LOCARD

DELIVERABLE

D4.6 State of the Art Report on Trusted Computing and Trusted Execution Environment

Project Acronym:	LOCARD	
Project title:	Lawful evidence collecting and continuity platform development	
Grant Agreement No.	832735	
Website:	http://locard.eu/	
Contact:	info@locard.eu	
Version:	1.0	
Date:	31 October 2019	
Responsible Partner:	ARC	
Contributing Partners:		
Reviewers:	Miltiadis Anastasiadis Agusti Solanas	
Dissemination Level:	Public	X
	Confidential – only consortium members and European Commission Services	
	Classified Information: RESTREINT UE (Commission Decision 2015/444/EC)	

This project has received financial support from the European Union Horizon 2020 Programme under grant agreement no. 832735.

Revision History

Revision	Date	Author	Organization	Description
0.1	10/09/2019	Nikolaos Koutroumpouxos, Christoforos Ntantogian	ARC	Initial draft
0.2	02/10/2019	Christos Xenakis, Christoforos Ntantogian	ARC	Version for review
1.0	31/10/2019	Christos Xenakis, Christoforos Ntantogian	ARC	Final version

Table of Contents

List of Tables

List of Figures

1 Executive Summary

The purpose of this document is to present and analyse research endeavours and the state of the art relevant to trusted computing in the two most prevalent instantiations (i) Trusted Execution Environments and (ii) Trusted Platform Modules. Trusted computing will be able to provide the required guarantees for the LOCARD project as it can support trusted evidence collection which can be used in a court of law, the cornerstone of LOCARD. Trusted computing enhances the ability of a device in the context of confidentiality integrity and accountability by providing a separate execution environment which is isolated from the main environment which could be compromised. This way, evidence collection and data logging can be offloaded to such an environment so that the outcomes of these procedures is considered secured, trusted and complete. In this report, we outline the main benefits, shortcomings and ongoing research around the topic of trusted computing. The aim is to provide a complete documentation that covers trusted computing and its applications in the context of LOCARD. This report will be usable by other work packages as an input reference when trusted computing is required as a security enhancement for the task in hand.

2 Introduction

A basic target of LOCARD is to provide end-to-end security to its entities using trusted computing technologies both at the client and server-side. This solution will enhance the overall security of the architecture by offering a trustworthy hardware-based root of trust that will be leveraged and extended to secure sensitive data in-transfer and in-rest. In particular, for client devices, the Trusted Execution Environment (TEE) will be used to securely store and manage encryption/decryption keys along with the identity attributes of users. In this way, TEE guarantees secure authentication, key management and security of identity attributes. On the server-side, TEE will be used to deliver a system environment where blockchain nodes can execute private Smart Contracts in an isolated manner (i.e., non-authorized nodes will not be able to view the execution code and intermediate results) decoupling in this way the privacy of Smart Contract execution from the protection of the privacy of the entire blockchain.

Furthermore, the proliferation of data that is a side-effect of using blockchains, poses a privacy risk since there is an aggregation of large amounts of possibly sensitive data. Although cryptographic protocols (e.g. secure multiparty computation and zero-knowledge proofs) offer attractive solutions for privacy in such a scenario, they have yet to achieve the required maturity level to run general-purpose computations efficiently and thus, to be widely deployed. A promising alternative is the use of TEEs for running smart contracts, which could enhance the security and privacy of the used data. By using a TEE, one does not have to trust the host system of the enclave, which runs the blockchain node and participates in its protocols. Thus, a server-side TEE solution can boost the security properties of blockchains and the related smart contracts. It is worth noting that Microsoft recently has filled two patents that fuse TEE with blockchain technology. [1], [2]

On the client-side, a TEE can be used to enhance the security of the private keys stored in the user device and all related sensitive operations that run on it. [3] Due to the lack of authentication in blockchain, there is a need to design an "identity wallet" that would protect identity attributes at the client-side as well as to propose new protocols for seamless authentication in blockchain technology. A prominent technology that delivers TEE in mobile devices is ARM TrustZone. [4]

This deliverable is within the context of the task T4.7 that studies end-to-end security through the exploitation of Trusted Execution Environments (TEEs). More precisely, there is an investigation of trusted computing technologies, that is TEEs and Trusted Platform Modules (TPMs), which includes background information, security issues, implementations and technologies that use such solutions. Furthermore, we research blockchains and smart contracts, their applications and the possibility of executing Smart Contracts in trusted computing environments using technologies such as Intel SGX. The aim of this deliverable is to provide a complete image of the state of the art when it comes to TEEs, TPMs, and blockchain.

3 Trusted Execution environment

A trusted execution environment on a computing device allows for the invocation of trusted applications without obstructing the normal computing environment. In addition, trusted applications running on this environment can securely interact with each other in such a way that other non-trusted entities cannot access the trusted applications and their resources. [5] Therefore, the trusted execution environment provides two basic services on a high level:

- Isolated execution of trusted applications with restricted interprocess communication.
- Unobstructed execution of normal applications within the same computing environment.

3.1 Trusted Execution Environment Background

3.1.1 GlobalPlatform

GlobalPlatform is a standardization body that took it upon itself to define the ideal hardware security guard that will be used on mobile phones. In order to do so, it took into consideration some security requirements that were specified by the Open Mobile Terminal Platform which aimed to define a "trusted environment"; GlobalPlatform improved upon this specification and finally presented the final product that was named "trusted execution environment". [6] [7] This product immediately caught the attention of major organizations that backed the development of this technology and soon the trusted execution environment became a well-defined security component that exists in most mobile phone and computer processors today.

GlobalPlatform provides a wide range of specifications that define every aspect of the trusted execution environment, from the hardware level to the final API and how it should be implemented. It also provides certifications for compliance with the specification and organizes conferences and workshops in order to spread the knowledge for trusted execution environments. The specification that GlobalPlatform created is widely accepted as the de facto standard with most vendors implementing it in their devices with minor exceptions.

3.1.2 Trusted Execution Environment High-Level Security Requirements

The primary purpose of the TEE is to protect and isolate its resources from other environments present on the ecosystem that the TEE belongs to. This isolation is enforced through hardware mechanisms that are not controllable by non-secure environments. Apart from the software protection, the TEE should also be protected against some physical attacks such as side-channel attacks, but intrusive techniques are not in the scope of the TEE security. Within the context of hardware security, the ecosystem that hosts the TEE should remove or disable any debug hardware interfaces with direct access to the TEE. The final security requirement that the specification makes is that the TEE must be instantiated through a secure boot process, that produces a chain of trust and ensures that the devices are properly booted without any tampering. The secure boot process provides guarantees to the authenticity and the integrity of the device and all its components.

3.1.3 TEE Hardware Architecture

The trusted execution environment (from now on referenced as TEE) is a component of a computing system, and as such, it should be defined where it belongs. The TEE typically resides either within the System-on-Chip (SoC) or as an external security processor that connects directly to the system bus. This is shown in **Figure 1-1.** where the blue entities represent these two possible points of TEE instalment. When the TEE is an external entity, the division between the two worlds (trusted and untrusted) is obvious, as the TEE is an independent entity that interacts with the system. In the other case, where the TEE is part of the SoC, then the hardware separation is not so apparent. More specifically, the TEE can either share the SoC CPU and use it securely only when needed or have separate CPU core(s) just for the trusted operations. [8] Some hardware implementations are shown in **Figure 1-2.**

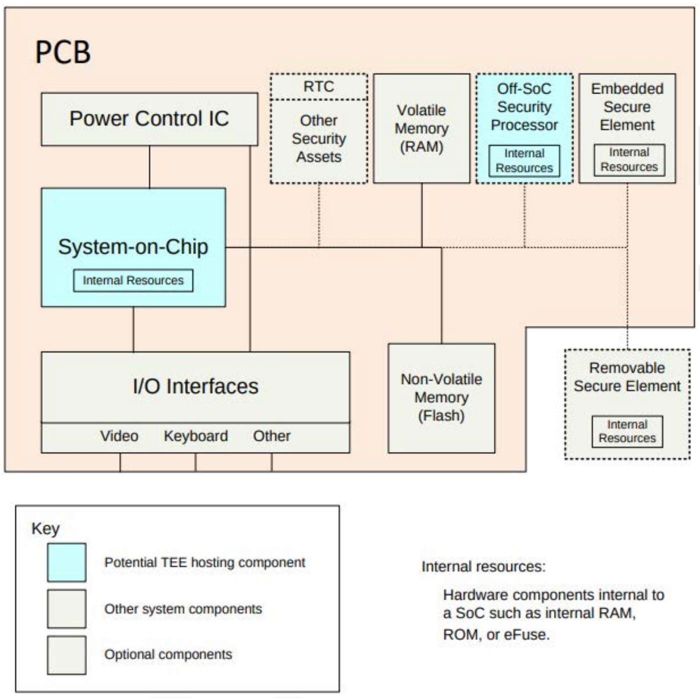

Figure 1: Potential TEE Hosting Components [9]

The TEE manages three classes of resources [9]:

- **In-package resources:** These resources exist only within the TEE and are considered protected from any adversary. Communication between these resources is also considered physically secure and as such, there is no need to encrypt it.
- **Off-package, cryptographically protected resources:** In order to extend the in-package resources, a TEE can utilize external resources that can be deemed trustworthy with the usage of cryptography. The protection of these memory areas is achieved through proven cryptographic methods, with the TEE being the only entity that holds the decryption capabilities. Although these resources are cryptographically secure, since they exist out of the package, the communication between the TEE and these resources can be vulnerable to man-in-the-middle (MITM) attacks. These off-package resources include the trusted replay-protected external non-volatile memory areas and the trusted volatile memory areas
- **Exposed or partially exposed resources:** These resources are both off-package and not cryptographically protected. Some examples are trusted DRAM-based buffers, trusted screen frame

stores, and input/output (I/O) devices. All the aforementioned examples require the isolation provided by the TEE but do not require any encryption.

These resources can and will be shared between the TEE and the Rich Execution Environment (REE), so as to provide the intended functionality. As expected, any TEE resource can only be accessed by the TEE due to the direct (hardware) and indirect (off-package encryption) isolation that it provides. Some of these resources can be accessed by the REE through API entry points or other services that the TEE exposes through the TEE Client API. [10] On the other hand, REE resources are always accessible by both the TEE and the REE, with the TEE having direct access to any memory location of the REE [11]. Depending on the implementations shown in **Figure 1-2**, the resource sharing scheme is different; for example, PCB A has complete isolation of the TEE and does not facilitate any resources from the REE.

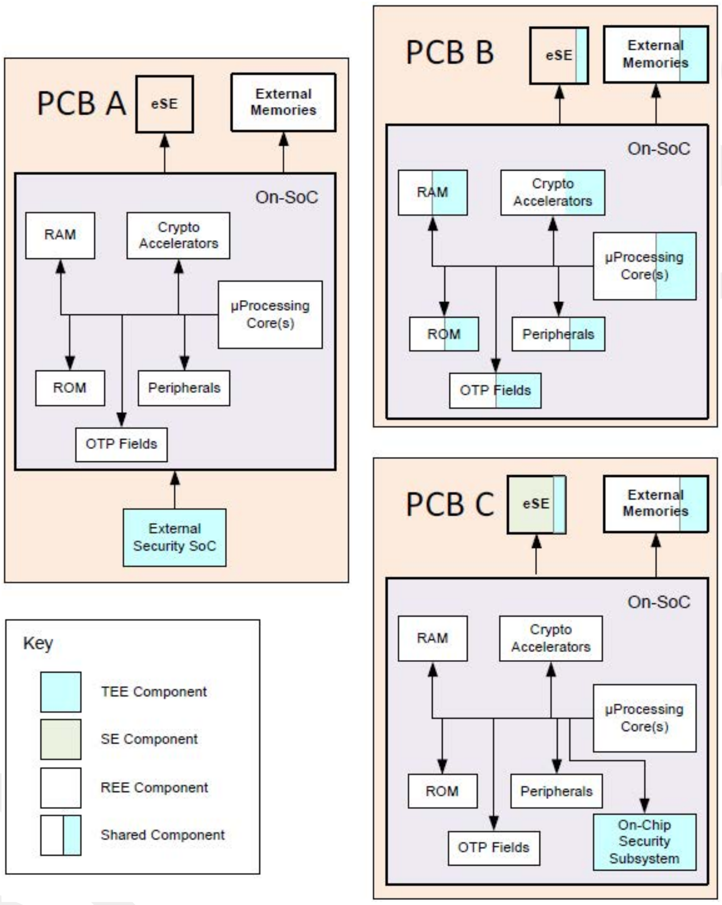

Figure 2: Possible Architectures of TEE Enabled Systems [9]

From an abstract point of view, the TEE aims to have two worlds, the trusted and the untrusted world. When the untrusted world is in charge, then the system operations and resources should be considered untrusted and accessible by anyone. The untrusted world hosts the REE (Rich Execution Environment), which provides an operational environment for the host operating system and applications to run on. The trusted world, on the other hand, exists with the purpose of hosting sensitive functionalities that need a high level of trust and authorization. Only specific components have access to the hooks provided by the trusted world, and they need to be authenticated first before the operation. The trusted world is not just a fence that protects its components, it provides a usable and programmable environment for developers to deploy their sensitive applications. It has its own libraries with the aim of reducing the secure code base and assure that the TEE is used only when absolutely needed.

The world change takes place with the help of an underlying call manager that is responsible for handling requests for transitions between the worlds. This manager is either a supervisor or in the case of ARM TrustZone, the secure monitor. When an untrusted application needs to run sensitive functionalities that exist within the TEE, it issues a system call to the manager. If the information provided by the application is correct (authentication data, authorization data, the ID of the called trusted application, etc.), then the manager proceeds with the world change and gives control to the trusted world. On the other side, the trusted world validates the command is received and executes the appropriate commands. When the specified functionality is finished, then it issues a new system call to the underlying manager with all the relevant information (process ID of calling application, results, etc.) and the manager returns control to the untrusted process along with said data.

The low-level communication between the REE and the TEE is handled by hardware drivers that are responsible for properly handling the messages to be exchanged, assessing their authenticity, validity, feasibility, etc. As these drivers are only accessible by privileged users, communication services are installed that provides a low-level API for applications to use. The TEE Client API [10], the specification of this service, provides functions to accomplish a rich communication between the two entities. Furthermore, there is a definition for the TEE Protocol Specification, that adds another layer of abstraction to the TEE Client API, this way, developers can provide a higher-level API for applications to utilize the TEE within. According to the specification of GlobalPlatform, the TEE TA Debug API and the TEE Management Framework Specification are both using the TEE Protocol Specification layer.

3.1.4 TEE Software Architecture

Secure Boot

As the core component of a TEE is trust, it first must be verified for its integrity. Only trusted and verified images should be run on the device to maintain the desired security level. The secure boot scheme uses cryptographic checks in order to assert the integrity of all the secure world images to be loaded. It is based on a concept that is called “chain of trust”, which is a waterfall model that aims to propagate trust through multiple layers of software. More specifically, the root of trust in this chain is a hardware-bound cryptographic key that is used to check the integrity and authenticity of the first piece of software to be executed. After the first software is authenticated it then can be trusted to authenticate the next piece of software and so on, creating a chain of trusted software images, each authenticated by the previous trusted software. This chain leads up to the TEE images, which are always authenticated before they are trusted to perform any sensitive operation.

Secure OS

From the software perspective, the TEE requires some basic building blocks which are: the secure operating system, trusted applications, the rich operation system, normal applications, the message broker (hypervisor/ARM trusted firmware) and the management framework. More specifically, the secure operating system runs on the secure world and it is responsible for providing a basic set of operating system functionalities while also preserving a high level of security so as to conform with the GlobalPlatform specification. The secure OS practically is the implementation of the secure part of the TEE specification, as it should employ any available technology (hardware and/or software) to protect its assets from the normal world, while also isolate trusted applications from each other. Some implementations of secure operating systems include Google's Trusty, Qualcomm's QSEE, Trustonic's Kinibi and Linaro's optee_os, with the most widely deployed and used being QSEE and Kinibi. All of the aforementioned secure operating systems are based on the ARM TrustZone technology, there are other operating systems that are based on other technologies like the Intel SGX and AMD SME/SEV.

Trusted Applications

The second component of the TEE software architecture that we are going to explore, are the trusted applications (TA) or trustlet as they are called for convenience. Trustlets are pieces of software that run as applications within the trusted operating system and are developed either by first-party or third-party developers. Each trustlet is run isolated from each other, with their resources protected in a manner that they cannot be attacked neither from the normal world nor from another trustlet. The primitives used to protect trustlets from each other are software/cryptography-based, whereas the cross-world protection also utilizes hardware-based isolation technologies like TrustZone. The only entity that can possibly access another trustlet's resources is the secure OS kernel in a manner that is identical to the normal operating system kernel accessing its user-space applications.

