution of a cryptographic operation.

Concerning the key material level, the Crypto-Tile embeds dedicated seal/unseal mechanisms similar to the ones used for the Global Crypto-Tile level. A dedicated FSM for each key regulates its usage similarly to how it is illustrated for the cryptographic operations in Figure 8. Counters and configuration registers limit the key utilization according to the target operation. For instance, the usage of cryptographic keys can be configured for specific AES operations such as the GCM one or only for encryption processes. In such a case, the usage of that key in other AES operations different from the GCM one or in decryption processes is not permitted.

In addition, each level presents logging and alerting resources to check unauthorized access and interrupt all the ongoing operations in case of misusage, as well as triggers for two distinct levels of panic mode. According to the severity of the panic mode, the key material can be preserved to allow for retrieving it after the termination of the panic mode, or it can be flushed to prevent an attacker from taking it over.

5.3. Physical Implementation

Regarding the physical implementation and according to the security partitioning model in Figure 4, the Crypto-Tile has an impact on both the circuit level and the hardware micro-architecture level. For this reason, we provided our module with countermeasures for both timing and power analysis attacks.

We integrated a clock randomization mechanism to counteract timing attacks [54,55]. In this way, the execution of each cryptographic operation is unpredictable and an attacker cannot gain information by observing it. For instance, the usage of 128-bit or 256-bit keys in the AES operations determines a different execution time, because the former requires 10 processing rounds, while the latter requires 14 processing rounds. With the clock randomization approach, an attacker is no longer able to distinguish between the key size from the execution time of an AES operation. In addition, such a mechanism aids also in lowering the success rate of power analysis because DPA and CPA require the time alignment of the power traces [88]. It is worth pointing out that the utilization and the configuration of the clock randomization mechanism are strictly regulated by specific authorization procedures at the interface security policy level because its usage has direct effects on the security strength of the cryptographic coprocessors, hence also it must considered as a security-critical asset.

Concerning power analysis attacks, we integrated also other dedicated protections to make them ineffective. For instance, the ECC operations are very vulnerable to SPA, i.e., by observing just one power trace, an attacker can retrieve the secret key without additional statistical computations such as in DPA and CPA [16]. Thanks to the methodology described in [88], we were able to analyze and evaluate the resistance of our ECC coprocessor against Single Power Analysis (SPA) without the necessity of the physical chip and using gate-level simulations, concluding that the dedicated countermeasures are able to guarantee the required protection and defeat successfully these kinds of attacks [16]. Indeed, by exploiting the same gate-level simulations on the netlist that we used to measure the power consumption of the Crypto-Tile (Section 5.1), we were able to gather and analyze the power consumption profile of the engines during the execution of the cryptographic operations. As an example, in Figure 9 we report the power consumption profile of the ECC engine before the integration of dedicated security mechanisms against power analysis attacks.

As illustrated in Figure 9, without equipping the cryptographic ECC engine with specialized countermeasures, it is possible to distinguish if the bit of the key processed by the engine is 0 or 1, because according to its values, the engine shows a different power profile. Hence, an attacker can retrieve the value of the secret private key with a single SPA attack, i.e., by analyzing only a single power trace. In Figure 10, it can be noted how the power profile of the ECC engine changed after the integration of the dedicated countermeasures; after the modifications, the power profile is invariant with respect to the value of the key bits.

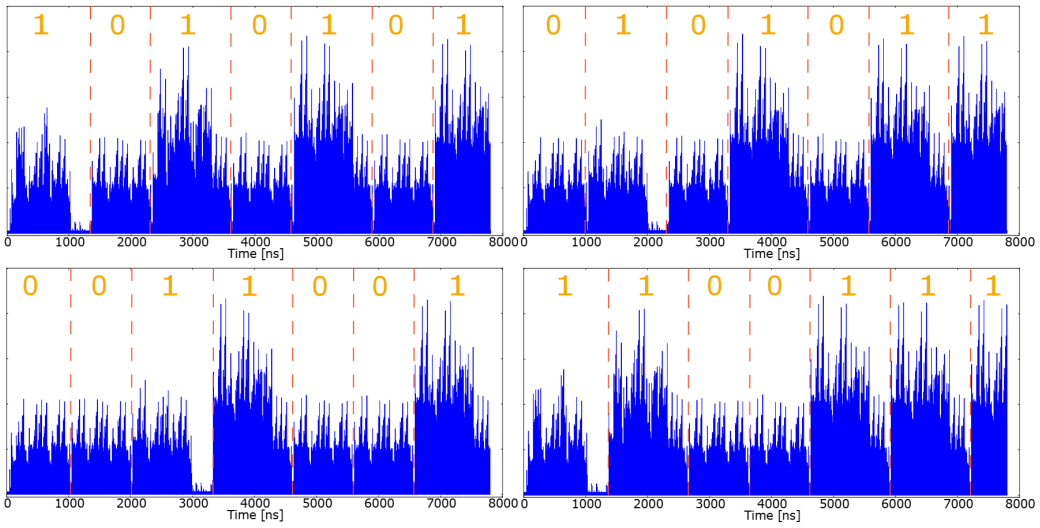

Figure 9. Power trace of the ECC engine acquired during the execution of a signature generation, before the integration of countermeasures against power analysis attacks. Extracted from [16].