The trustlets are powerful applications that are compiled from memory sensitive languages (C) and are also capable of accessing hardware memory addresses in some implementations. Moreover, they are software that is considered as trusted to do sensitive functions for the normal and the secure world. Because of this, most TEE implementations have chosen to secure the confidentiality, integrity, and authenticity of the trustlets in multiple ways. The authenticity and the integrity of the trustlets are protected by a signature on the hash of the trustlet binary that can be verified using a certificate chain that is rooted in a hardware bound public key. This way, an attacker cannot load a trustlet that is not verified by the hardware vendor. The confidentiality and integrity of the trustlet, or more correctly, the confidentiality and the integrity of the trustlet data is protected with a cryptographically secure storage that also has TEE binding functionalities which assert that the data cannot be cloned to other TEEs.

Secure Monitor

The Secure Monitor mode is responsible for the switch from one world to the other. It acts as a middleman between the secure world and the normal world and handles all requests between them. In most use cases, the functionality is similar to the context switch of any operating system, that is, it should correctly and safely store the initial context state, give control to the target context and finally restore the initially stored state. There are strict constraints when it comes to transitions from the normal world to the secure world, whereas the other way there is more flexibility. That is because the secure world holds sensitive information and has high privileges, so it is only logical that userspace normal world application access attempts are put under the microscope. This does not happen the other way around, something that may initially sound normal, but as we will see later, it rises a class of vulnerabilities that leverage the trusted world to attack the normal world.

As evident, the security of the secure monitor is of crucial importance, as it acts as a gatekeeper to the highly privileged and sensitive secure world. It is advised that the execution of the secure monitor code is always run with interrupts disabled to minimize any malicious attempt to manipulate the execution flow of this sensitive piece of code. Moreover, the secure monitor is always run on the secure world, it only provides some external interfaces for the normal world to use, but it completely runs within the context of the secure world so as to provide the maximum possible level of security.

Normal World

The normal world is the normal operating system that is controlled by the end-user. There are a few minor changes so that the normal world will incorporate the secure world in its design. As the focus of this work is the trusted world, we are going to focus only on the components of the normal world that are relevant. There are three common implementations for the normal world to access the secure world; by directly accessing the driver and using a middleware library, with the most common method being the latter.

TEE Driver

The drivers are kernel modules that are responsible for providing hooks for the userspace applications to execute privileged functions. Controlling any peripheral, requires kernel privileges that a simple application does not have, in order to enable usage of these peripherals, the kernel developer installs a set of drivers that aim to enable the usage of installed peripherals by normal users. This is the case for the secure monitor. Because the system calls to initialize a world switch, there is always a secure driver in the system so that the normal world can easily communicate with the secure world.

The driver is a simple block device, that can be written to and read from. Its purpose is to handle any trusted world access intent and either propagate it to the secure monitor or block it. Furthermore, it manages any allocated shared memory between the two worlds so as to provide unobstructed communication. Finally, and more importantly, the driver populates the system calls that are sent to the secure monitor to initialize a context switch. These calls are very important as they provide the largest attack surface to the trusted world as we will see.

TEE Library

The purpose of a library is to provide readily available functionality for other programs to use without the complexity of implementing said functionality. A core component of the TEE is the libraries and APIs it encloses within it that are divided across the two worlds. The secure world library provides the **trusted core framework API** that defines basic data structures while also managing trustlet instances, inter-trustlet communication, and memory. The **trusted storage API** has functions that store securely persistent or transient data objects in the trusted storage of the TEE. There is also, the **cryptographic operations API** that as its name implies, has hooks that map to all the supported cryptographic operations which include: symmetric and asymmetric cryptographic functions, hashing functions, MAC functions, authenticated encryption functions, and key derivation functions. Finally, there is the **time API** that provides time-related functions, the **arithmetic API** that manages arithmetic operations between TEE specific data types and the **peripherals API** which handles any external trusted peripherals that might exist.

On the other side, the normal world library aims to ease the process of calling trustlets and the communication between the two worlds. This library is much simpler than its trusted counterpart, as it does not implement any software logic, one can think of it as the window from the normal world to the trusted world. It has functions for context management, session management and command invocation which should be called whenever a normal program wishes to delegate a sensitive operation to a trustlet. Furthermore, it provides operations for the allocations and registration of shared memory spaces that will be used by both normal and trusted applications to exchange data. Finally, it defines common data structures and constants in order to provide a common language that the two worlds can use to communicate.

3.2 ARM TrustZone

ARM TrustZone [12] [13] [14], according to the official terminology is a system-wide approach to security for a wide array of client and server computing platforms, including handsets, tablets, wearable devices and enterprise systems." [8] As a feature, TrustZone can be seen as a special kind of hardware supported virtualization of CPU state, memory, interrupt signals and I/O data, with the purpose of isolation. This virtualization technology enables each physical CPU core to provide an abstraction of two virtual ones, orthogonally dividing the process state into two logical realms, or worlds as they are named: The normal world of the REE and the secure world of TEE. TrustZone can be seen as a technology that enables any part of the system to be made secure, while also sharing it with the normal world.

3.2.1 NS Bit Memory Separation

To achieve world separation, the TrustZone technology employs some unique techniques with the most distinctive one being the NS bit. This bit is defined in the AMBA3 AXI bus protocol specification [178] and it is an additional 33d bit to the 32-bit address space that is provided to both the secure and non-secure world and it is appended to all read and write bus messages of the system. If the bit is set to high, then the message is non-secure while a low NS bit signifies that the message is to be accessed only by the secure world.

This extra bit works fine within the boundaries of the ARM made components, but when a peripheral that does not "speak" ARM receives such a message then it will probably not work properly. To this end, the AMBA3 specification defines a separate bridge from the AXI bus that has the NS bit to the APB bus [179] that can work properly with any peripheral. This bridge acts as a gatekeeper that will not accept non-secure messages when the NS bit is set to low. This way, the peripherals are separated from the normal world when sensitive operations are executed (e.g. when a user types a password, the keyboard is bound to the secure world).

As with any address space, including those without TrustZone technology, care must be taken to ensure that the 33-bit address space is used in such a way that data remains coherent in all of the locations that it is stored, otherwise, data corruption may result. If we consider the case where a Secure world entity wants to access a non-secure entity that is cached. A design may implement either of the following choices:

- The secure world makes a non-secure access to the non-secure world.
- The secure world makes a secure access to the non-secure world and the non-secure world accepts the secure transaction. The slave treats these accesses as non-secure.

In the second design, the hardware must support address space aliasing. In this aliased memory system, the same memory location appears as two distinct locations in the address map, one secure and one non-secure. As a result, it is possible to have multiple values representing the same data present in the cache simultaneously. For modifiable data this aliasing causes coherency problems; if one copy of the data is modified while the other exists in the cache you will have versions of the data, but both will be different. System designers must be aware of potential data coherency problems and must take steps to avoid them.

3.2.2 Processor Architecture

The current ARM processors that support the ARM TrustZone technology belong mainly in the Cortex-A family of ARM and a lightweight version of TrustZone can also be found in ARMv8-M, an indicative list of these processors is:

- ARM1176JZ(F)-S™ processor
- Cortex™-A8 processor
- Cortex-A9 processor
- Cortex-A9 MPCore™ processor
- Cortex-M23 processor
- Cortex-M33 processor

Each of the physical processor cores in these designs provides two virtual cores, one considered Non-secure and the other Secure, and a mechanism to robustly context switch between them, known as monitor mode. The value of the NS bit that is added to all bus messages is derived automatically from the virtual core that made the call. This way there is a seamless integration of the TrustZone technology in the system where non-secure cores can access only non-secure data and secure calls can see everything.

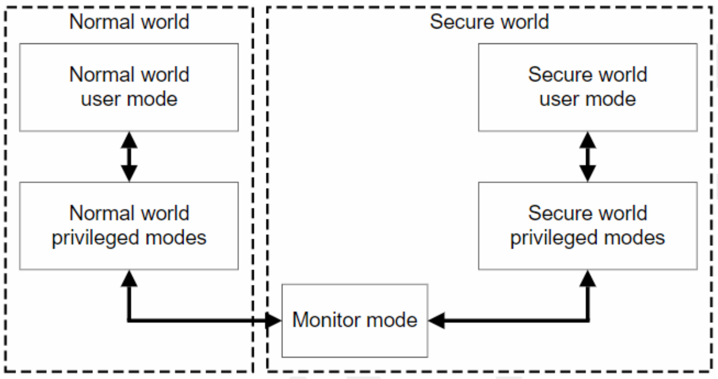

Figure 3: The Relationship Between the Normal World, the Secure World and the Monitor Mode [13]

3.2.3 Securing the level one memory system

The memory infrastructure outside of the core separates the system into two worlds, and a similar partitioning needs to be applied within the core to separate the data used and stored within the components of the level one (L1) memory system.

The basic component of the level one memory system is the memory management unit (MMU) which translates the virtual addresses that are seen by the software to actual physical addresses that are bound to the hardware. This address translation is managed by a software-based solution that dictates the exact mapping that should take place, the cacheability of the addresses and access permissions. Within a TrustZone processor, the hardware provides two virtual MMUs, one for each virtual processor. This enables each world to have a local set of translation tables, giving them independent control over their virtual address to physical address mappings and total isolation between the two memory spaces.

The ARMv6 and ARMv7 L1 translation table descriptor MMU software includes an NS field which is used by the secure-world to determine the value of the NS-bit to use when accessing the physical memory locations associated with the secure-world MMU. The non-secure world hardware completely ignores this field, and the memory access is always made with the NS-bit enabled. This design enables the Secure virtual processor to access either Secure or Non-secure memory and denies the non-secure virtual processor any possibility of accessing secure world memory.

Furthermore, ARM processors tag entries in a table named “Translation Lookaside Buffer” (TLB), which caches the results of address translations, with the identity of the context that made this translation. This allows entries from both contexts (secure and non-secure) to coexist within the TLB and enhance the performance of memory translations by reducing the stress of the MMU.

Another important entity of the level one memory system is the cache. It is a desirable feature of any high-performance design to support data of both security states in the caches. This removes the need for a cache flush when switching between worlds and enables high-performance software to communicate over the world boundary. To enable this the level one memory system, and where applicable level two and beyond, processor caches have been extended with an additional tag bit which records the security state of the transaction that accessed the memory. This tag will mandate who can access the corresponding entry and what he can do to it. Any non-locked down cache line can be evicted to make space for new data, regardless of its security state. It is possible for a secure line load to evict a non-secure line, and for a non-secure line load to evict a secure line.

3.2.4 Secure Interrupts

ARM processors provide an interrupt model that is composed of two kinds of interrupts, FIQ and IRQ. FIQ interrupts provide a method of performing a fast interrupt in a digital data processor having the capability of handling more than one interrupt is provided. When a fast interrupt request (FIQ) is received a flag is set and the program counter and condition code registers are stored on a stack. At the end of the interrupt servicing routine, the return from interrupt instructions retrieves the condition code register which contains the status of the digital data processor and checks to see whether the flag has been set or not. If the flag is set it indicates that a fast interrupt was serviced and therefore only the program counter is unstacked. On the other hand, regular interrupt requests (IRQ) are handled in the priority they come and are always put on hold when an FIQ arrives.

The ability to trap IRQ and FIQ directly to the monitor, without the intervention of code in either world, allows for the creation of a flexible interrupt model for secure interrupt sources. This way the monitor can route these interrupts to the correct world. Together with a security-aware interrupt controller, it allows for a design that can provide secure interrupt sources that cannot be manipulated by the normal world software. The recommendation from ARM is to use IRQ as a non-secure world interrupt source and FIQ for the secure world. IRQs are the most common interrupts in common operating systems, this way, applying FIQs in the secure world will allow for TrustZone integration with minimal changes. If the processor is running the correct virtual core when an interrupt occurs there is no switch to the monitor and the interrupt is handled locally in the current world. If the core is in the other world when an interrupt occurs the hardware traps to the monitor, the monitor software causes a context switch and jumps to the restored world, at which point the interrupt is taken.

To provide the aforementioned interrupt functionality there should exist three interrupts table in the system. One for each entity, the normal world, the secure world and the monitor.

3.2.5 Multiprocessor Support for TrustZone

The ARM architecture allows for configurations with up to four processors in a cluster. These processors can be set-up in either in Symmetric Multi-Processing (SMP) mode or in Asymmetric Multi-Processing (AMP) mode.

When a processor is executing in SMP mode the cluster's Snoop Control Unit (SCU) will transparently keep data that is shared across the SMP processors coherent in the L1 data cache. When a processor is executing in AMP mode the executing software must manually maintain memory coherency if it is needed.

These multiprocessor systems may implement the ARM Security Extensions, giving each processor in the cluster a separate TrustZone implementation to work with. The ARM processor which currently implements both the multiprocessor features and the security features is the Cortex-A9 MPCore processor. With each of the processors within the multiprocessor cluster having a separate TrustZone implementation, they will have a normal world and a secure world per processor. This gives a four-processor cluster a total of eight virtual processors and twenty-four translation tables, each with independent control over their MMU configuration.

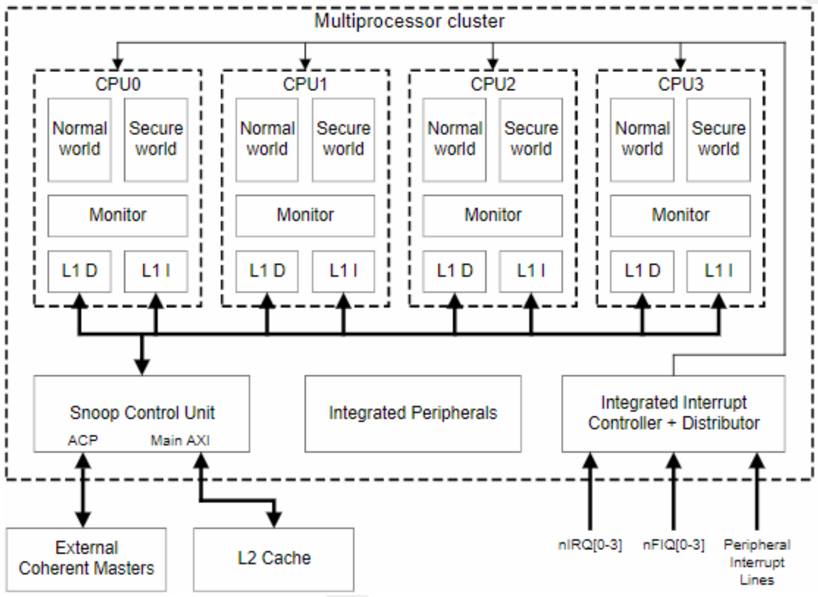

Figure 4: TrustZone Multiprocessor Support [13]

3.2.6 ARM TrustZone Software Architecture

Until now we have seen all the hardware components that are provided by the TrustZone technology. But hardware alone, cannot provide enough functionality for the deployment of a TEE as we have defined it in the previous chapters. That is why secure software TrustZone aware software should be installed in a system that wishes to use the TrustZone technology to create the secure and non-secure world isolation of a TEE. The ARM architecture specifies the open component of security extensions which can be used to create a custom Secure world software environment to meet their requirements. This section presents some of the possibilities that a software architect might want to consider when designing a secure world software stack.

The overall structure of the software architecture will be heavily influenced by the nature of the available Secure world processing resource. A system may provide a TrustZone-enabled core, such as the ARM1176JZ(F)-S processor or may provide a dedicated processor for the Secure world, such as a Cortex-R4 processor.

The design that consists of the two separate physical processors is the simplest as each processor has a self-contained operating system with a minimal overall impact on the system design. Although, the most common and cost-efficient solution is incorporating the TrustZone technology within a single SoC together with the normal world dye.

There are many possible choices when designing the software architecture of the secure and the non-secure world. The most complicated one is having a full-fledged OS just for the secure world, on the other hand, the simplest solution is having just the libraries that provide the required functionality installed in the secure portion of the system, with many possible choices between these options.

Secure Operating System: Having a dedicated OS just for the secure world is the most secure option as it allows for having a flexible design within the secure world. This enables the developers to run concurrently multiple secure applications, dynamically installing new secure applications while having complete independence from the untrusted world. The most extreme option for this design resembles a design of separate CPUs for both worlds in an Asymmetric Multi-Processing (AMP) configuration. The next option is to have the virtual processors in a Symmetric Multi-Processing configuration that will allow for a closer relationship between the two worlds. For example, the secure world might inherit the priority of the normal world that will enable better response times for media applications. An example of a separate secure world OS can be seen in Figure 5.