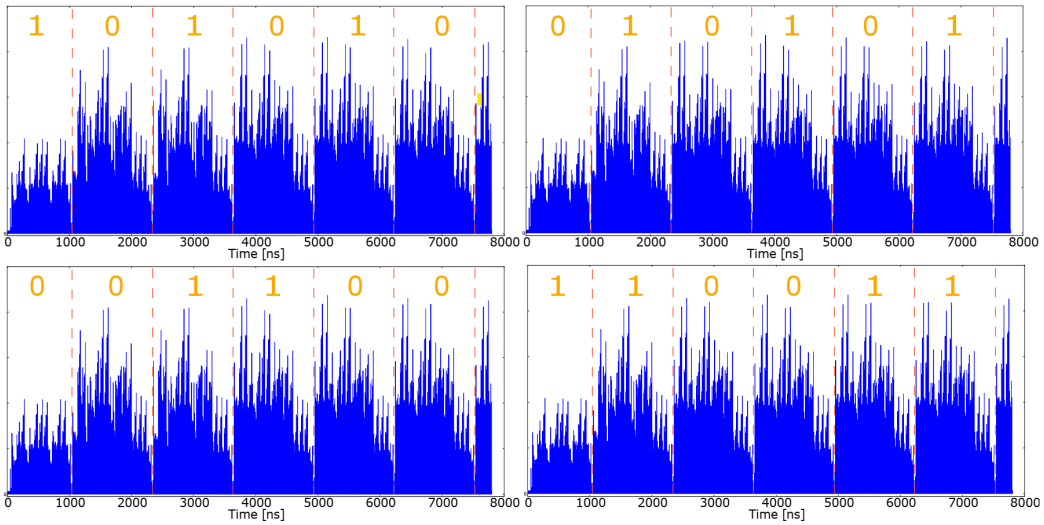

Figure 10. Power trace of the ECC engine acquired during the execution of a signature generation, after the integration of countermeasures against power analysis attacks. Extracted from [16].

5.4. Comparison with Existing TEEs

According to the results of the case study illustrated in Sections 5.1–5.3, we propose a brief comparison between the hardware-based security solutions analyzed so far and the proposed hardware coprocessor for the Rhea GPP. The isolation approach adopted in the Rhea chip is similar to the one proposed in the AMD PSP, Rambus CMRT family, and Synopsis tRoot products, that is the physical and complete isolation of the security functions in a dedicated zone of the chip that is separated from the main computational unit of the system. All the analyzed solutions include programmable units (e.g., microprocessors or microcontrollers), secure memories, and hardware accelerators for the most computationally intensive cryptographic algorithms. It is difficult to fully characterize the communication mechanism between the secure zone and the non-secure zone for each solution. The AMD PSP adopts a mailbox-based mechanism to exchange requests of security services from the main CPU (i.e., non-secure zone) to the secure zone, and a similar approach is adopted in the Rhea chip. Instead, this kind of information cannot be extracted for the Rambus CMRT family and the Synopsis tRoot. The solution proposed by ARM (i.e., the ARM TrustZone) and Intel (i.e., the Intel SGX) is opposite to the previous ones.

The execution of the secure software is allotted to the main processing unit(s) together with the normal software, and the isolation between the Normal world (non-secure zone) and the Secure world (secure zone) relies on software assets aided by dedicated hardware extensions: for instance, such software assets take the name of Enclave in the Intel SGX. On the one hand, this latter solution lowers the cost in terms of physical resources; on the other hand, the former approach guarantees a stronger isolation between the two zones. Indeed, as an example, the work in [89] demonstrates that Hardware-assisted Isolated Execution Environments (HIEEs) like the ARM TrustZone and the Intel SGX are susceptible to Denial-of-Service (DoS) and spoofing attacks. By contrast, such attacks cannot be performed against the AMD PSP and the Security Subsystem of the Rhea GPP because the physical isolation between the secure zone and the non-secure zone makes them immune.

6. Conclusions

In this work, we explored the critical aspects of designing robust and highly qualified security modules for Trusted Environments. We discussed the need for security modules that can withstand sophisticated attacks and ensure the integrity, confidentiality, and authenticity of sensitive information. Our research focused on developing a comprehensive design methodology and a set of metrics that can guide the creation of such security modules. The proposed methodology places a strong emphasis on the identification of potential vulnerabilities at different and various levels and the application of security principles in every aspect of the design process. We showed that our approach results in modules that are fully equipped to withstand both known and emerging threats, according to the presented case study. Furthermore, we introduced a set of metrics to assess the security modules. These metrics provide a structured way to evaluate the robustness, efficiency, and effectiveness of the modules, ensuring that they meet the stringent requirements of Trusted Environments. Our study has also highlighted the strict connection between such metrics through the case study. Indeed, it shows how the security services provided by the developed security module and the implementation platform (hardware or software) concur to the definition of security-critical assets, whose access must be regulated by means of dedicated and specialized procedures and resources at the interface security policy level.