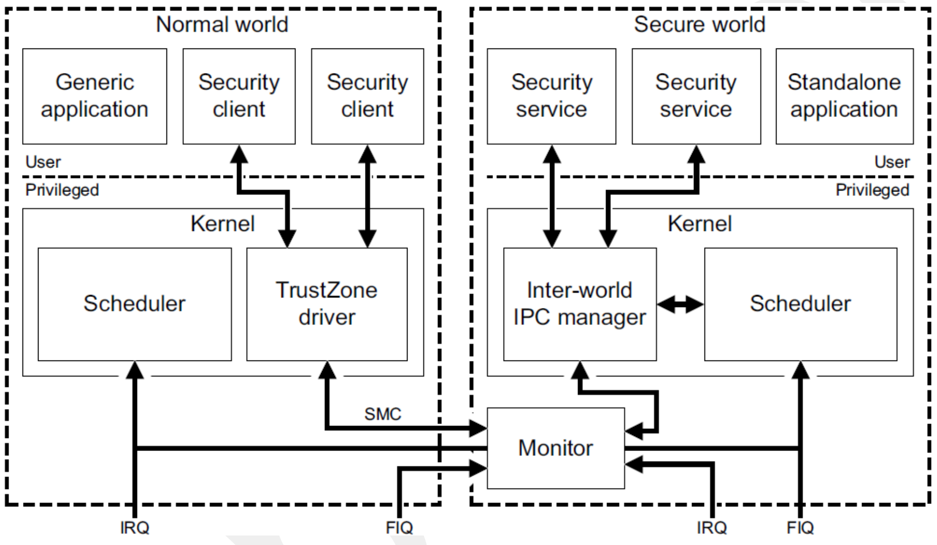

Figure 5: Software Architecture of the Normal World OS and the Secure World OS [13]

One of the advantages of having this design is that the MMU of each processor can be utilized to sandbox each secure application in the secure world by providing separate virtual address spaces for each one and effectively isolating them from each other. This way independent secure applications can be run concurrently in the secure world without the need of trusting their peer applications. The kernel design can enforce the logical isolation of secure tasks from each other, preventing one secure task from tampering with the memory space of another.

Synchronous Library: Many use cases do not need a separate operating system just for the secure world, especially in low powered devices that do not have the capacity of running two operating systems at the same time. The option of having just a library that utilizes the TrustZone technology installed is enough to handle one job at a time with a scheduling architecture that roots in the normal world. In this case, the secure world is the slave and the normal world the master, in a scheme that binds the secure world to the normal and deeming it incapable of running independently from it.

Intermediate Options: There are many possible options in-between these two extreme designs. For example, there might be a design where the secure world OS does not implement an interrupt system of its own and utilizes the system found within the normal world OS as a virtual interrupt. This design might be vulnerable to a denial of service attack if the normal world OS was compromised and was not able to provide this virtual interrupt service. Alternatively, the MMU could be used to statically separate different components of an otherwise synchronous Secure world library.

3.2.7 Booting Process of a Secure System

A core component and a common attack target for a secure system is its boot process. For example, while the device is powered off, an attacker might try to replace the secure world image with a tampered one that opens certain attack paths. If the system boots without first checking the authenticity of the binary blobs that run within it, then the system is vulnerable and easily exploitable.

Boot Sequence of a TrustZone Enabled System

The first concept that needs to be understood for the boot process of a TrustZone enabled system is what exactly happens during boot up. Any processor that has the TrustZone extension installed boots up directly into the trusted world, this allows for checking the authenticity of the normal world.

The very first boot process is the ROM SoC bootloader which is responsible for initializing crucial peripherals such as memory controllers before switching to the second level boot of the device which is contained within non-volatile memory such as a flash memory. Afterward, the boot process will fully initialize the secure world before booting and initializing the normal world. By then, the system is considered to be in a running state. Figure 6 depicts this process.

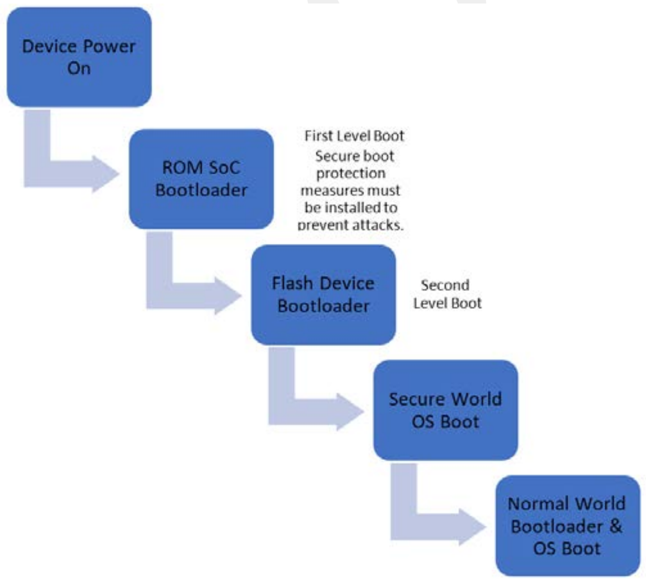
Device Power On
ROM SoC Bootloader
First Level Boot Secure boot protection measures must be installed to prevent attacks.
Flash Device Bootloader
Second Level Boot
Secure World OS Boot
Normal World Bootloader & OS Boot

Figure 6: Boot Sequence of a TrustZone Enabled System

Secure Boot

The secure boot process adds cryptographic verification steps on each stage of the boot process. The target of this process is to check the integrity of each binary that is loaded in the system before it has an opportunity to run. The cryptographic signature protocol used is based on public-key cryptography such as RSA-PSS where a trusted vendor uses their private key to generate a valid signature on the code he wants to publish and each device comes with the corresponding public key preinstalled with the ability to verify each received software for its integrity and authenticity. The public-key does not need to be kept confidential, but it does need to be stored within the device in a manner that means it cannot be replaced by a public-key that belongs to an attacker.

The secure boot process utilizes a concept called a chain of trust. Starting with an inherently trusted component (such as the hardware) every consequent component can be verified before it is allowed to execute. The root of trust can change on each stage, for example, initially the bootloader can be verified using the OEM public key as described above, but then the bootloader might contain a separate public key that will be used to verify the next stage boot component. Since the bootloader image is verified and trusted then the new public key contained within it is also trusted to be used for verification.

Storing the first stage public key is a puzzling problem. If the key is stored as-is in the device hardware, then the system suffers from a class-break attack that will deem all the devices vulnerable if the private key was to be stolen or reverse engineered. One-time programmable hardware on the SoC such us poly-silicon fuses can be used to store unique values in each SoC during device manufacture. This will allow for a different number of public keys to be stored on different devices that will partially mitigate this class break attack.

The simplest defence against hardware attacks is to keep sensitive code within on-SoC memory locations. If the code is never exposed outside the SoC then it is hard for an attacker to probe components within an integrated SoC design to mount hardware attacks due to the complexity of installing the needed physical probes within the SoC. The secure boot is responsible for loading code on the SoC memory, that is why care must be taken to properly verify any piece of code that is loaded and runs in the SoC so as to prevent any attack windows. Assuming the running code and required cryptographic hashes are already in safe on-SoC memory, the binary or public-key being verified should be copied to a secure location before being authenticated using cryptographic methods. A design that authenticates an image, and then copies it into the safe memory location risk attack. The attacker can modify the image in the short window between the check completing and the copy taking place.

Monitor Mode

As described above, the monitor mode is the management software stack that handles the context switching between the normal and the secure world while it also handles the communication between the two worlds. The monitor mode acts as a gatekeeper from the less privileged normal world to the high privileged secure world. Normal world entry to monitor mode is tightly controlled. It is only possible via the following exceptions: an interrupt, an external abort, or an explicit call via an SMC instruction. The secure world entry to the monitor mode is a little more flexible and can be achieved by directly writing to CPSR, in addition to the exception mechanisms available to the normal world. The monitor mode is a security-critical component as it provides the only interface between the two worlds.

3.3 Software Guard Extensions

Trusted Execution Environments with Intel SGX Intel’s Software Guard Extensions (SGX)[25, 26, 27] enable application code to be executed with confidentiality and integrity guarantees. SGX provides trusted execution environments known that isolate code and data using hardware mechanisms in the CPU. Intel SGX is the latest iteration for trustworthy computing that most of the future Intel processors will have this feature and use it as a TEE for addressing security problems. However, researchers raised security concerns about it. Recently, Costan and Devadas [28] analysed the security features of SGX and raised concerns such as *cache timing attacks* and software side-*channel attacks*. Additionally, [29] mentioned that SGX does not protect against software side-channel attacks including using performance counters. Moreover, SGX for desktop-like environments needs to establish a secure channel between I/O devices (e.g., keyboard and video display) and an enclave to prevent sensitive data leakage [30,31]. Fortunately, Intel Protected Audio Video Path (PVAP) technology can securely display video frames and play audio to users; Intel Identity Protection Technology (IPT) provides security features including Protected Transaction Display (e.g., entering a PIN by a user)[32].

Moreover, SGX needs Enhanced Privacy Identification (EPID) support for remote attestation [33]. The EPID is a security mechanism exclusively built-in ME and serves as the hardware root of trust [34]. During the manufacturing stage, a unique EPID private key is programmed in ME, and the system uses the EPID private key to provide to the localhost or a remote server that it is a genuine intel platform. The design of SGX assumes that the rmware could be malicious, it becomes unclear if the ME rmware is malicious since SGX relies on many hardware features (IPT, PVAP, and EPID) implemented by ME .

3.4 Other Hardware Implementations

On top of the aforementioned TEE solutions, more implementations have been brought to fruition.

3.4.1 AMD PSP [35]

The American multinational semiconductor company AMD has been integrating into its processors a Trusted Execution Environment subsystem since 2013, called AMD Platform Security Processor (PSP). It utilizes an Arm processor, as well as Arm’s TrustZone software solution, which is responsible for the execution of some security-sensitive operations and is differentiated from the main processor and operating system. In addition to the common functions, TEE implementations offer, like storing biometric information and/or encryption private keys, AMD PSP is also suitable for the execution of Digital Rights Management (DRM) solutions, making it more difficult for the users to disable or bypass them.

3.4.2 IBM Secure Service Container [36]

The IBM Secure Service Container is a software solution for IBM Cloud Private, that hosts container-based applications for hybrid and private cloud workloads on IBM LinuxONE and Z servers. Basically, IBM Secure Service Container is an environment where microservices-based applications can be deployed securely without the need for code changes to take advantage of the capabilities this environment offers. IBM Secure Service Container provides:

- Protection against tampering during installation
- Access with administrative rights is restricted in order to prevent malpractice
- Ad-hoc data encryption regardless of its state (in flight or at rest)

More specifically, it is a container used for installation purposes and running specific firmware or software appliances. An appliance is a unification of i) operating system, ii) middleware and iii) software components that work autonomously and provide fundamental services and infrastructures that focus on resource consumption and security. Firmware appliances are delivered with the mainframe system, compared to the software appliances which are delivered through software distribution channels.

3.4.3 MultiZone [37]

MultiZone is the first Trusted Execution Environment that uses that RISC-V standard (open-source hardware abstract computer model). It enables the development of a policy-based security environment suited for the RISC-V, fully supporting numerous operating systems. MultiZone is comprised of the:

- nanoKernel – lightweight kernel providing policy-driven, as well as the hardware-enforced separation of ram, rom, i/o and interrupts.
- Messenger – communications foundation for the secure message transmission across zones on a no-shared memory basis.
- Configurator – combines executables with policies and kernel to generate assigned firmware image, which seemingly deems tampering impossible.
- Signed Boot – 2-stage signed boot loader to verify the integrity and authenticity of the firmware image, utilizing sha-256 along with Error Correction Code.

MultiZone merges seamlessly with existing IDEs (e.g. Eclipse or other command line-based toolsets). Applications' programming code is developed, compiled and linked separately for each zone producing elf or hex files. MultiZone policies are set to achieve the desired ram, rom, i/o and interrupt isolation for each zone. Finally, the MultiZone configurator comes into play to incorporate the zone elf or hex files with the nanoKernel and bootloader into a signed firmware image. The entire system can be written, compiled and debugged with the existing GNUs or Eclipse toolsets.

3.4.4 OP-TEE [38]

OP-TEE is an open-source project, which contains a full implementation to construct a complete Trusted Execution Environment, making it possible to develop and integrate security services and applications. OP-TEE is created according to the GlobalPlatform TEE System Architecture specifications [39], while at the same time it provides the TEE Internal core API v1.1 as defined by the GlobalPlatform TEE Standard as an asset that can be used for the development of Trusted Applications. OP-TEE Trusted OS can be used from Linux based Operating Systems using the GlobalPlatform TEE Client API Specification v1.0, which also is utilized to trigger secure execution of applications within the TEE.

OP-TEE is under a BSD style license and can run trusted applications without being subjected to the restrictions posed by their licensing model.

The OP-TEE project is supported by the Linaro Security Working Group. In-depth information can be found at GitHub OP-TEE repositories [40] and OP-TEE official site [41].

The OP-TEE project is comprised of several secure and non-secure embedded components, as well as a number of tools designed for development and debugging purposes.

In Figure 7 we can observe the main OP-TEE embedded components, i.e. the OP-TEE core and trusted application standard libraries, and the Client API library, the OP-TEE supplicant daemon and the OP-TEE Linux kernel driver on the secure and non-secure sides respectively.

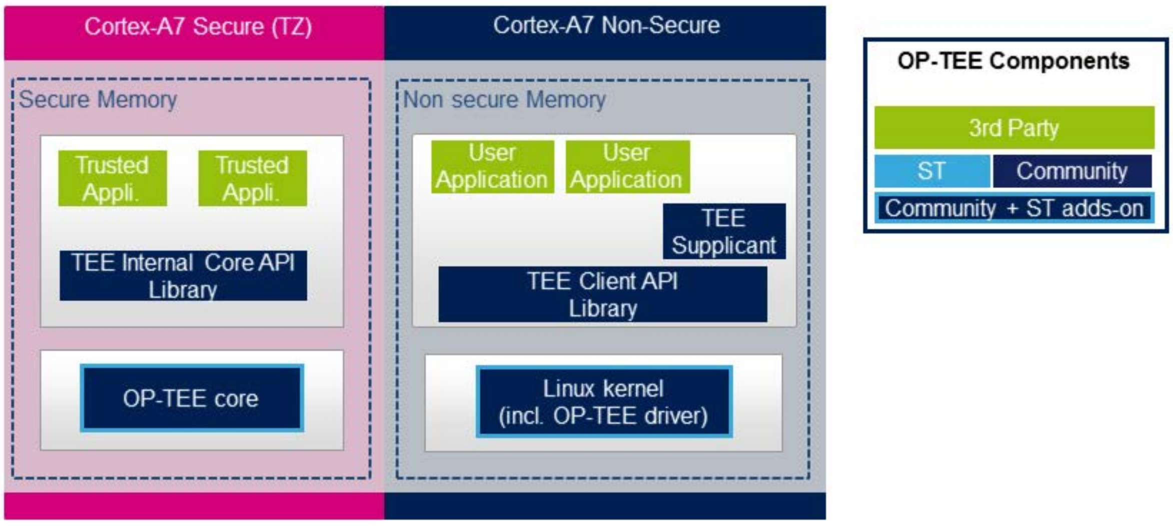

Figure 7: OP-TEE Architecture [38]

3.5 Software Implementations

3.5.1 OpenTEE

Open-TEE is an open-source implementation of the Trusted Execution Environment. Hardware-based implementations of TEE do not allow access. Debugging low-level TEE applications either requires expensive hardware debugging tools or leaves the developer with only primitive debugging techniques like “print tracing” (e.g. using printf statements in C to keep track of how values of variables change during program execution). Open-TEE was designed as a hardware-independent tool for developers and researchers that want to build trusted protocols, systems or applications on top of it. Open-TEE allows the compiling and running of any application that complies with the Global Platform specifications. Once a trusted application is fully debugged, it can be compiled for any actual hardware TEE.

The goals of Open-TEE can be summarized as follows [46]:

1. Enable developer access to TEE functionality
2. Provide a fast and efficient prototyping environment
3. Promote research into TEE Services
4. Promote community involvement

3.6 TEE Vulnerabilities

TrustZone Code Execution [15]

This is the first of a set of three attacks against a TrustZone based TEE implementation in MSM8974 SoCs found in commodity mobile devices. The first attack achieves code execution in the context of a trusted application in the TEE. In this attack, the author, first of all, exploited the MediaServer process to gain arbitrary code execution within its context. The MediaServer process is one of the entities that are able to communicate with the underlying TEE and that is why it is a high-value initial target.

With MediaServer privileges, the author then proceeded to probe the QSEECom driver. This driver is responsible for managing the TEE device while it also runs in the normal-world kernel context. It was found that this driver contained a bug that allowed code escalation to kernel privileges of the normal world. This way the author escalated to a higher-privileged position that allowed him to directly speak to the TEE device without going through the QSEECom driver.

With this position, the author probed the secure world with direct SMCs that allowed him to discover a set of vulnerabilities in the corresponding MediaServer trusted application that ultimately let him run arbitrary code within the context of this application. This way the author managed to escalate from normal user privileges to trusted application privileges that let him access any assets that this application had access to.

TrustZone Kernel Exploitation [16]

Continuing on the previous attack, the author was not able to access any information in the secure world due to the fact that each trusted application is isolated from each other and from the secure world kernel. This time though he went through a different path, instead of gaining normal-world kernel privileges he directly found a way of exploiting the trusted application by using the QSEECom driver directly.

Once again, the author had trusted application code execution privileges and targeted the kernel of the secure world. By bypassing a set of lockdown security features and hijacking the system call architecture of the secure world, he was able to write shellcode binaries that were run in the context of the TrustZone kernel. This way he was finally able to run arbitrary secure-world kernel-privileged code, the highest level of privileges in the system as it can access any resource from any other entity in the system.

Extraction of Master keys from TrustZone [17]

With the highest privileges, any resource could be accessed except for hardware bound keys that are concealed through the hardware. Although this master key is unattainable through the software, all the other keys that are derived from this key are stored in memory accessible by the software. The Android full-disk encryption (FDE) is one of those keys that the author of the blog targeted to demonstrate the power of the privileges he obtained. The FDE key is responsible for encrypting and decrypting the android memory so that only legitimate users can access it. It is a crucial part of the Android security system as with it anybody can read anything from the device memory.

More specifically, the author reverse-engineered the keymaster trusted application which is responsible for managing all the cryptographic keys used in the Android ecosystem and discovered that the FDE key is not directly protected by any hardware-bound keys but by a software key which resided in the global buffer of the keymaster trusted application. By using a chain of exploits available to the author through the trusted world kernel privileges he was able to gain access to the FDE key, thus nullifying the disk encryption system of the Android ecosystem.

BOOMERANG Vulnerability [11]

BOOMERANG is a class of vulnerabilities that arises due to this semantic separation between the TEE and the untrusted environment. These vulnerabilities permit untrusted user-level applications to read and write any memory location in the untrusted environment, including security-sensitive kernel memory, by leveraging the TEE's privileged position to perform the operations on its behalf. BOOMERANG can be used to steal sensitive data from other applications, bypass security checks, or even gain full control of the untrusted OS.

This exploitation is possible due to the fact that the two worlds have not well-defined means of communication in an ecosystem with ubiquitous implementations of trusted execution environments in many consumer devices. The problematic behaviour roots in the different memory access control systems that are installed in the two worlds that are not well equipped to inherit the restrictions set by them. The target of this attack is to send restricted normal world addresses to the high-privileged secure world and convince it to write or read from these addresses since it has access to them. So, the attack targets the normal world and begins from the normal world, but it utilizes the secure world as a middleman to execute the actual attack.

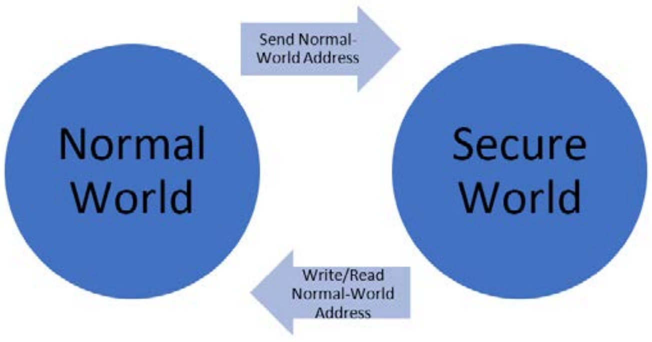

Figure 8: The Boomerang Attack

More specifically, the pointer sanitization that controls what addresses are sent to be used by the secure world, only checks in specific regions of the shared memory between the two worlds. The sanitization process checks if the addresses checked are within the allowed range that the user has access to, if these addresses are out of this range then the execution is halted. The boomerang attack requires that the attacker hides these addresses in a part of the shared memory where the memory sanitizer does not check in order to bypass this security measure. When the secure world receives this address, it has no way of checking the province of this address, it assumes that it has been checked and it will act blindly upon it. With this behaviour in hand, several trusted applications were identified that could be used as primitives for writing arbitrary binary values in any specific memory address of the normal world.

Downgrade Attack [18]

The downgrade attack is a form of attack that can be performed on the current implementations of the widely deployed ARM TrustZone technology. The attack exploits the fact that the trustlet (TA) or TrustZone OS loading verification procedure may use the same verification key and may lack proper rollback prevention across versions. If an exploit works on an out-of-date version, but the vulnerability is patched in the latest version, an attacker can still use the same exploit to compromise the latest system by downgrading the software to an older and exploitable version. Experiments were made on popular devices on the market including those from Google, Samsung, and Huawei, and found that all of them have the risk of being attacked.

Research has shown that most TEE implementations include systems for version controlling the binaries installed within the system for rollback attack prevention, it was found that all the vendors checked do not use this feature. So almost all devices in the market are vulnerable to the downgrade attack. This attack is very easy to mount as the string that specifies the trusted application binary location path can be easily manipulated and without many privileges, an older version of the trusted application could be loaded from the SD card of the system.

Cache Timing Attacks

There is a set of attacks that exploit the side channel of timing the cache usage of the secure world.

Cache Timing Attack on AES in Virtualization Environments [19]**:** This attack does not target any specific TrustZone implementation but aims to show that the isolation characteristic of system virtualization can be bypassed by the use of a cache timing attack. Using Bernstein's correlation in this attack, an adversary is able to extract sensitive keying material from an isolated trusted execution domain. The authors of the research have demonstrated the attack on an embedded ARM-based platform running an L4 microkernel as a virtualization layer by extracting an AES key that was used in a virtualized environment. This attack is mounted against an isolated virtualized environment running on the same processor as the "normal world" environment, proving that cross-world side-channel cache-timing attacks are possible, something that can be applied in TrustZone TEEs.

Prime+Count Attack [20]**:** The researchers have demonstrated a side-channel that does not use the classic fine-grained methods (prime+probe, flush+reload, etc.) but instead uses the prime+count method that is a coarser-grained approach that significantly reduces the noise introduced by the pseudo-random replacement policy and world switching. This is not an attack per se but more of a demonstration for the methods that could be used to "smuggle" data through unmonitored side channels of the system.

ARMageddon Attack [21]**:** The ARMageddon attack provides a novel method for performing cross-core cache timing attacks against ARM-based CPUs that were not possible before. More specifically, the researchers have shown how to solve key challenges to perform the most powerful cross-core cache attacks Prime+Probe, Flush+Reload, Evict+Reload, and Flush+Flush on non-rooted ARM-based devices without any privileges. According to the research, this attack outperforms most cache-timing attacks at the time of the writing while it provides proof of concept demonstrators that sniff gestures of the user in secure inputs and steal cryptographic keys from a Java AES implementation. Although the attacks are not made against a TrustZone TEE, the research shows evidence that it can be escalated to affect even the secure world of the system.

TruSpy Attack [22]**:** This attack exploits the cache contention between the normal world and the secure world to recover secret information from the secure world. Two attacks are proposed in TruSpy, namely, the normal world OS attack and the normal world Android app attack. In the OS-based attack, the attacker is able to access virtual-to-physical address translation and high precision timers. In the Android app-based attack, these tools are unavailable to the attacker, so the researchers devise a novel method that uses the expected channel statistics to allocate memory for cache probing. They also show how an attacker might use the less accurate performance event interface as a timer.

CLKscrew [23]

The need for power- and energy-efficient computing has resulted in aggressive cooperative hardware-software energy management mechanisms on modern commodity devices. Most systems today, for example, allow the software to control the frequency and voltage of the underlying hardware at a very fine granularity to extend battery life. Despite their benefits, these software-exposed energy management mechanisms pose grave security implications that have not been studied before.

The CLKscrew attack exploits the software power management system to set the operational frequency and voltage in a combinational setting that will induce faults during sensitive operations of the system. Due to the software nature of this method, this is one of the few hardware attacks that can be mounted remotely.

More specifically, the attacker investigates possible combinations of operational frequency and voltage to find the exact settings needed so that the CPU begins to introduce faults in its computations. The attack then can begin by first clearing the cache of the victim core to reduce the random noise that could be introduced by it (step 1). In the next step, the attacker profiles the execution of the targeted code to find the exact time that the code to be faulted is executed (step 2). Based on this profiling, the attacking process sets a timing anchor (step 3) after which a specific number of no-op operations are performed until the targeted piece of code is executed. Then the attack takes place by setting the fault-inducing settings on the processor for the time that the targeted operation executes and afterwards it is set to the normal operational settings. (steps 5,6).

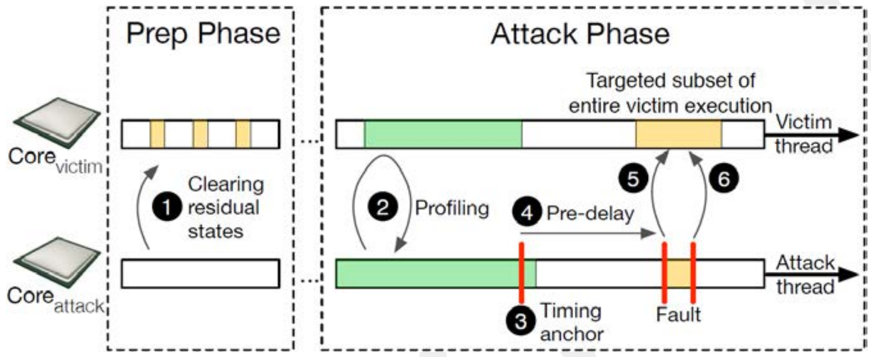

Figure 9: The CLKscrew Attack [23]

BADFET [24]

The researchers that demonstrated the BADFET attack proposed a novel solution that reinvented the electromagnetic fault injection attacks. The main contribution is that instead of targeting the processor during its operation, they targeted the peripherals that store sensitive data. So, the CPU does not directly introduce the faults during its operation, but intermediate data stored outside of the CPU are faulted so that the CPU operates with faulty data.

To perform this second-order electromagnetic attack the researchers utilized a computer-controlled targeting system that was equipped with a high-power precision laser that will induce the faults in the peripherals. In the research, an attack was demonstrated against the TrustZone equipped Cisco 8861 IP phone. The researchers first used their attack setup to interrupt the boot process which in turn forced the device to give a u-boot console. It is through the u-boot console that the attackers managed to find a way to access the TEE of the device and gain a terminal with the privileges of the trusted world.

3.6.1 Attack Comparison

The attacks that have been presented this far achieve TrustZone based TEE exploitation by either gaining execution within the secure world context or uncovering secrets hidden in the secure world. Depending on how and what the attack achieves, each attack is analysed in the following table (Table 1).

The characteristics of each attack include the target of each attack which can be either the secure world, the normal world or both of them. A strong attack targets the secure world or both of them, without diminishing the potential of just normal world attacks which can provide the attacker with normal world kernel privileges. Furthermore, there is also the characteristic of whether the attack requires normal world root privileges, something that defines the applicability of each attack due to the fact that these privileges might not be available on all devices.

Depending on what the attack can achieve, there are also the characteristics of whether an attack uncovers secrets from the secure world, gains code execution in the secure world and gains kernel code execution in the secure world. Once again, the more the attack achieves the stronger it is as it has a larger field of effect in the system.

We have marked with X the attacks that achieve the attack target (seen in the first row of the table) while with a question mark (?) the attacks that have the potential of achieving the corresponding target.

Table 1: List of Characteristics for Each Attack.

	Targets Secure World	Targets Normal World	Requires Root Privileges	Uncovers TrustZone Secret	Gains TrustZone Execution	Gains TrustZone Kernel Execution
TrustZone Code Execution	X		X		X	
TrustZone Kernel Exploitation	X				X	X
Extraction of Master keys from TrustZone	X			X	X	X
BOOMERANG Vulnerability		X	X			
Downgrade Attack	X			?	?	?
Cache Timing Attack on AES in Virtualization Environments	X		X	X		
Prime+Count Attack	X	X	?	X		
ARMageddon Attack	X	X	?	X		
TruSpy Attack	X	?	?	X		

CLKscrew	X		X	?	?	?
BADFET	X			X	X	

The main problems identified in the TrustZone implementations can be categorized in the following categories:

- Closed Source Design
- Coding Bugs
- Shared Architecture
- Unused Features
- Unstandardized Communications

The closed source of the code that implements the TEE of commodity devices cripples the ability of the community to identify and fix possible vulnerabilities. Choosing to open source the core components of each TEE implementation will benefit the vendors by letting other experts check their code and improve their security. After all, the security of the system must be consolidated by openly reviewed protocols and not the obscurity of closed source proprietary solutions.

Most of the implementations that were presented by the researchers in this deliverable suffered from serious coding bugs. In many occasions, these bugs lead to vulnerabilities that ultimately let the researchers to completely exploit the TEE installed on the device. This situation could be improved if proper coding techniques were utilized such as static code analysis and dynamic probing of the running software. Vendors should make a greater effort to develop secure and bug-free code because as the results show, their implementations have problems that could have been easily avoided.

The shared architecture of the TEE sacrifices the complete hardware isolation in order to reduce the manufacturing cost. More specifically, depending on the implementation, the secure and the non-secure world both share CPU cores, cache memory, and non-volatile memory. Although the TrustZone technology provides techniques to properly isolate one world from each other, it is possible for attackers to find side channels that will ultimately allow them to gain sensitive information from the secure world. This problem can be solved by adding separate hardware for the secure world, but this will increase the manufacturing cost or properly check their software code in order to avoid side channels, but this path might not lead to a guaranteed solution to the problem.

Furthermore, the developers of most TEE implementations do not use the basic feature of version controlling the trusted application binaries in the secure world. This has led to the major vulnerability of the downgrade attack that allowed an adversary to sideload an older and possibly vulnerable trusted application binary and exploit this binary to gain access to the secure world. Missing this feature is a major issue because the functionality was already provided and not using it is wasteful.

Finally, the lack of standardization for the communication between the two worlds has led to vulnerabilities that were unforeseen. More specifically since the secure world has very high privileges, it can act upon any memory that belongs to the secure world even on addresses that contain kernel data and code. That is why in the boomerang vulnerability the researchers were able to use this miscommunication to push the secure world into exploiting the normal world kernel.

Closing, the trusted execution environment is marketed as a security solution that can solve many trust problems that rooted in the fact that security-sensitive operations were run in the same environment as the normal operating system. Although sometimes it is claimed that it is one of the highest security solutions, the TEE can have many security problems that can totally break the security guarantees it should have provided. That is why care must be taken to fix these issues or the TEE technology could be deprecated and replaced by other dedicated solutions such as Trusted Platform Modules (TPMs).

3.7 TEE-Assisted Technologies / Research

3.7.1 TEE-Assisted Identity Management & Authentication

Using TEE technology for identity management and authentication poses an interesting research topic, since using TEEs could greatly improve the security of such functions. The work of Spantzel et al. [43] describes how a trusted execution environment (TEE) can be utilized effectively to provide trustworthy client-side and server-side biometric verification. More specifically, this work can be beneficial to the service providers, since the TEE can be attested to provide high assurance about the correctness of the device biometric verification. Furthermore, this solution provides high confidence regarding the privacy and user control of the sensitive biometric data of the user. In addition, in [43] the researchers have shown a way to use TEEs to protect the user's login credentials.

Furthermore, in [42], the researchers propose the usage of TEEs for device authentication in a smart grid system. The solution assumes a mutual authentication scheme, where each device will have a private cryptographic key that must be protected. Current solutions for protecting such keys exist but generally require user interaction which makes them unusable in a smart grid system due to their automated nature. To address this challenge, the researchers have designed, implemented and tested a system that provides high security for the device keys using TEE based technologies. By using DRTM (Dynamic Root of Trust Management) late-launch functionality, the proposed solution ensures that the key is only available within a protected trusted environment.

Moreover, in [44], there is an examination of TEE solutions (and Secure Elements) for the protection of continuous authentication schemes. The researchers have implemented an SGX-based continuous authentication system as a proof of concept which is able to provide confidentiality, integrity, performance and trust assurances over another untrusted implementation.

The goal of [45] is to present the problem of secure user enrolment in the context of mobile system-wide TEEs. The researchers describe the challenges behind user identity binding to a mobile device on current smartphone platforms and argue that current mobile device architectures do not facilitate secure enrolment and migration for such TEEs. For this reason, they propose possible architecture changes that would enable the implementation of secure and practical enrolment and consequently enable the adoption of the secure deployment of various mobile security services.

3.7.2 TEE Assisted FIDO

The FIDO (Fast IDentity Online) UAF protocol for passwordless authentication proposes the use of a Truslet (trusted application running inside TEE) which may be designed to implement the UAF Authenticator functionality. [47] More specifically, the TEE can be used for the implementation of the user verification technology (which can be performed with methods such as biometrics or PIN) as well as the key protection.