In conclusion, our research offers a valuable contribution to the field of TEE security. We have provided a systematic methodology and a set of metrics to guide the design and evaluation of security modules. By following these guidelines, developers and security professionals can create solutions that are more resilient and trustworthy. As Trusted Environments become increasingly vital in today's interconnected world, the principles outlined in this paper are instrumental in achieving the highest level of security and reliability.

Author Contributions: Conceptualization, L.C., P.N. and S.D.M.; methodology, L.C.; validation, L.C., P.N. and S.D.M.; investigation, L.C.; data curation, L.C. and P.N.; writing—original draft preparation, L.C.; writing—review and editing, L.C., P.N., S.D.M. and S.S.; supervision, S.S.; project administration, S.S.; funding acquisition, S.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partially funded by the European Union's Horizon 2020 research and innovation program European Processor Initiative (EPI), under the grant agreement No. 101036168 (EPI SGA2), and by the Italian Ministry of University and Research (MUR) through the project CN4 - CN00000023 of the program Recovery and Resilience Plan (PNRR), under the grant agreement no. I53C22000720001, and through the project FoReLab of the program "Departments of Excellence".

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AES	Advanced Encryption Standard
AI	Artificial Intelligence
AXI	Advanced eXtensible Interface
BSI	Bundesamt für Sicherheit in der Informationstechnik
CAVP	Cryptographic Algorithm Validation Program
CBC	Cipher Block Chaining
CFB	Cipher FeedBack
CCM	Counter with CBC-MAC
CMAC	Cipher-based Message Authentication Code
CMRT	CryptoManager Root-of-Trust
CoT	Chain-of-Trust
CPA	Correlation Power Analysis
CPU	Central Processing Unit
CTR	CounTeR
CSPRNG	Cryptographically Secure Pseudo-Random Number Generator
DoS	Denial-of-Service
DPA	Differential Power Analysis
ECB	Electronic CodeBook
ECC	Elliptic Curve Cryptography
ECIES	Elliptic Curve Integrated Encryption Scheme
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
ECMQV	Elliptic Curve Menezes-Qu-Vanstone
EM	Electromagnetic Emission
EMA	Electromagnetic Emission Analysis
EPC	Enclave Page Cache
EPCM	Enclave Page Cache Map
EPI	European Processor Initiative
FSM	Finite State Machine
GCM	Galois Counter Mode
GE	Gate Equivalent
GPIO	General Purpose Input/Output
GPP	General Purpose Processor
HIEE	Hardware-assisted Isolated Execution Environment
HMAC	Hash-based Message Authentication Code
HPC	High-Performance Computing
HSM	Hardware Security Module
HW	Hardware
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet-of-Things
LUT	Look-Up Table
MAC	Message Authentication Code
MCU	Master Control Unit
NIST	National Institute of Standard and Technology
OFB	Output FeedBack
OTP	One-Time Programmable
PQC	Post-Quantum Cryptography
PSP	Platform Security Processor
RISC	Reduced Instruction Set Computer
RNG	Random Number Generator
RoT	Root-of-Trust
SCA	Side-Channel Attack
SGX	Software Guard eXtensions
SoC	System-on-Chip
SHA	Secure Hash Algorithm

SPA	Simple Power Analysis
SRAM	Secure RAM
SW	Software
TEE	Trusted Execution Environment
TZASC	TrustZone Address Space Controller
TZMA	TrustZone Memory Adapter
UART	Universal Asynchronous Receiver-Transmitter
UEFI	Unified Extensible Firmware Interface
XTS	XEX encryption mode with Tweak and ciphertext Stealing