In FIDO UAF, access to credentials for performing the actual authentication can be protected by a user verification step. This user verification step can be based on a PIN, a biometric or another method. The authenticator functionality might be implemented in different components, including combinations such as TEE and SE, or fingerprint sensor and SE. In that case, the SE implements only a part of the authenticator functionality. Examples of hybrid SE authenticators are [48]:

1. User PIN code capture and verification are implemented entirely in a TEE relying on Trusted User Interface and secure storage capabilities of the TEE and, once the PIN code is verified, the FIDO UAF crypto operations are performed in the SE.
2. User fingerprint is captured via a fingerprint sensor, the fingerprint match is performed in the TEE, relying on matching algorithms. Once the fingerprint has been positively checked, the cryptographic operations are executed in the Secure Element.

Moreover, the authentication keys in FIDO UAF are protected by TEE and only used to sign valid FIDO sign assertions. The (fingerprint) matcher is also implemented in TEE. If the authenticator supports Transaction Confirmation, then the display will also be implemented in TEE through Trusted User Interface (TUI).

3.7.3 TEE Assisted Secure Boot

During the initialization of a computer system, the flash contents are accessed so the system will be able to boot. The problem is that Integrity and confidentiality of flash content are not assured, and it is possible to load an infected version of the OS. This problem persists because TEE security is not established; Secure Boot can provide the assurance that the content used during the boot procedure has not been tampered with. The only requirement is that the TEE needs to be securely initialized before running any REE code. [50] The TEE plays a significant role in the secure boot process, as the TEE boots after the primary ROM boot but before the REE. The TEE can boot the REE as part of the boot sequence; by doing so, the REE image can be verified and remedial action may take place if needed. [49]

3.7.4 TEE Assisted Digital Forensics

Digital forensics is one of the most intricate steps of the cybercrime investigation process. It is the scientific acquisition, analysis, and preservation of data contained in electronic media whose information can be used as evidence in a court of law. The practice of digital forensics can be a career unto itself, and often is. The corporate digital forensic practitioner is not a law enforcement officer, it is a wise practice to follow the same procedures as law enforcement does when performing digital forensics.

To be presentable in court, digital forensic evidence needs to be admissible, authentic, complete, reliable and believable [51]. According to [52], the goals of digital forensics are thus preservation, identification, extraction, documentation, and interpretation of data while ensuring confidentiality, integrity, and authenticity. In order to maintain the forensic viability of log files, they have to be created and stored keeping legal investigation procedures in mind [53].

According to the importance of the Digital forensics, we can break down the digital forensic process into four main phases that some of the work performed may overlap into multiple phases, but the phases themselves are different:

- **Collection**: This phase is the preservation of evidence for analysis. Current best practices state that digital evidence needs to be an exact copy—normally a bitstream copy or bit-for-bit duplication—of the original media. The bitstream copy is then run through a cryptographic hashing algorithm to ensure that it is unaltered. In modern digital forensics, often this is done by physically removing the hard drive from the device, connecting it to a write-blocking unit, and using forensic software that makes forensic duplicates of the data.
- **Examination**: This phase is the methodical combing of the data to find evidence. This includes extracting documents and emails, searching for suspicious binaries, and data carving.
- **Analysis**: Analysis is the process of using the evidence you recovered to help solve the crime. The analysis pulls together all the bits and pieces and deciphers them into a story of what happened.
- **Reporting**: Reporting is the phase where all the other phases are documented and explained. This phase should contain documentation of the hardware, the tools used, the techniques used, and the findings.

In software forensics that can be in the source code format or object code form, authorship analysis techniques are used to determine who wrote specific software. This is very important for discovering the authors or programmers of viruses and worms, logic bombs, Trojan horses, computer fraud, or code plagiarism. It should be noted that several software forensic works have been proposed based on authorship analysis techniques. The effect of trusted computing on software forensics is still not totally clear yet. However, it seems that the authorship identification techniques will be used in trusted computing software forensics and there is a need to study the benefits of remote attestation, digital signature, and sealed storage in software forensics.

4 Trusted Platform Module

4.1 Background

4.1.1 Trusted computing standards

The Trusted Platform Module (TPM) is a low-cost security module that delivers the basis of a safe computing environment. It is typically implemented as a tamper-resistant integrated circuit (IC). It is designed to be a building block for trusted computing. Unlike smart cards, the TPM is bound to a specific platform. It is essential to understand the fundamental design objectives of Trusted Computing (TC). In order to completely understand the development of the TPM. Trusted computing allows users to assess the trustworthiness of computers with which they interact and create a foundation of trust for software processes [54]. The Trusted Computing Group (TCG) has attempted to describe what trusted computing means and in the process, it produced a series of standards to be used in order to design trusted platforms. The TSG defines trusted computing as the expectation that “a device will behave in a particular manner for a specific purpose" [55]. Their definition really claims the fact that whoever interacts with a trusted device can be given assurance that the device will behave exactly as they expect it to. The purpose of the Trusted Platform Module (TPM) is to provide this assurance to the client and the users interacting with the client [54]. The TPM provides a fundamental set of security features that have been defined by the Trusted Computing Group TCG. The TCG specifications enable more secure computing environment and the Trusted Platform Module (TPM) is the building block to achieve its goals. The adoption of Trusted Computing using the TPM is a significant step toward refining confidence in business over the Internet and allows existing applications to benefit from enhanced security.

A Trusted Platform Module (TPM) is an implementation of a defined set of capabilities that are intended to provide authentication and attestation functionality for a computing device and protect information by controlling access to plain-text data. A TPM is self-sufficient as a source of authentication and as a means of enhancing the protection of information from certain physical attacks. A TPM requires the cooperation of a TCG "Trusted Building Block" (outside the TPM, which is also part of the computing device) in order to provide attestation and protect information from software attacks on the computing device. Typical TPM implementations are affixed to the motherboard of a computing device. A computing device that contains both a TPM and a Trusted Building Block is called a Trusted Platform. Trusted Platforms offer improved, hardware-based security in numerous applications, such as file and folder encryption, local password management, S-MIME email, VPN and PKI authentication and wireless authentication for 802.1x and LEAP

A TPM usually is implemented as a chip integrated into the hardware of a platform (such as a PC, a laptop, a PDA, a mobile phone). A TPM owns shielded locations (i.e. no other instance but the TPM itself can access the storage inside the TPM) and protected functionality (the functions computed inside the TPM cannot be tampered with). The TPM can be accessed directly via TPM commands or via higher layer application interfaces (the TCG Software Stack, TSS). The TPM offers two main basic mechanisms: It can be used to prove the configuration of the platform it is integrated into and applications that are running on the platform, and it can protect data on the platform (such as cryptographic keys). These mechanisms can be performed by means of the crypto co-processor, hash and HMAC algorithm, key generator, etc. provided by TPM.

In order to prove a certain platform configuration, all the parts that are engaged in the boot process of the platform (BIOS, master boot record, etc.) are measured (i.e. some integrity measurement hash value is computed), and the final result of the accumulated hash values is stored inside the TPM in a so-called Platform Configuration Register (PCR). An entity that wants to verify that the platform is in a certain configuration requires the TPM to sign the content of the PCR using a so-called Attestation Identity Key (AIK), a key particularly generated for this purpose. The verifier checks the signature and compares the PCR values to some reference values. Equality of the values proves that the platform is in the desired state. Finally, to verify the trustworthiness of an AIK's signature, the AIK has to be accompanied by a certificate issued by a trusted Certification Authority, a so-called Privacy CA (PCA). Note that an AIK does not prove the identity of the TPM owner.

Keys generated and used by the TPM have different properties: Some (so-called non-migratable keys) cannot be used outside the TPM that generated them; some (like AIKs) can only be used for specific functions. We highlight the fact that keys can be tied to PCR values (by specifying PCR number and value in the key's public data). This has the effect that such a key will only be used by the TPM if the platform (or some application) configuration is in a certain state (i.e. if the PCRs the key is tied to contains a specific value). In order to prove the properties of a particular key, for example, that a certain key is tied to specific PCR values, the TPM can be used to generate a certificate for this key by signing the key properties using an AIK. For requesting a TPM to use a key (i.e. for decryption), the key's authorization value has to be presented to the TPM. This together with the fact that the TPM specification requires a TPM to prevent dictionary attacks provides the property that only those entities that know the key's authorization value can use that key. Non-migratable keys are especially useful for preventing unauthorized access to some data present in a platform. Binding such a key to specific PCR values and using it to encrypt data to be protected achieves two properties: The data cannot be decrypted on any other platform (because the key is non-migratable), and the data can only be decrypted when the specified PCR contains the specified value (i.e. when the platform is in a specific secure configuration and is not manipulated).

The current version of the TPM specification is 2.0. Specification for the TPM library version 2.0 has been released for public review by the TCG. The TPM main specification is an industry specification that enables trust in computing platforms in general. The main specification is broken into parts to make the role of each document clear. A version of the specification requires all parts to be a complete specification. A TPM designer MUST be aware that for a complete definition of all requirements necessary to build a TPM, the designer MUST use the appropriate platform-specific specification for all TPM requirements.

TPM 2.0 is designed to be a self-contained computing device. This allows the TPM device to be trusted to carry out computations without relying on external computing resources. The following are short descriptions of the subsystems in a TPM 2.0 device while a detailed explanation can be obtained from TPM 2.0 specification [56].

- I/O Buffer – This component enables the host computing system to communicate with the TPM. It can be a shared memory. Data to be processed by the TPM will be validated at this point.
- Cryptography Subsystem – The cryptographic engine supports commonly used cryptographic functions like hashing, asymmetric operations such as digital signature and key exchange, symmetric encryption, random number generator, and key derivation function. These cryptographic functions can be used by the other TPM components or the host computer.
- Authorization Subsystem – Before a TPM command is executed, this subsystem checks that proper authorization data has been given by the calling application.
- Volatile Memory – This memory holds transient TPM data, including Platform Configuration Registers (PCR), data objects and session data. PCR contains the integrity measurements of critical components in the host computer. A data object can either be a cryptographic key or other data. The TPM uses sessions to manage the execution of a series of commands.
- Non-Volatile (NV) Memory – This memory is used to store persistent TPM data that includes the platform seed, endorsement seed, storage seed, and monotonic counter. Additional PCR banks can be created in this memory.
- Management Subsystem – This subsystem oversees the operation of the various TPM states. Basic TPM states include power-off, initialization, start-up, shut down, self-test, failure and field upgrade.
- Execution Engine – This firmware contains the program instructions and data structures that are required to run a TPM command. These program instructions and data structures cannot be altered by the host computing platform. In the event of a firmware upgrade, there are security mechanisms to ensure that the update is authorized, and the new firmware is checked for authenticity and integrity.

4.1.2 Trusted computing certification mechanisms

Trusted Computing (TC) is an important technology when we need to process highly sensitive data and its use is significantly increasing. In essence, TC allows verifying that the hardware and software used to execute important processes are legitimate, which if we use a certified platform assures that the execution will not contain security flaws. For this purpose, TC uses a complex and interesting architecture embedded in a chip as a security pattern. The TC provides four main security functions, which are secure I/O, memory curtaining, sealed storage, and remote attestation. With the secure I/O, the system input/output data are encrypted and only captured by the related running application. Using the memory curtaining, the data stored in the system memory are encrypted and only the related application can read them. The sealed storage function means the data saved on the system's hard drive are encrypted and readable only to the related application in the original drive. The remote attestation function is used by the trusted platform to prove its security properties to a remote party. These functions of the TPM are executed by accessing what is called the "TCG software stack (TSS)," an interface that provides standard application programming interfaces (APIs) and that can be used or accessed by the platform's applications.

One of the most relevant features of Trusted Computing is the concept of the chain of trust. This chain starts using TC to verify the BIOS, then the operating system (OS) can be verified by the BIOS, and applications can be verified by the OS. Also, Trusted Computing provides sealed data protection against stealing and provides a remote attestation mechanism for online transactions. These abilities allow, for example, for an entity to verify that a secure OS is running, a browser is protected against man-in-the-middle attacks and key loggers or discard online cheating in platforms (Video On Demand, games) by means of verifying the combination OS / Application. Typically, Trusted Computing provides secure storage of keys and a key generation engine using as seed a unique Endorsement Key (EK), issued for each chip at manufacturing time.

In today's organizations, IT-infrastructures are formed by different systems and devices working in a non-trusted environment, and demand of trusted elements as part of the infrastructure is increasing [57]. The root of trust of Trusted Computing is the basis to address this issue and provides means to build secure environments for systems and devices. Attestation is a mechanism for software to prove its identity. The goal of attestation is to prove to a remote party that an operating system and application software are intact and trustworthy.

According to the trust definition included in RFC 4949, trust implies "a feeling of certainty" that either a system will not fail or that the system meets its specification (by actually doing what it claims to do or by not performing unwanted functions). In all systems dealing with strict security standards such as PCI-DSS it is also important to store securely secret keys.

TC's architecture is complex and a piece of deep knowledge in this technology is needed to exploit its functionalities, which hinders even system designers with some security expertise to use TC to take advantage of its functionalities:

- Attestation. We need a way to attest that a platform is legitimate and that any software executing in it is also legitimate.
- Secure cryptography. We need ways to generate keys in a secure way and use them to encrypt/decrypt documents and data.
- Secure signature generation. We need to be able to generate signatures using the most advanced algorithms.
- Secure hashing. We need to be able to apply secure hashing functions to verify that data has not been changed.
- Secure processing. We need a way to assure that during execution the software does not leak or modify information.
- Dynamic Root of Trust (DRT). This defines the first measurement made by the hardware and if correct it is possible to proceed to further measurements.
- Authenticated Boot. We need a mechanism that records the boot process to establish the transitive chain of Trust.

- Sealed-bound key. We need a mechanism that guarantees that keys used for encryption/decryption actually reside in a trusted platform. We need to bind the generation of these keys to the platform state.

4.1.3 TPM Functionalities

The TPM provides an entire suite of tools used for secure authentication. Existing security systems have an inherent flaw, since they run on top of unknown hardware. Therefore, an insecure system, even when running in a secure network, may compromise all the underlying infrastructure. Key encryption is a prime example. For instance, anyone with the private key has full authenticated access to a network. If one system gets compromised, existing keys can easily be extracted and later used for an attack. In order to overcome that, systems with a TPM store sensitive data in a secure location on a separate piece of hardware. The TPM chip uniquely identifies the hardware and does not allow sensitive data like keys to leave the TPM. In this regard, sensitive data can be used from within the TPM but cannot be directly accessed outside of it. Therefore, even if someone managed to get unrestricted access to the computer, they would not have access to the sensitive data stored on the TPM. The basic premise for the TPM is the same idea behind mobile phones. The TPM is a chip that authenticates the hardware itself. The chip is designed such that only a small subset of safe actions are allowed (ex. Encrypting a message with a key is allowed, however, access to the key directly is restricted). Since most attacks originate from unknown hardware, being able to identify the device eliminates the possibility of someone stealing a key and reusing it later on. Neverhteless, this is not the only benefit of TPM's, since they provide an entire cryptographic suite of tools including secure storage as well as random number generators, several key engines, and even registers for processing from within the TPM. The basic components-functionalities of a TPM are the following:

Attestation Identity Keys (AIK)

AIKs are generated from Endorsement Keys (EK) which is stored in non-volatile memory. However, AIKs may be stored outside the TPM securely. In fact, TCG recommends it and states that the TPM must provide enough room in volatile memory where one or more AIKs can be loaded to speed up execution [58], [59]. Furthermore, each TPM can generate multiple AIKs. This feature allows a user to be anonymous [58].

Non-Volatile Storage

This non-volatile memory stores several long-term (embedded) keys and authentication credentials such as Endorsement Key (EK), Storage Root Key (SRK), owner's password and persistent flags [59]. The SRK naturally is the root of this secure storage and thus manages it. The EK, however, is a unique feature in TPM and should deserve more details. In order for TPM to operate, the EK pair must be embedded in it; the private key is permanently embedded in it (i.e. unique to each TPM and thus the platform). The public key, however, is stored in a certificate and it is only used in a limited number of operations. EK is used to generate an alias, Attestation Identity Keys or AIKs, which are used for routine operation. EK is also used for encrypting data sent to the TPM during the ownership process. The EK pair is generally provided by the manufacturers before shipping [58].

A certificate can then be created from the pair which contains the public key and security properties of the TPM. This certificate is generally signed by Trusted Platform Module Entity (TPME) a certification authority, who can verify the certificate contains a public key whose corresponding private key is stored in the TPM with the specified security properties [58]. Essentially to verify that the EK pair is stored in a genuine TPM. Users who wish to customize their own security can delete and reinstall the EK. However, they need to make sure that their certificate is signed by the proper TPM.

Platform Configuration Registers (PCR)

PCRs are a set of registers that store integrity metrics, which is a unique feature for TPM. The metrics measure the integrity of any code before launching, ranging from BIOS to application codes [58]. Like AIKs, PCRs can be implemented in either volatile or non-volatile storage. One strict requirement is that these registers must be reset at system restart or whenever there is a power loss. The reason behind this is to ensure PCRs contain the most recent integrity metric should a system reconfigured and rebooted [58]. The standards require TPMs to have at least 16 PCRs of size 20 bytes. Registers 0-7 are reserved for TPM use. Registers 8-15 are free for any applications to use, including operating system [58], [59].

Program Code

The program code contains the firmware that is used to initialize the device. Because of its purpose, it is permanently stored on the tamper-proof TPM and assume to be trustworthy, in the sense that if it fails, all security policy will be broken. Then, the program code is the obvious “root of trust" for integrity measurements, which is also known as Core Root of Trust for Measurement (CRTM) [58].

Execution Engine

The execution engine simply executes the program code described in Section 3.5, which performs initialization and measurement taking [58], [59].

Random Number Generator (RNG)

The standard dictates that the random number generator is to be seeded by a truly random bitstream. The random numbers produced are used to generate cryptographic keys, nonce creation, and to strengthen passwords [58], [59].

SHA-1 Engine

As the name implies, this engine implements secure hash algorithm, SHA-1, which hashes the input data and produces a 20-byte digest. It is used for digital signatures and generation of a hash of key for the number of cryptographic procedures [58], [59].

RSA Key Generation

As we all know, the key generation can be computationally intensive, especially with the RSA algorithm. Thus, the standard requires the TPM to support keys up to 2048-bit modulus [59].

RSA Engine

Because of its complexity, RSA is housed in its dedicated engine. Like RSA, this engine is used for signing, encryption, and decryption. Note that encryption and decryption are done by separate storage key pairs [58].

Opt-In

As the name implies, the user has to opt-in to use the TPM, which means that the user has to take ownership and configure the TPM. In the process of taking ownership, the TPM will transitions through several states as described below. Changing the state of these flags requires authorization. The opt-in block ultimately provides mechanisms and protection to maintain the TPM state via the state of these flags.

Remote Attestation

A fundamental use case is determining the trust state of a platform (Figure 2 -1). The ability of a TPM to accomplish this objective depends on the proper implementation of Roots of Trust. Figure 2-1 shows the sequence diagram of the use case “Attest a platform”. First, the challenging party creates an unpredictable value. This value, known as a nonce, is simply a unique identifier and it is used for preventing an attacker from recording the response to a specific remote attestation request. The attestation party receives the attestation request (with the nonce). Then TPM (Trusted Platform Module as an implementation of Trusted Computing) functions are used to create a signature that is unique and uses an unforgeable key for the concatenation of the nonce and the value of a configuration register. Step 4 sends back a signature with the nonce from the module to the attestation party together with the value of the configuration register, and hashes for the bootloader, the kernel, and the application. Next (step 5), this is sent to the challenging party. This checks the signature and the nonce (step 6). Then, the challenging party checks the three hashes in its database of trusted bootloaders, kernels, and applications (step 7). When these are found, the challenging party creates the combined hash of all three components and compares it to the value of the configuration register received from the attesting side for validation.

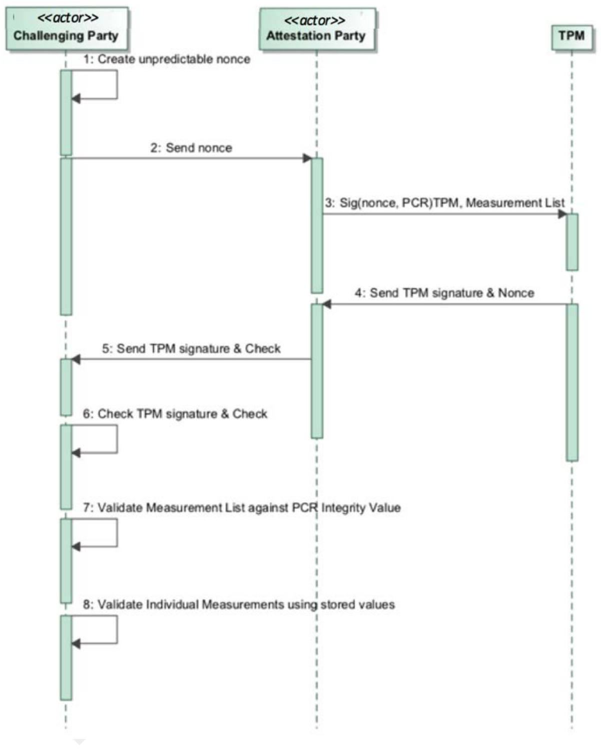
<<actor>> Challenging Party
<<actor>> Attestation Party
TPM
1: Create unpredictable nonce
2: Send nonce
3: Sig(nonce, PCR)TPM, Measurement List
4: Send TPM signature & Nonce
5: Send TPM signature & Check
6: Check TPM signature & Check
7: Validate Measurement List against PCR Integrity Value
8: Validate Individual Measurements using stored values

Figure 10: Sequence diagram of the use case “Attest a platform”

4.1.4 Trusted Computing Implementations

The most common implementation of Trusted Computing technology is known as a Trusted Platform Module (TPM), which is a device that enables trust in computing platforms (TPM Design principles 2011). It is split into several parts to make clear the role of each part, but all parts are required to follow TC defined standards. TPM is a system component that has a state that is separate from the system to which it reports (the host system). The only interaction between the TPM and the host system is through the interface defined in this specification. TPMs are implemented on physical resources, either directly or indirectly.

A TPM may be constructed using physical resources that are permanently and exclusively dedicated to the TPM, and/or using physical resources that are temporarily assigned to the TPM. All of a TPM's physical resources may be located within the same physical boundary, or different physical resources may be within different physical boundaries. Most TPMs are implemented as single-chip components that are attached to systems (typically, a PC) using a low-performance interface (such as, Low Pin Count, or LPC). The TPM component has a processor, RAM, ROM, and Flash memory. The only interaction with these TPMs is through the LPC bus. The host system cannot directly change the values in TPM memory other than through the I/O buffer that is part of the interface.

Figure 2-2 shows in a sequence diagram for the boot process in TPM, which starts in a predefined state with the execution of the BIOS "boot block" code. This is known as the Core Root of Trust for Measurement (CRTM). This is considered trusted due to its being in charge to initiate the boot. In essence, the integrity-protected boot process is based on taking hash-based measurements or fingerprints of the complete stack of executable code in the boot chain and this is stored in secure storage before that code is given control of the processor. Similarly, the CRTM takes measures of the BIOS code and stores the value in a Platform Configuration Register (PCR). TPM contains several PCRs. These allow secure storage and reporting of security-relevant metrics as measurements. These measurements can be used to detect changes to previous configurations and decide how to proceed to prevent platform integrity. The CRTM then passes control to the BIOS in charge of taking measurements of option ROMS, CPU microcode updates, and the OS loader before passing control to the latter.

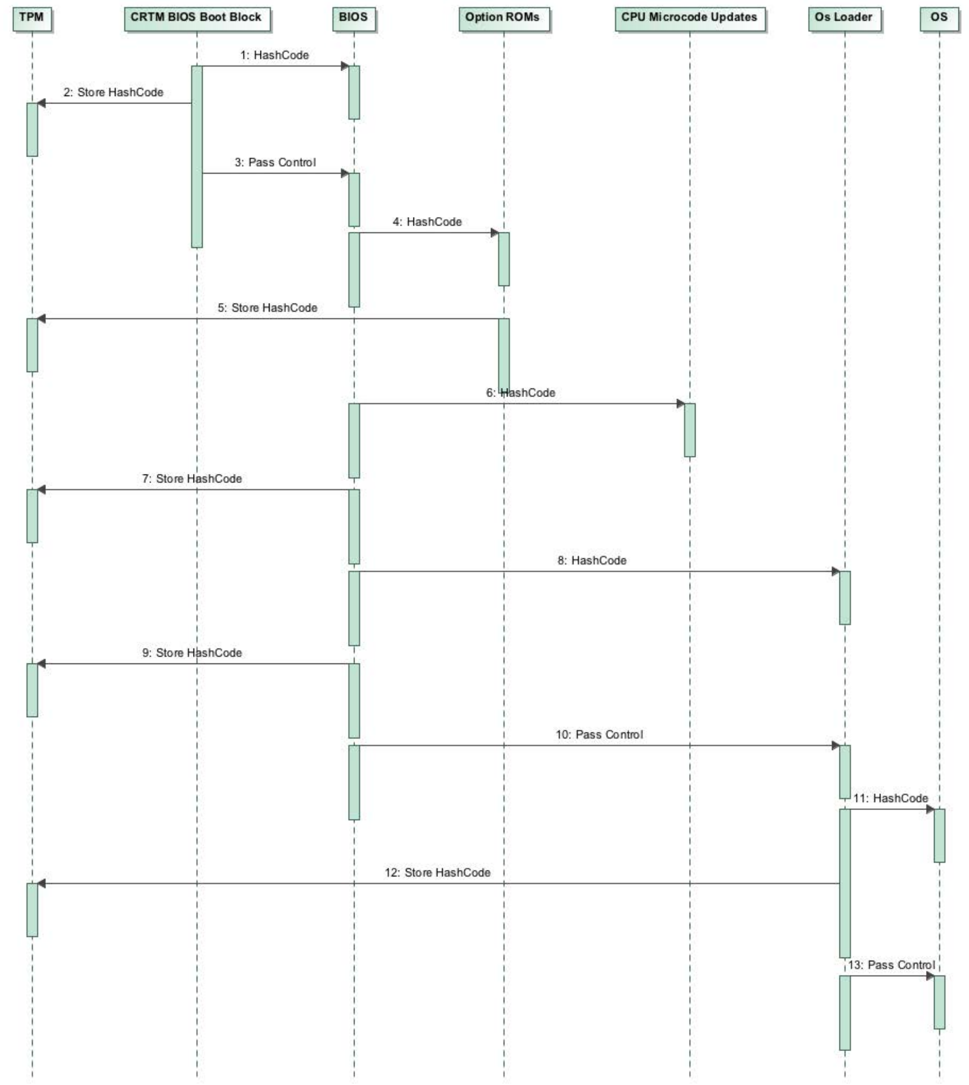

Figure 11: Sequence diagram of the use case "Boot Secure Chain of Trust in TPM"

TC has been used in the development of secure operating systems such as Microsoft's Palladium [60] and the open-source OS Linux's TrouSers (TrouSers). Palladium was cancelled several years ago. In the case of Trousers, having the TPM verify the BIOS means that placing rootkits in the BIOS is hard. Similarly, having the BIOS verify the OS means that placing rootkits in the OS is hard. Also having the OS verify the Application means that placing Trojans in applications is hard.

There are several solutions that only provide file/folder encryption keys protected by TPM such as HP, Infineon, and Information Security Corp solutions. VeriSign's PTA is software that transparently handles digital certificate functions, such as key management and storage while providing an enhanced experience for the end-user. This software provides Hardened PKI Protection and management of credentials issued by a Certificate Authority using TPM. BitLocker Drive Encryption is a whole-disk encryption feature included as part of the Enterprise and Ultimate editions of the Windows 10 operating systems (Home, Pro, and Enterprise).

Wave Systems' EMBASSY Trust Suite (ETS) delivers advanced levels of security to the client PC using the TPM security chip. Among functionalities we found file/folder encryption using keys protected by TPM; client-based Single Login based on username/password with auto-fill lets users remember only one password and register others in TPM for autofill as needed; Protected information repository TPM wrapping/sealing capability protects sensitive information; Enterprise login platform authentication using TPM; Remote access TPM-protected remote access credentials used for VPN, 802.1x, etc.; hardened PKI protection and management of credentials issued by Certificate Authority using TPM; TPM system backup and key recovery in cases of platform failure, platform replacement, or hard drive failure; TPM and User Management manage TPM and user security policies; and Platform attestation privacy CA capabilities to create AIKs (Attestation Identity Keys) using TPM.

Ultimaco Safeguard [61] from IBM SafeGuard Easy (SGE) delivers end-user protection for sensitive information on laptops and workstations. Ultimaco provides full hard disk encryption using Keys that are protected by the TPM; Container encryption keys for container access as well as content encryption are protected by the TPM; Workgroup data is protected by encryption and keys are stored in the TPM; High-security user authentication by means of pre-boot-authentication user is authenticated before operating system is up and running. Machine binding data on media is linked to the PC platform, enforced by TPM credentials. Secured remote administration and mutual authentication of clients and servers for remote administration is secured by keys generated by the TPM. Softex OmniPass and Theft Guard provide file/folder Encryption Keys protected by TPM. Client-based single Log-in username/password with autofill as other tools; and protected information repository. Thales HSM is a Surveillance, detection and intelligence system [62].

A myriad of different attestation mechanisms has been implemented on TPM functionalities. In this section, we included those we considered the most relevant. We found an approach based on the verification of software status and code executed in a system known as IMA [63], which stands for Integrity Measurement Architecture. Hash values are computed from binaries before execution. This approach is considered the first practical implementation of the Trusted Computing Group remote attestation mechanism on Linux OS and is based on a simple concept. Nevertheless, this approach presents two major limitations since the hash verification of an executable file does not provide information about its behaviour while running. Also, this approach lacks flexibility and dynamism due to the fact that patches and upgrades of programs are not considered. The next approach based on IMA is known as PRIMA [64], it stands for Policy-reduced Integrity Measurement Architecture, and is based on taking integrity measurements enhanced with information flow control using SELinux policies. Analysing this approach, we found some relevant shortcomings. This approach also relies on the binary measurements of trusted subjects since the information flow it considered is just between low-level and high-level integrity subjects. Therefore, PRIMA needs a complementary approach to capture the dynamic behaviour of general-purpose platforms that needs verification of many other security policies.

The property-based attestation approach [65] is built on the claim that an approach should not depend on software or hardware configurations as Trusted Computing Group specifications indicate for attestation and sealing. This approach proposes to perform evaluations of the trusted state of the systems using the properties, as a whole of the system itself. To some extent, this proposal solves the problem of privacy found in present security component strictly adhering to the Trusted Computing Group specification. This mechanism provides a more flexible attestation mechanism that IMA. However, this approach needs additional TTPs that extract properties from configuration received releasing property certificates to the challenger. The flexibility of this solution is interesting since it is based on properties rather than fixed to specific configurations of the system, thus it is more open to heterogeneous environments. However, it is not clear how mapping configurations to properties is done, which is not described nor published and probably not yet defined. Indeed, according to [66] it has been difficult to agree on which are these high-level properties. The involvement of an additional TTP requires strong trust relations with additional costs and besides increases the chance of inceptions and wrong interpretations.

Haldar et al. [67] proposed and developed the Semantic Remote Attestation mechanism as an attempt to create a platform-independent remote attestation technique. This approach proposes the use of a Trusted Virtual Machine (TVM) with the capability to enforce the requirements for those applications that run within this virtual machine. This technique is more flexible compared with other binary attestation techniques and tends to measure the behaviour of code that runs inside a TVM. In a similar way to PBA, there is no semantic description to let a general program to be remotely attested using this approach. Another issue is that there is no implementation of this technique made public. Remote Attestation on Program Execution [68] proposed to measure system calls made during program execution to be analysed for possible malicious actions. This approach is an evolution of binary attestation since it captures application runtime behaviour from the surveillance of system calls and the underlying interaction with the platform creating a system trap table (SysTrap). The main drawback of this approach is that only system call measurements are considered. This fact hinders to decide whether system calls were made for malicious purposes to a challenger. Another issue is derived from the SysTrap table that is protected using sealing. This fact forces that a sealed object to be unsealed on the same platform where sealing was performed, which means that the object will have to be unsealed during attestation challenge and thus it would be prone to attacks during that time. Trustable Remote Verification of Web Services [69] is an approach for attesting web services using Trusted Computing concepts. This approach is based on having static compile-time checking of the code that performs the operations depicted by the web service. They describe operations using Contracts defined in JML and implement the concept of transitive trust. The strongest contribution of this approach is the preservation of web service privacy while still being able to perform attestation. Nevertheless, this approach presents important shortcomings, although this technique pretends to perform web service behavioural attestation, it only makes attestation of the code. The fact that its reliability depends on the concept of transitive trust can be a drawback since it is still controversial in the literature today.

Model-Based Behaviour Attestation (MBA) [70], is a high-level framework for identifying, specifying and remotely attesting the behaviours associated with different components of a policy model. This technique combines existing techniques extracted from different scenarios specified by the challenger in a behavioural policy that includes the details on how the behaviour of the platform will be attested. In this line, three different behaviour profiles are described (i) active subject/object behaviour; (ii) state transition behaviour; and (iii) attribute update behaviour. Since this technique is not tied to a particular software or hardware platform it is more flexible than other techniques and makes it capable to attest a wide variety of security models. This technique combines different remote attestation techniques to create a remote attestation technique capable and flexible to meet real scalable computing environments requirements. This approach lacks implementation details . There is another relevant drawback to be considered if a measurement agent operating at the remote platform might be trusted since this leads to a circular argument to be solved before a plausible MBA implementation is ready to be provided.