References

1. Duo, W.; Zhou, M.; Abusorrah, A. A Survey of Cyber Attacks on Cyber Physical Systems: Recent Advances and Challenges. *IEEE/CAA J. Autom. Sin.* **2022**, *9*, 784–800. [CrossRef]
2. Igbekele Emmanuel, O.; Ekele Victoria, C.; Omonigho Efeoghene, I.; Nwachuwku Praise, C. Overview of Recent Cyberattacks: A Systematic Review. In Proceedings of the 2023 International Conference on Science, Engineering and Business for Sustainable Development Goals (SEB-SDG), Omu-Aran, Nigeria, 5–7 April 2023; Volume 1, pp. 1–8. [CrossRef]
3. Li, Y.; Liu, Q. A comprehensive review study of cyber-attacks and cyber security; Emerging trends and recent developments. *Energy Rep.* **2021**, *7*, 8176–8186. [CrossRef]
4. Sabt, M.; Achemlal, M.; Bouabdallah, A. Trusted Execution Environment: What It is, and What It is Not. In Proceedings of the 2015 IEEE Trustcom/BigDataSE/ISPA, Helsinki, Finland, 20–22 August 2015; Volume 1, pp. 57–64. [CrossRef]
5. Hoang, T.T.; Duran, C.; Serrano, R.; Sarmiento, M.; Nguyen, K.D.; Tsukamoto, A.; Suzaki, K.; Pham, C.K. Trusted Execution Environment Hardware by Isolated Heterogeneous Architecture for Key Scheduling. *IEEE Access* **2022**, *10*, 46014–46027. [CrossRef]
6. Crocetti, L.; Di Rienzo, R.; Verani, A.; Baronti, F.; Roncella, R.; Saletti, R. A Novel and Robust Security Approach for Authentication, Integrity, and Confidentiality of Lithium-ion Battery Management Systems. In Proceedings of the 2023 IEEE 3rd International Conference on Industrial Electronics for Sustainable Energy Systems (IESES), Shanghai, China, 26–28 July 2023. [CrossRef]
7. Zhang, N.; Sun, K.; Shands, D.; Lou, W.; Hou, Y.T. TruSpy: Cache Side-Channel Information Leakage from the Secure World on ARM Devices. *IACR Cryptol. ePrint Arch.* **2016**, *2016*, 980.
8. Lipp, M.; Gruss, D.; Spreitzer, R.; Maurice, C.; Mangard, S. Armageddon: Cache attacks on mobile devices. In Proceedings of the 25th USENIX Security Symposium (USENIX Security 16), Austin, TX, USA, 10–12 August 2016; pp. 549–564.
9. Zhang, X.; Xiao, Y.; Zhang, Y. Return-Oriented Flush-Reload Side Channels on ARM and Their Implications for Android Devices. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. Association for Computing Machinery, Vienna, Austria, 24–28 October 2016; p. 858–870.
10. Green, M.; Rodrigues-Lima, L.; Zankl, A.; Irazoqui, G.; Heyszl, J.; Eisenbarth, T. AutoLock: Why Cache Attacks on ARM Are Harder Than You Think. In Proceedings of the 26th USENIX Security Symposium USENIX Security 17), Vancouver, BC, Canada, 16–18 August 2017; pp. 1075–1091.
11. Bukasa, S.K.; Lashermes, R.; Le Bouder, H.; Lanet, J.L.; Legay, A. How TrustZone could be bypassed: Side-channel attacks on a modern system-on-chip. In Proceedings of the Information Security Theory and Practice. WISTP, Heraklion, Greece, 28–29 September 2017; Lecture Notes in Computer Science (LNCS); Springer: Berlin/Heidelberg, Germany, 2018; Volume 10741, pp. 93–109.
12. Longo, J.; De Mulder, E.; Page, D.; Tunstall, M. SoC It to EM: ElectroMagnetic Side-Channel Attacks on a Complex System-on-Chip. In Proceedings of the 17th International Workshop on Cryptographic Hardware and Embedded Systems (CHES), Saint-Malo, France, 13–16 September 2015; Lecture Notes in Computer Science (LNCS); Springer: Berlin/Heidelberg, Germany, 2015; pp. 620–640.
13. Majéric, F.; Bourbao, E.; Bossuet, L. Electromagnetic security tests for SoC. In Proceedings of the 2016 IEEE International Conference on Electronics, Circuits and Systems (ICECS), Monte Carlo, Monaco, 11–14 December 2016; pp. 265–268. [CrossRef]
14. Nannipieri, P.; Crocetti, L.; DI Matteo, S.; Fanucci, L.; Saponara, S. Hardware Design of an Advanced-Feature Cryptographic Tile within the European Processor Initiative. *IEEE Trans. Comput.* **2023**, 1–14. [CrossRef]
15. Zulberti, L.; Di Matteo, S.; Nannipieri, P.; Saponara, S.; Fanucci, L. A Script-Based Cycle-True Verification Framework to Speed-Up Hardware and Software Co-Design: Performance Evaluation on ECC Accelerator Use-Case. *Electronics* **2022**, *11*, 3704. [CrossRef]
16. Di Matteo, S.; Baldanzi, L.; Crocetti, L.; Nannipieri, P.; Fanucci, L.; Saponara, S. Secure Elliptic Curve Crypto-Processor for Real-Time IoT Applications. *Energies* **2021**, *14*, 4676. [CrossRef]
17. Nannipieri, P.; Di Matteo, S.; Baldanzi, L.; Crocetti, L.; Zulberti, L.; Saponara, S.; Fanucci, L. VLSI Design of Advanced-Features AES Cryptoprocessor in the Framework of the European Processor Initiative. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2022**, *30*, 177–186. [CrossRef]
18. Nannipieri, P.; Bertolucci, M.; Baldanzi, L.; Crocetti, L.; Di Matteo, S.; Falaschi, F.; Fanucci, L.; Saponara, S. SHA2 and SHA-3 accelerator design in a 7 nm technology within the European Processor Initiative. *Microprocess. Microsyst.* **2021**, *87*, 103444. [CrossRef]

19. Nannipieri, P.; Di Matteo, S.; Baldanzi, L.; Crocetti, L.; Belli, J.; Fanucci, L.; Saponara, S. True Random Number Generator Based on Fibonacci-Galois Ring Oscillators for FPGA. *Appl. Sci.* **2021**, *11*, 3330. [CrossRef]
20. Baldanzi, L.; Crocetti, L.; Di Matteo, S.; Fanucci, L.; Saponara, S.; Hameau, P. Crypto Accelerators for Power-Efficient and Real-Time on-Chip Implementation of Secure Algorithms. In Proceedings of the 2019 26th IEEE International Conference on Electronics, Circuits and Systems (ICECS), Genoa, Italy, 27–29 November 2019; pp. 775–778. [CrossRef]
21. Nannipieri, P.; Di Matteo, S.; Zulberti, L.; Albicocchi, F.; Saponara, S.; Fanucci, L. A RISC-V Post Quantum Cryptography Instruction Set Extension for Number Theoretic Transform to Speed-Up CRYSTALS Algorithms. *IEEE Access* **2021**, *9*, 150798–150808. [CrossRef]
22. Crocetti, L.; Di Matteo, S.; Nannipieri, P.; Fanucci, L.; Saponara, S. Design and Test of an Integrated Random Number Generator with All-Digital Entropy Source. *Entropy* **2022**, *24*, 139. [CrossRef] [PubMed]
23. Di Matteo, S.; Gerfo, M.L.; Saponara, S. VLSI Design and FPGA Implementation of an NTT Hardware Accelerator for Homomorphic SEAL-Embedded Library. *IEEE Access* **2023**, *11*, 72498–72508. [CrossRef]
24. ARM Security Technology—Building a Secure System Using TrustZone Technology. Technical Report, ARM, 2005–2009. Available online: https://developer.arm.com/documentation/PRD29-GENC-009492/c (accessed on 19 September 2023).
25. ARM. Arm TrustZone Technology. Available online: https://developer.arm.com/ip-products/security-ip/trustzone/trustzone-system-ip (accessed on 19 September 2023).
26. ARM. TrustZone for Cortex-A. Available online: https://developer.arm.com/ip-products/security-ip/trustzone/trustzone-for-cortex-a (accessed on 19 September 2023).
27. ARM. TrustZone for Cortex-M. Available online: https://developer.arm.com/ip-products/security-ip/trustzone/trustzone-for-cortex-m (accessed on 19 September 2023).
28. Khalid, F.; Masood, A. Hardware-Assisted Isolation Technologies: Security Architecture and Vulnerability Analysis. In Proceedings of the 2020 International Conference on Cyber Warfare and Security (ICCWS), Islamabad, Pakistan, 20–21 October 2020; pp. 1–8. [CrossRef]
29. Intel Software Guard Extensions (Intel SGX)—Key Management on the 3rd Generation Intel® Xeon® Scalable Processor. Technical Report, Intel. 2019. Available online: https://builders.intel.com/docs/networkbuilders/intel-software-guard-extensions-intel-sgx-key-management-on-the-3rd-generation-intel-xeon-sc-1617436024.pdf (accessed on 19 September 2023).
30. Intel Software Guard Extensions (Intel SGX)—Developer Guide. Technical Report, Intel. 2018. Available online: https://www.intel.com/content/dam/develop/public/us/en/documents/intel-sgx-developer-guide.pdf (accessed on 19 September 2023).