4.1.5 Approaches

In [71] is presented the design and implementation of a system that enables trusted computing for an unlimited number of virtual machines on a single hardware platform. They virtualized TPM and integrated their approach into a hypervisor environment to make TPM functions available to virtual machines. As a result, the TPM's secure storage and cryptographic functions are available to operating systems and applications running in virtual machines. Our new facility

In England et al. [72], the authors describe a paravirtualization design that allows a VMM to time share a TPM among its VM. The concept of associating a TPM context to a particular VM is proposed. A TPM context will contain important data that defines a TPM state, for example, keys and sessions. When a particular VM wishes to use the physical TPM, the associated TPM context is loaded into that physical TPM.

Some Techniques focuses on enhancing TPM with Hardware-Based Virtualization [73]. The work found in [72] presents the design of a TPM that supports hardware-based virtualization. In this design, the VMM time multiplexes the hardware TPM in a manner similar to. However, the difference is that the hardware TPM has to be modified to include additional non-volatile memory to store the various TPM contexts.

4.2 TPM Security Issues

This section includes the most known security issues related to TPMs. The first issue included is the most fundamental problem faced by the users of TPMs: how to bootstrap trust in the computer containing the TPM. Followed by that, we present two issues related to the TPM "authorization data" or authdata as mentioned in the TCG documentations. We also discuss security issues associated with the use of the TPM in authenticating processes running on a platform and the security loophole that allows attackers to reset the secured measurements stored in the PCRs.

As we have discussed earlier, one of the main functions of the TPMs is to establish trust in the software executing on a computer. However, a crucial aspect the TPM specifications have overlooked is how to bootstrap trust in a computer containing the TPM to begin with. For example, if a user does not trust his computer, he cannot perform most of the activities such as doing online banking, reading/writing confidential documents, visiting confidential websites, etc., regardless of the existence of a TPM.

According to the TCG specifications, the TPM's manufacturer provides the TPM with an Endorsement Certificate (EC) that certifies that the TPM is a genuine, hardware TPM and authenticates the TPM's public key, this is known as Cuckoo Attack. Although, the EC only guarantees that the public key belongs to some TPM, thus allowing an adversary to mount a cuckoo attack.

TPM stores cryptographic keys and other sensitive data in protected locations within the TPM itself. In order for a user process to access these protected items, the process has to prove to the TPM the knowledge of the corresponding authdata. Currently, this is done by using authdata as an HMAC key in any communication between the user process and the TPM. The authdata is chosen by the user process at the time of storing the protected items within the TPM. The TPM specification allows up to 160 bits for authdata allowing it to be a high-entropy value. However, since the TPM specification does not impose any restriction on the entropy of the authdata, the user process can naively derive the actual authdata from a human-memorable low-entropy password. This is when it becomes feasible for an attacker to mount an offline dictionary attack on the authdata and extract the sensitive items stored by a user process in the TPM. In [74], Chen and Ryan identify this attack and proposes three attractive approaches to change the existing TPM command set.

In [75], the authors show that the sharing of authdata as currently allowed in the TPM specification has a major undesirable consequence. Specifically, an adversarial process sharing the authdata with honest processes can mount a two-way impersonation attack. In other words, the adversarial process can impersonate an honest process to the TPM and obtain access to the protected items stored by that process, and also the adversarial process can impersonate the TPM to the honest processes and fake all the storage capabilities of the TPM enabling an Impersonation Attack on Shared authdata. The authors in [75] actually present their solution as a new authorization protocol named Session Key Authorization Protocol (SKAP). Next, using ProVerif [76], which is a widely used tool for assessing the security properties of protocols, the authors have shown that their new protocol prevents the impersonation attack.

Currently, TPM implementation considers only static measurements of the state of data and software. This means that measurements are taken once during load time and changes in the measured values afterwards are not monitored. A potential attacker can modify the loaded binary in the RAM. This is possible given the kernel vulnerabilities, for example in Linux, that allowed attackers to bypass memory mapping and IPC restrictions. Such modifications of a TPM trusted process are not reflected in the TPM software state measurements and thus constitutes a security breach. The modification of the program in the RAM leaves a security hole that can be exploited by an adversary to supply crafted input that likely to have an influence on the behaviour of the program enabling a software attack known as Time-of-Check Time-of-Use Vulnerability.

4.3. TPM Assisted Technologies & Research

4.3.1 TPM Assisted Identity Management & Authentication

TPM chips have a unique and secret RSA key burned in as they are produced; this means that TPM can be used by computer programs to authenticate hardware devices. Applying security measures to the hardware level provides more protection than a software-only solution.

Authentication is frequently required by operating systems (via password or other means) to protect keys, data or whole systems. In case the authentication mechanism is implemented in software only, the access is prone to dictionary attacks. Because TPM is realized in a dedicated hardware module, a brute force attack prevention mechanism was built in, which in turn protects against guessing or automated dictionary attacks, while still allowing the user a decent and reasonable number of tries. In the absence of this level of protection, only passwords with high complexity would provide sufficient protection. [77]

TPM can also be used for attestation, which is to present verifiable evidence about a machine to a remote party. Most TPM keys can be used for machine authentication, as TPM chips are soldered to the motherboards, while at the same time keys are cryptographically bound to TPM. When referring to machine authentication, the types are either signing-based or decryption-based.

4.3.2 TPM Assisted FIDO

TPMs typically have a built-in attestation capability however the attestation model supported in TPMs is currently incompatible with UAF's basic attestation model. The future enhancements of UAF may include compatible attestation schemes.

Typically, TPMs also have a built-in PIN verification functionality which may be leveraged for UAF. In order to support UAF with an existing TPM module, the vendor should write an Authenticator Specific Module (ASM) which [78]:

- Translates UAF data to TPM data by calling TPM APIs
- Creates assertions using TPMs API

- Reports itself as a valid UAF authenticator to FIDO UAF Client

A special Assertion Scheme, designed for TPMs, must be also created and published by FIDO Alliance. When FIDO Server receives an assertion with this Assertion Scheme it will treat the received data as TPM generated data and will parse/validate it accordingly.

TPM also allows the implementation of Direct Anonymous Attestation (DAA) using elliptic curves as basic building blocks for its implementation are part of the TPMv2 specification [79]. The scheme is also known as ECDAA [80]. and provides significantly improved performance compared with the original DAA. ECDAA combines security and privacy as each authenticator has its own attestation key (instead of the "group keys" proposed in Basic Attestation) however there is no risk of a privacy leakage as the authenticator uses a specific secret once to communicate with a DAA Issuer and uses the resulting DAA credential in the DAASign protocol with each relying party.

4.3.3. TPM Assisted Secure Boot

TPM has been adopted and utilized by operation systems (e.g. Windows 8,8.1), which are making use of these micro-chips in a number of areas, including the secure boot process. TPMs are used in the process of checking the integrity of every component during the start-up, before loading it in the operating system. More specifically, when a computer system is initiated, the firmware checks the signature from each piece of the boot software, including firmware drivers and the operating system itself. If the signatures are valid and verified successfully the PC boots, and the operating system takes control from the firmware. Moreover, it enables remote attestation of the code stack on a running system, which means that the chain of trust firmware records the hash of the loaded firmware and stores the records in the network of TPM processors. The network can consist of one physical TPM per master processor on low-to-mid range platforms, or redundant TPMs through master or alternate-master processors on multi-node enterprise platforms. The chain of trust firmware also records all events appropriately in TPM event logs. This way it is not possible to boot an infected or tampered operating system which could cause a plethora of problems to its users. [81]

4.3.4 TPM Assisted Device Attestation

TPMs have embedded in them the functionality of remote attestation that has been proposed as a promising security mechanism. It allows a third party, the verifier, to ensure the integrity of a remote device, the prover. Although the capability of remote attestation has existed for years in TPMs, it remains a vibrant research topic since the actual protocol can vary in both security and performance. One solution that proposed the feasibility of control-flow attestation schemes was C-FLAT [171]. C-FLAT is an attestation protocol that measures the valid execution paths executed by low-end devices although it incurs significant performance overhead; thus, it is suitable only for small size binaries. Subsequent works such as LO-FAT [172] and LiteHAX [173], aim to reduce this attestation overhead by proposing new protocols which are either optimized (control-flow graph shortened) or utilize commonly installed hardware accelerators to ease the strain for the device that is undertaking the attestation.

Most of the aforementioned protocols were designed to mainly support single-device attestation where a trusted party, the verifier is able to check the integrity of a single remote device, the prover. However, as aforementioned, applying such techniques results in a huge overhead, not just to end devices but also to the network that handles all the attestation messages. Towards this direction, collective attestation protocols like SANA [174], SEDA [175], DARPA [176] have also been proposed that decongest the network traffic by collecting many attestations to single data packets, thus, reducing the traffic introduced by a large number of edge devices trying to attest to their integrity. This is achieved by securely aggregating the attestations of multiple devices in a single token through which the verifier must be able to confirm that each of the involved devices proved correctly that their execution flow is correct. [177]

4.3.5 TPM Assisted Digital Forensics

The TPM is another emerging technology that will enhance existing encryption schemes. The TPM is a chipset, that provides for hardware-based encryption functionality, being installed in newer machines that stores keys, passwords, and certificates.

Computer forensics in non-trusted computing still faces several challenges such as [82], [83], [84], [85], [86]:

1. **Malicious people:** Malicious people can plant digital data to frame someone for a crime. For this reason, the criminal can claim in court that the crime was committed by a third party such as a malicious user or software;
2. **Volume scalability:** The size of user data storage is increased from time to time, this issue is called volume scalability;
3. **Lack of a balance:** There is a need to create a balance between privacy protection and the digital forensic process;

   Digital forensics tries to reveal the identity of users involved in any kind of digital crime, while privacy protection techniques protect the identity of a user. Thus, finding a balance between digital forensics and privacy is an active topic today. In non-trusted computing, several pieces of research have been conducted to address this issue. For example, [87][88][89][90] proposed some privacy protection protocols. These protocols are privacy protection techniques to protect user's privacy. In trusted computing, there is a need to study how a Privacy Certificate Authority (Privacy CA) and Direct Anonymous Attestation (DAA) protocols can be refined to be integrated with digital forensics (how to reveal the identities of criminal users if needed). The Privacy CA and DAA are two trusted computing protocols used for preserving the privacy of users in trusted environments.

4. **Different criminal tools**: Criminals have several tools (such as data hiding and encryption tools) to avoid leaving digital evidence.

Regarding these challenges in non-trusted computing forensics, it is clear that trusted computing forensics will solve the first challenge by using remote attestation, digital signature, and sealed storage functions. The second and third challenges will remain open issues as in non-trusted computing forensics. Finally, the last challenge will be a greater problem with trusted computing because it will provide criminals with a strong tool to avoid investigation, namely full disk encryption.

Fast forensics is another concept in this area. Fast forensics is defined as "those investigative processes that are conducted within the first few hours of an investigation, that provide information used during the suspect interview phase. Due to the need for information to be obtained in a relatively short time frame, fast forensics usually involves an on-site/field analysis of the computer system in question." The implementation of fast forensics caused to have some additional resources and procedures to perform some examination and initial analysis functions outside the lab. The focus is to provide important intelligence to give investigators key pieces of evidence or leads to use in interviews or other searches. Some fast-forensic techniques utilize Linux or other forensic boot disks to perform on-scene searches or document extraction. The boot disks run in memory only and mount the hard drives as read-only so as not to corrupt the evidence.

4.3.6 Other TPM Assisted Technologies

Integrating trusted computing with other security techniques is discussed in different researches that focused on hardware-based security solutions. For example, to mitigate botnet attacks on internet banking websites, in [91], authors proposed a solution that integrates the existing technologies using a biometric USB thumb drive which needs to be built with a fingerprint sensor, RSA SecurID and TPM chip. The proposed solution is based on using the biometric fingerprint sensor on a USB thumb to identify the actual owner of the flash that could prevent unauthorized access and theft of the device. The capability of TPM that is embedded inside the USB thumb is to store and protect the fingerprint of the user for whom the thumb is issued by the e-banking service. All the messages that are exchanged between the bank server and the endpoint device should be encrypted using the built-in TPM on both ends that is responsible for encrypting, decrypting and verifying the header files of the messages.

To establish hardware-based trust in the log producing application without manipulation of existing logs, authors in [92] proposed a solution based on TPM and AMD’s Secure Virtual Machine technology (SVM) to establish a root of trust for client syslog daemons. Without considering the effects of software manipulation, their solution is based on enabling a hardware-based trust-relationship between log entries and logging machines which ensures that log data and log producer can be trusted.

ReVirt which is represented in [93] is an SVM-based logging and replaying system that performs system logging at the Virtual Machine Monitor (VMM) layer and removing the need to trust the operating system. Besides that, it also records all the operations that are necessary to recreate an entire attack. Logging is done at an instruction-by-instruction level, this process gives the administrator a complete picture of the entire process that happened on the guest virtual machine even in the presence of non-deterministic attacks and execution. Since ReVirt is a powerful logging tool in SVM, it may leak sensitive information if the logs are not protected in a correct manner. If an attacker manages to get hold of ReVirt logs, he/she will gain access to all the sensitive information. However, it is very difficult to remotely control the ReVirt logging virtual machine and one method to minimize this threat is to encrypt the ReVirt’s logs.

Other TPM technology can be found in System Management Mode (SMM) [94] that is a mode of execution similar to Real and Protected modes available on x86 platforms (Intel started to use SMM in its Pentium processors since early 90s). It provides a hardware-assisted isolated execution environment for implementing platform-specific system control functions such as power management. It is initialized by the Basic Input/Output System (BIOS). SMM is triggered by asserting the System Management Interrupt (SMI) pin on the CPU. This pin can be asserted in a different way, which includes writing to a hardware port or generating Message Signalled Interrupts with a PCI device. After that, the CPU saves its state to a special region of memory called System Management RAM (SMRAM). Then, it atomically executes the SMI handler stored in SMRAM. SMRAM cannot be addressed by the other modes of execution. The requests for addresses in SMRAM are instead forwarded to video memory by default. This caveat, therefore, allows SMRAM to be used as secure storage. The SMI handler is loaded into SMRAM by the BIOS at boot time. The SMI handler has unrestricted access to the physical address space and can run privileged instructions. The RSM instruction forces the CPU to exit from SMM and resume execution in the previous mode[95].

Moreover, Trust Computing Group (TCG) introduced Dynamic Root of Trust for Measurement (DRTM) [96], also called "late launch", in the TPM v1.2 specification [97] in 2005. It is an alternative to the Static Root of Trust for Measurement (SRTM). Unlike SRTM which operates at boot time, DRTM allows the root of trust for measurement to be initialized at any point. To implement this technology, Intel developed Trusted eXecution Technology (TXT) [98], providing a trusted way to load and execute system software (e.g., OS or VMM). TXT uses a new CPU instruction, SENTER, to control the secure environment. Intel TXT does not make any assumptions about the system state, and it provides a dynamic root of trust for late launch. Thus, TXT can be viewed as a hardware-assisted isolated execution environment to run security-sensitive tasks. AMD has a similar technology called Secure Virtual Machine [99], and it uses the SKINIT instruction to enter the secure environment. Note that both TXT and SVM introduce a significant overhead on the late launch operation.

Additionally, Flicker [100] and TrustVisor [101] employ DRTM with a small trusted computing base to create a HIEE. Flicker creates an on-demand secure environment using DRTM, while TrustVisor uses DRTM to securely initialize a light-weight hypervisor that uses hardware virtualization (VT-x/SVM) to protect the applications running in the secure environments. The two systems use the TPM to provide remote attestations and to securely store data for executing sensitive workloads. Bumpy [102] is a Flicker-based system for securing sensitive network input. It handles inputs in a special code module that is executed in an isolated environment using the Flicker.

It should be noted that there are several issues that are still not clear and need further work [103]. First, there is still a research gap when it comes to proposing a trusted computing digital forensic framework. At least one tool must be developed for identifying, collecting, and analysing digital evidence from the Stored Measurement Log (SML). The SML is a system file that stores the last values of the Platform Configuration Registers (PCRs) but with more details. The PCRs contain hash values representing the configuration specifications of the platform’s hardware and software components, where each component has a hash value generated by hashing the component’s specifications and storing them in the PCR. For example, one PCR is used for storing the integrity measurement of the Java Virtual Machine (JVM) but the SML stores the integrity measurements of all of the installed java applications.