31. Hoekstra, M.; Lal, R.; Pappachan, P.; Phegade, V.; Del Cuvillo, J. Using Innovative Instructions to Create Trustworthy Software Solutions. In Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy, Tel-Aviv, Israel, 23–24 June 2013; Volume 11, pp. 2487726–2488370.
32. McKeen, F.; Alexandrovich, I.; Berenzon, A.; Rozas, C.V.; Shafi, H.; Shanbhogue, V.; Savagaonkar, U.R. Innovative Instructions and Software Model for Isolated Execution. In Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy, Tel-Aviv, Israel, 23–24 June 2013; Article 10.
33. Anati, I.; Gueron, S.; Johnson, S.; Scarlata, V. Innovative Technology for CPU Based Attestation and Sealing. In Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy, Tel-Aviv, Israel, 23–24 June 2013; Article 13.
34. BIOS and Kernel Developer's Guide (BKDG) for AMD Family 16h Models 30h-3Fh Processors. Technical Report, AMD, 2013–2016. Available online: https://www.amd.com/system/files/TechDocs/52740_16h_Models_30h-3Fh_BKDG.pdf (accessed on 19 September 2023).
35. Rambus. Hardware Root of Trust: Everything You Need to Know. Available online: https://www.rambus.com/blogs/hardware-root-of-trust/ (accessed on 19 September 2023).
36. Rambus. Introducing the Rambus CryptoManager Root of Trust (CMRT). Available online: https://www.rambus.com/blogs/introducing-the-rambus-cryptomanager-root-of-trust-cmrt/?lang=zh-hans (accessed on 19 September 2023).
37. Synopsys. Synopsys tRoot Vx Hardware Secure Modules. Available online: https://www.synopsys.com/dw/ipdir.php?ds=security-troot-hw-secure-module (accessed on 19 September 2023).
38. Champagne, D.; Lee, R.B. Scalable architectural support for trusted software. In Proceedings of the HPCA-16 2010 The Sixteenth International Symposium on High-Performance Computer Architecture. IEEE, Bangalore, India, 9–14 January 2010; pp. 1–12.
39. Szefer, J.; Lee, R.B. Architectural support for hypervisor-secure virtualization. *ACM SIGPLAN Not.* **2012**, *47*, 437–450. [CrossRef]
40. Suh, G.E.; Clarke, D.; Gassend, B.; Van Dijk, M.; Devadas, S. AEGIS: Architecture for tamper-evident and tamper-resistant processing. In Proceedings of the 17th annual international conference on Supercomputing, San Francisco, CA, USA, 23–27 June 2003; pp. 160–171.
41. Keller, E.; Szefer, J.; Rexford, J.; Lee, R.B. Nohype: Virtualized cloud infrastructure without the virtualization. In Proceedings of the 37th Annual International Symposium on Computer Architecture, Saint-Malo, France, 19–23 June 2010; pp. 350–361.
42. Costan, V.; Lebedev, I.; Devadas, S. Sanctum: Minimal hardware extensions for strong software isolation. In Proceedings of the 25th USENIX Security Symposium (USENIX Security 16), Austin, TX, USA, 10–12 August 2016; pp. 857–874.
43. *SP 800-57 Part 1 Rev. 5*; Recommendation for Key Management: Part 1—General; NIST: Gaithersburg, MD, USA, 2020.

44. Tiri, K. Side-Channel Attack Pitfalls. In Proceedings of the 2007 44th ACM/IEEE Design Automation Conference, San Diego, CA, USA, 4–8 June 2007; pp. 15–20.
45. Li, Y.; Chen, M.; Wang, J. Introduction to side-channel attacks and fault attacks. In Proceedings of the 2016 Asia-Pacific International Symposium on Electromagnetic Compatibility (APEMC), Shenzhen, China, 18–21. May 2016; Volume 1, pp. 573–575.
46. Spreitzer, R.; Moonsamy, V.; Korak, T.; Mangard, S. Systematic classification of side-channel attacks: A case study for mobile devices. *IEEE Commun. Surv. Tutor.* **2017**, *20*, 465–488. [CrossRef]
47. Yan, M.; Gopireddy, B.; Shull, T.; Torrellas, J. Secure hierarchy-aware cache replacement policy (SHARP): Defending against cache-based side channel attacks. In Proceedings of the 2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA), Toronto, ON, Canada, 24–28 June 2017; pp. 347–360. [CrossRef]
48. Cho, J.; Kim, T.; Kim, T.; Shin, Y. Real-Time Detection on Cache Side Channel Attacks using Performance Counter Monitor. In Proceedings of the 2019 International Conference on Information and Communication Technology Convergence (ICTC), Jeju, Republic of Korea, 16–18 October 2019; pp. 175–177. [CrossRef]
49. Kim, H.; Yoon, H.; Shin, Y.; Hur, J. Cache Side-Channel Attack on Mail User Agent. In Proceedings of the 2020 International Conference on Information Networking (ICOIN), Barcelona, Spain, 7–10 January 2020; pp. 236–238. [CrossRef]
50. Kaushik, P.; Majumdar, R. Timing attack analysis on AES on modern processors. In Proceedings of the 2017 6th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), Noida, India, 20–22 September 2017; pp. 462–465. [CrossRef]
51. Rani, R.; Venkateswarlu, S. Security against Timing Analysis Attack. *Int. J. Electr. Comput. Eng. (IJECE)* **2015**, *5*, 759–764. [CrossRef]
52. Jayasinghe, D.; Ragel, R.; Elkaduwe, D. Constant time encryption as a countermeasure against remote cache timing attacks. In Proceedings of the 2012 IEEE 6th International Conference on Information and Automation for Sustainability, Beijing, China, 27–29 September 2012; pp. 129–134. [CrossRef]
53. Seo, S.C.; Kim, H. SCA-Resistant GCM Implementation on 8-Bit AVR Microcontrollers. *IEEE Access* **2019**, *7*, 103961–103978. [CrossRef]
54. Hettwer, B.; Das, K.; Leger, S.; Gehrer, S.; Güneysu, T. Lightweight Side-Channel Protection using Dynamic Clock Randomization. In Proceedings of the 2020 30th International Conference on Field-Programmable Logic and Applications (FPL), Gothenburg, Sweden, 31 August – 4 September 2020; pp. 200–207. [CrossRef]
55. Jayasinghe, D.; Ignjatovic, A.; Parameswaran, S. SCRIP: Secure Random Clock Execution on Soft Processor Systems to Mitigate Power-based Side Channel Attacks. In Proceedings of the 2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), Westminster, CO, USA, 4–7 November 2019; pp. 1–7. [CrossRef]
56. Zhou, T.; Zhu, Y.; Jing, N.; Nan, T.; Li, W.; Peng, B. Reliable SoC Design and Implementation of SHA-3-HMAC Algorithm with Attack Protection. In Proceedings of the 2020 IEEE International Conference on Smart Cloud (SmartCloud), Washington, DC, USA, 6–8 November 2020; pp. 88–93. [CrossRef]
57. Ma, J.W.; Guan, X.G.; Zhou, T.; Sun, T. A new countermeasure against side channel attack for HMAC-SM3 hardware. In Proceedings of the 2017 IEEE 12th International Conference on ASIC (ASICON), Guiyang, China, 25–28 October 2017; pp. 327–330. [CrossRef]
58. Belaid, S.; Bettale, L.; Dottax, E.; Genelle, L.; Rondepierre, F. Differential power analysis of HMAC SHA-2 in the Hamming weight model. In Proceedings of the 2013 International Conference on Security and Cryptography (SECRYPT), Reykjavik, Iceland, 29–31 July 2013; pp. 1–12.
59. He, Z.; Wu, L.; Zhang, X. High-speed Pipeline Design for HMAC of SHA-256 with Masking Scheme. In Proceedings of the 2018 12th IEEE International Conference on Anti-counterfeiting, Security, and Identification (ASID), Xiamen, China, 9–11 November 2018; pp. 174–178. [CrossRef]
60. Oku, D.; Yanagisawa, M.; Togawa, N. A Robust Scan-based Side-channel Attack Method against HMAC-SHA-256 Circuits. In Proceedings of the 2017 IEEE 7th International Conference on Consumer Electronics—Berlin (ICCE-Berlin), Berlin, Germany, 3–6 September 2017; pp. 79–84. [CrossRef]
61. Kabin, I.; Dyka, Z.; Kreiser, D.; Langendoerfer, P. Unified field multiplier for ECC: Inherent resistance against horizontal SCA attacks. In Proceedings of the 2018 13th International Conference on Design Technology of Integrated Systems In Nanoscale Era (DTIS), Taormina, Italy, 9–12 April 2018; pp. 1–4. [CrossRef]
62. Werner, F.T.; Djordjević, A.R.; Zajić, A.G. A Compact Probe for EM Side-Channel Attacks on Cryptographic Systems. In Proceedings of the 2019 IEEE International Symposium on Antennas and Propagation and USNC-URSI Radio Science Meeting, Atlanta, GA, USA, 7–12 July 2019; pp. 613–614. [CrossRef]
63. Patranabis, S.; Chakraborty, A.; Mukhopadhyay, D.; Chakrabarti, P.P. Fault Space Transformation: A Generic Approach to Counter Differential Fault Analysis and Differential Fault Intensity Analysis on AES-Like Block Ciphers. *IEEE Trans. Inf. Forensics Secur.* **2017**, *12*, 1092–1102. [CrossRef]
64. Randolph, M.; Diehl, W. Power Side-Channel Attack Analysis: A Review of 20 Years of Study for the Layman. *Cryptography* **2020**, *4*, 15. [CrossRef]

65. Chong, K.S.; Shreedhar, A.; Lwin, N.K.Z.; Kyaw, N.A.; Ho, W.G.; Wang, C.; Zhou, J.; Gwee, B.H.; Chang, J.S. Side-Channel-Attack Resistant Dual-Rail Asynchronous-Logic AES Accelerator Based on Standard Library Cells. In Proceedings of the 2019 Asian Hardware Oriented Security and Trust Symposium (AsianHOST), Xi'an, China, 16–17 December 2019; pp. 1–7. [CrossRef]
66. Benhani, E.M.; Bossuet, L.; Aubert, A. The Security of ARM TrustZone in a FPGA-Based SoC. *IEEE Trans. Comput.* **2019**, *68*, 1238–1248. [CrossRef]
67. Lipp, M. Cache Attacks and Rowhammer on ARM. Master's Thesis, Institute for Applied Information Processing and Communications, Graz University of Technology, Gratz, Austria, 2016. Available online: https://mlq.me/download/master_thesis.pdf (accessed on 19 September 2023).
68. Carru, P. Attack TrustZone with Rowhammer. 2017. Available online: https://grehack.fr/data/2017/slides/GreHack17_Attack_TrustZone_with_Rowhammer.pdf (accessed on 19 September 2023).
69. Tang, A.; Sethumadhavan, S.; Stolfo, S. CLKSCREW: Exposing the Perils of Security-Oblivious Energy Management. In Proceedings of the 26th USENIX Security Symposium (USENIX Security 17), Vancouver, BC, Canada, 16–18 August 2017; USENIX Association: Berkeley, CA, USA, 2017; pp. 1057–1074.
70. *FIPS 197*; Announcing the Advanced Encryption Standard (AES). NIST: Gaithersburg, MD, USA, 2001.
71. *SP 800-38B*; Recommendation for Block Ciphers Modes of Operation: The CMAC Mode for Authentication. NIST: Gaithersburg, MD, USA, 2005.
72. *SP 800-38C*; Recommendation for Block Ciphers Modes of Operation: The CCM Mode for Authentication and Confidentiality. NIST: Gaithersburg, MD, USA, 2007.
73. *SP 800-38D*; Recommendation for Block Ciphers Modes of Operation: Galois/Counter Mode (GCM) and GMAC. NIST: Gaithersburg, MD, USA, 2007.
74. *IEEE Std 802.1AE-2018 (Revision of IEEE Std 802.1AE-2006)*; IEEE Standard for Local and metropolitan area networks-Media Access Control (MAC) Security. IEEE: Piscataway, NJ, USA, 2018. [CrossRef]
75. *SP 800-38E*; Recommendation for Block Ciphers Modes of Operation: The XTS-AES Mode for Confidentiality on Storage Devices. NIST: Gaithersburg, MD, USA, 2010.
76. *FIPS 186-4*; Digital Signature Standard (DSS). NIST: Gaithersburg, MD, USA, 2013.
77. *FIPS 180-4*; Secure Hash Standard (SHS). NIST: Gaithersburg, MD, USA, 2015.
78. *FIPS 202*; glsSHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. NIST: Gaithersburg, MD, USA, 2015.
79. Baldanzi, L.; Crocetti, L.; Falaschi, F.; Belli, J.; Fanucci, L.; Saponara, S. Digital Random Number Generator Hardware Accelerator IP-Core for Security Applications. In *Applications in Electronics Pervading Industry, Environment and Society. ApplePies 2019*; Lecture Notes in Electrical Engineering (LNEE); Springer: Berlin/Heidelberg, Germany, 2020; Volume 627, pp. 117–123. [CrossRef]
80. Moody, D.N. Post-Quantum Cryptography: NIST's Plan for the Future Status Update on Elliptic Curves and Post-Quantum Crypto. Available online: https://csrc.nist.gov/csrc/media/projects/post-quantum-cryptography/documents/pqcrypto-2016-presentation.pdf (accessed on 19 September 2023).
81. Moody, D.N. Update on the NIST Post-Quantum Cryptography Project. Available online: https://csrc.nist.gov/CSRC/media/Presentations/NIST-Status-Update-on-Elliptic-Curves-and-Post-Qua/images-media/moody-dustin-threshold-crypto-workshop-March-2019.pdf (accessed on 19 September 2023).
82. Barker, E.; Chen, L.; Keller, S.; Roginsky, A.; Vassilev, A.; Davis, R. *Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography*; Technical Report; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2017.
83. Jablon, D. IEEE P1363 standard specifications for public-key cryptography. In Proceedings of the IEEE NIST Key Management Workshop CTO Phoenix Technologies Treasurer, Gaithersburg, MD, USA, 1–2 November 2001; pp. 1–26.
84. *IEEE 1363a*; Standard Specifications for Public-Key Cryptography—Amendment 1: Additional Techniques. IEEE: Piscataway, NJ, USA, 2004.
85. *FIPS 198-1*; The Keyed-Hash Message Authentication Code (HMAC). NIST: Gaithersburg, MD, USA, 2008.
86. Crocetti, L.; Nannipieri, P.; Di Matteo, S.; Fanucci, L.; Saponara, S. Review of Methodologies and Metrics for Assessing the Quality of Random Number Generators. *Electronics* **2023**, *12*, 723. [CrossRef]
87. Group, O. CVA6: A Linux-Capable RISC-V CPU. Available online: https://www.hackster.io/news/cva6-a-linux-capable-risc-v-cpu-299a40a5f871 (accessed on 19 September 2023).
88. Crocetti, L.; Baldanzi, L.; Bertolucci, M.; Sarti, L.; Carnevale, B.; Fanucci, L. A simulated approach to evaluate side-channel attack countermeasures for the Advanced Encryption Standard. *Integration* **2019**, *68*, 80–86. [CrossRef]
89. Zhang, F.; Zhang, H. SoK: A study of using hardware-assisted isolated execution environments for security. In Proceedings of the Hardware and Architectural Support for Security and Privacy, Seoul, Republic of Korea, 18 June 2016; pp. 1–8.