5. TEE/TPM Assisted Blockchain

5.1. Blockchain Background

5.1.1. Blockchain

Blockchain is essentially a distributed ledger shared among multiple entities, that carries out the process of documenting transactions and tracking assets in a network. Practically anything of value can be traced and traded on a blockchain network, with reduced risk and minimal costs for all parties involved. In other words, blockchain provides a mechanism to record and store transactions, e.g. bitcoins. Of course, blockchain is not restricted to bitcoin and has many uses beyond that, as bitcoin is only the first use case for blockchain. [104]

Blockchain networks are cost-effective and efficient, as they render duplication of effort obsolete and diminish the need for intermediaries. It can also be described as safe and protected because it uses consensus models to validate the information. Transactions are secure, authenticated, and verifiable. The participants in both the traditional and the blockchain transaction systems are the same, with the difference that the transaction record is now public and shared by all parties. [105]

The main features and traits a blockchain network has to offer are:

- Consensus: In order to consider a transaction as valid, all participants must agree on its validity.
- Provenance: All network entities know the asset's origin and how the owners that have possessed it over time.
- Immutability: The network's entities are not able to tamper with a completed transaction, as it has already been recorded to the ledger. If a transaction was erroneously made, a different transaction has to take place in order to reverse the mistake.
- Finality: In order to identify whether a transaction has been completed, or who is the owner of an asset, a look up at the shared ledger is needed.

From a business' perspective, the use of a blockchain network provides the following benefits:

- Timesaving: Transaction times for complex, multi-party interactions are dramatically reduced compared to the traditional exchange systems. This is mainly because its verification by a central authority is not required, but instead, every procedure needed is carried out by the network's entities.
- Cost-effective: The use of a blockchain network is economically beneficial for businesses because:
    - It requires less supervision as the network's health is monitored by its entities and not an external third party.
    - Mediators and brokers are not necessary because participants can exchange items of value directly.
- Security: Blockchain has features by design regarding protection against tampering with transactions, and consequently fraud and cybercrime. It is possible to create a blockchain network of a members-only nature, which means that each entity will have proof that they are indeed who they claim, while at the same time goods and assets are traded exactly as represented.

One of blockchain's main strengths is that it provides a high level of trust among its network participants. On account of building every new transaction on top of every other past transaction, possible breakage of trust and misbehaviour of specific entities are obvious and every individual in the network is made aware of it. By taking care of itself for malicious behaviour, the blockchain network does not have to hide behind legal and government shields to ensure its smooth and unobstructed operation; instead, it is an autonomous and decentralized system that is essentially self-sustained. If a third-party supervision is deemed necessary, blockchain can be "audited" very easily as every information that may be required by the overseers is instantly available and open to anyone.

Trust in blockchain networks is built through:

- Distribution and sustainability: The ledger is available for everyone, gets revised every time a new transaction takes place and is selectively copied among participants. Because it does not belong to a single organization, neither is controlled by one, the blockchain networks are independent of any individual entity and third-party decisions.
- Security, privacy, and indelibility: Unauthorized access to the network is prevented by applying permissions and cryptography, ensuring at the same time that participants are who they claim to be. It is possible for the privacy to be preserved by using cryptographic techniques and/or data partitioning techniques to give participants selective visibility into the ledger; both transactions and the identity of the involved parties can be masked. Participants can't alter the record of transactions; errors can be reversed only with new transactions.
- Transparency and auditability: The involved parties in a transaction have access to the same records, so they can validate transactions and verify identities or ownership without the need for third-party intermediaries. The transactions are time-stamped and can be verified in near real-time.

- Consensus: All or a predefined number of network participants must agree that a transaction is valid. This is achieved by using consensus algorithms. Each blockchain network can set the conditions under which a transaction or asset exchange can occur.

The name "blockchain" is derived by the way transaction data is stored, which is in blocks that are linked together forming a chain. As the number of transactions grows, so does the blockchain. Blocks record and confirm the time and sequence of transactions, which are then logged into the blockchain, within a discrete network governed by rules agreed on by the network participants.

Each block stores a hash, otherwise thought as a digital fingerprint or unique identifier, timestamped clusters of valid transactions, and the hash of the previous block. The previous block hash consists of the link between the blocks and prevents any block from being altered or a block being inserted between two existing blocks. This way, each subsequent block essentially verifies the previous block, hence the entire blockchain. The method makes the blockchain tamper-evident.

The main purpose of the blockchain is to hold transaction data, but it is not designed to store large amounts of data, exchanging messages or processing transactions. The blockchain contains verified proof of transactions, however, while blockchain essentially serves as a database for recording transactions, it benefits its users far beyond the advantages a traditional database offers.

People have been using ledgers for many decades, since back when they were not in digital form. With blockchain, the concept of a shared and distributed ledger was introduced, which is essentially an immutable record of all transactions on the network, which is also accessible by all network participants. By using a shared ledger, transactions are recorded only once, eliminating the duplication of effort that's typical of traditional business networks. The main characteristics of a shared ledger are described below:

- Documents every transaction that takes place across the network; the shared ledger is the system of record, where everything can be verified.
- It is shared and distributed among all the network's entities; using the replication method, each participant is in possession of a copy of the ledger.
- Relies on permissions, which means that the participants can observe only the transactions they have been authorized to view. The network entities have identities that link them to transactions, but they can make a decision regarding the transaction information other participants are authorized to view.

In a blockchain network where the participation list is transparent and the entities are trusted with each other, transactions can be verified and committed to the ledger through various means of consensus (agreement), including:

- Proof of stake: To validate transactions, a group of participants must hold a certain percentage of the network's total value. Proof-of-stake might provide increased protection from a malicious attack on the network by reducing incentives because of these attacks' resource-consuming nature.
- Multi-signature: Most of the validators (e.g. seven out of ten) must agree that a transaction is valid.
- Practical Byzantine Fault Tolerance (PBFT): An algorithm designed to detect when one node in a set of nodes generates different outputs from the others in the set, which may indicate malicious behaviour.

5.1.2 Smart Contracts

A contract is an instance of a computer program that runs on the blockchain, i.e., executed by all consensus nodes. Smart contracts have unique addresses that are used to identify them on the blockchain. Smart contracts are programs running on cryptocurrency (e.g., Ethereum) blockchains, whose popularity stems from the possibility to perform financial transactions, such as payments and auctions, in a distributed environment without the need for any trusted third party. Ethereum smart contracts are written in Solidity, a dedicated language resembling JavaScript, and shipped over the blockchain in the Ethereum Virtual Machine (EVM) bytecode format [106]. A contract is a set of functions, each one defined by a sequence of bytecode instructions. A remarkable feature of contracts is that they can transfer ether (a cryptocurrency similar to Bitcoin) to/from users and to other contracts. We propose a taxonomy of smart contracts into five categories, which describe their intended application domain [107].

- Financial
- Notary
- Game
- Wallet
- Library

We should add that, the famous platforms for smart contracts are as follows:

- **Ethereum** is the largest and most popular platform for building distributed applications, ranging from social networks and identity systems to prediction markets and many types of financial applications [108].
- **Bitcoin** is both a system and a cryptocurrency that was first introduced by Nakamoto [109]. Bitcoin offered a new, enhanced way of capital circulation, new markets, and new decentralized autonomous organizations. Although the main goal of Bitcoin is to transfer currency, the immutability and openness of its blockchain have inspired the development of protocols that implement (limited forms of) smart contracts.

5.2 Blockchain Security Applications

Taxonomy of blockchain-based applications:

Most authors classify blockchain applications into financial and non-financial ones [110]. In this document, we analyse an application-oriented classification, similar to the one proposed in [111]. We present a more comprehensive and in-depth classification of blockchain-based applications, which is graphically represented in Fig. 3-1. In the following, we provide a summary of the most popular and available blockchain-enabled applications based on the analysis of the available literature.

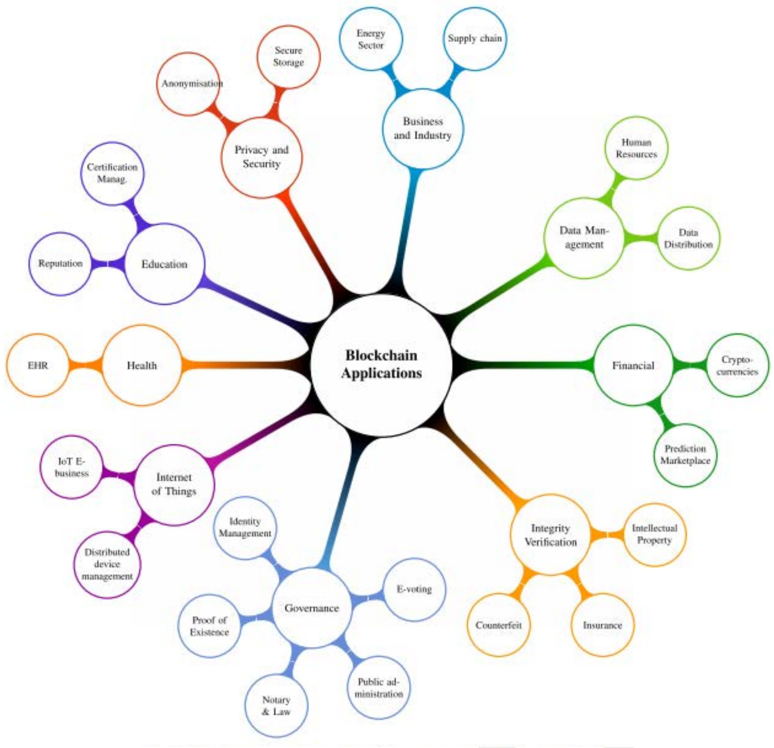

Figure 12: Mindmap abstraction of the different types of blockchain applications. [180]

Financial applications

Currently, blockchain technology is applied to a wide variety of financial fields, including business services, settlement of financial assets, prediction markets and economic transactions [112]. Blockchain plays an essential role in the sustainable development of the global economy, bringing benefits to consumers, to the current banking system and the whole society in general [113]. Blockchain technology offers a massive change to capital markets and a more efficient way for performing operations like securities and derivatives transaction[114],[115], digital payments[116],[117], loan management schemes [118], general banking services [119], financial auditing [120] or cryptocurrency payment and exchange (i.e., e-wallets) [121],[122].

Integrity verification

One of the most emerging blockchain-related fields is integrity verification[123],[124]. Blockchain integrity verification applications store information and transactions related to the creation and lifetime of products or services. The possible applications are (i) provenance and counterfeit, (ii) insurance; and (iii) intellectual property (IP) management.

Governance

Governments throughout the years are entrusted with managing and holding official records of both citizens and/or enterprises. Blockchain-enabled applications might change the way governments at the local or state level operate by disintermediating transactions and record-keeping [125]. The accountability, automation, and safety that blockchain offers for handling public records could eventually obstruct corruption and make government services more efficient. In particular, blockchain could serve as a secure communication platform for integrating physical, social, and business infrastructures in a smart city context[126];

Citizenship and user services

The integration of digital technologies in everyday life requires mechanisms able to determine accurately who the users are[127] and certify their basic attributes like name, address, credit record, as well as other personal characteristics [128]. According to[129], one-sixth of the world's population lack documented proof of their existence. This situation affects immigrants and refugees, since their countries may often refuse to hand over the documents if, for instance, they belong to the opposition. Therefore, blockchain becomes an instrument to reinforce equality and opportunities to worldwide citizens.

Voting

For several years e-voting has been considered a promising and inevitable development that could speed up voting processes, simplify and reduce the cost of elections, and the development of stronger democracies [130]. However, existing electronic voting systems rely on a proprietary and centralized design by a single entity, characteristics that harm the trust and confidence voters have to the voting process [131].

Internet of things

Around 90% of the data in the world today has been created in the past two years alone [132]. Such growth pace will increase due to a) the advent of the Internet of Things (IoT), b) to the population growth [133]. While the expansion possibilities of the blockchain and IoT technologies are already vast on their own, the symbiotic relationship between these two fields arises myriad more.

For instance, the distributed wireless sensor networks, which despite their drawbacks [134] are one of the pillars of technological and human evolution, demonstrate that blockchain architecture may enhance IoT by minimizing its deficiencies and maximizing its potential. The increasing attention and investments for implementing decentralized IoT platforms[135] are mainly driven by the blockchain technology and its inherent capabilities. The main idea is to provide secure and auditable data exchange in heterogeneous context-aware scenarios (Casino et al., 2017) with plenty of interconnected smart devices (Crosby et al., 2016).

Healthcare Management

Blockchain technology could play a pivotal role in the healthcare industry with several applications in public healthcare management, longitudinal healthcare records, automated health claims adjudication, online patient access, sharing patients' medical data, drug counterfeiting, clinical trial, and precision medicine [136]. Managing patients' Electronic Healthcare Records (EHRs) is probably the area with the highest potential growth [137],[138],[139]. An EHR contains a patient's short medical history, as part of her medical record, as well as data, predictions, and information of any kind relating to the conditions and the clinical progress of a patient throughout the course of treatment. A blockchain system for EHRs could be seen as a protocol through which users may access and maintain their health data that simultaneously guarantees security and privacy[140],[141].

Privacy and security

Centralized organizations – both public and private – amass large quantities of personal and sensitive information. Although the GDPR [142] aims to regulate the processing of this data, there is still a big gap to cover [143]. Blockchain is considered as an opportunity for enhancing the security aspects of big data [144] and its scalability when combined with other efficient storage systems that implement data mining methods [145]. Therefore, privacy and security-oriented applications that rely on blockchain technology can be found in the literature [146][147][148].

Energy Sector

The potential applications of blockchain in the energy sector have an enormous impact both in terms of processes as well as platforms [149]. For example, blockchain may reduce costs and enable new business models and marketplaces, can better manage complexity, data security, and ownership along grids, can engage prosumers in the energy market acting as an enabler for the creation of energy communities [150][151], can enhance the transparency and trust of the energy market system, can guarantee accountability while preserving privacy requirements, can enhance direct peer-to-peer trading to support the smooth operation of the power grid, and can better handle demand response and provide a framework for more efficient utility billing processes and transactive energy operations [152][153]

5.3 Smart Contracts Security Applications

Smart contracts can be used for identity management in conjunction with traditional public key infrastructures to ensure dynamic management of trust. [154] All entities interacting with a blockchain (including users and IoT) must have one public and one private key at least. Regardless of the actual key usage, these keys can be used as roots of trust with the smart contracts playing the part of a digital certificate. With this method, a smart contract must contain an array of various identifiers correlated to the public key owner. As an added side-effect, only the owner of the contract can modify this array of identifiers and given that the smart contract exists within the blockchain, additional security guarantees are provided. [155]

Smart contracts can be easily configured to restrict who can modify their variables to specific blockchain users. Besides this, smart contracts can be used for user authentication by using a method that triggers a smart contract event whenever it is called. That is, every time a smart contract user receives this event it can obtain the blockchain public key of the user that invoked the method in the first place. With both these functionalities, it is evident that smart contracts can be used as an access control tool that provides high security and accountability.

Furthermore, smart contracts can be used for proof of ownership, information verification and non-repudiation. By using the smart contract technology, users can announce that they own an item (thus have a proof of ownership), as well as create transcripts of interactions through the blockchain. An example found in [156] extends TLS technology to create proofs for the contents of a TLS session which are then sent to a blockchain, indicates that this solution can be used to extend existing solutions. Finally, since blockchains are append-only ledgers and cannot be modified, the system can achieve non-repudiation, that is, it is not possible for users to claim that they did not approve a transaction.

5.4. Blockchain Security Issues

Despite the security features blockchain inherently possess, cybersecurity issues do exist [157][158]:

- The 51% attack is the most common attack that is attempted on blockchain networks. In a 51% attack, one, or several, malicious entities gain majority control of a blockchain's hash rate. With the majority hash rate, they can reverse transactions to perform double-spends and prevent other miners from confirming blocks. Essentially, a group of miners controls more than 50% of the network's mining hash rate or computing power. In such a scenario, the attackers are able to prevent new transactions from gaining confirmations and halt payments between some or all users. They would also be able to reverse transactions that were completed while they are in control of the network, meaning they could double-spend coins.
- Blockchain may also get compromised because of a problem that does not even belong to the blockchain technology itself. Cryptocurrency exchange platforms have become lucrative honeypots for cybercriminals due to their massive crypto holdings and often the poor security practices applied to them. Many exchange platforms are innately centralized, so they render the decentralized benefits of blockchains useless. In order to combat this problem, it is proposed to store funds through either a hardware or paper wallet. These methods have minimal online fingerprints which keep the blockchain assets out of malicious online hackers' reach.
- Social engineering is also used by hackers to compromise the blockchain security. It comes in many forms, but the goal is always to either obtain the users' private keys and login information or more directly, their cryptocurrency. Phishing is one of the most common forms of social engineering. In a phishing attempt, a malicious actor sends an email, message, or even sets up a website or social media account imitating a trustworthy company brand. It is usually asked from the user to send credentials under the preteens of a giveaway or a critical issue to force a sense of urgency.
- Most of the popular blockchains (Bitcoin, Ethereum) have proven their resilience to all types of attacks. However, the apps built on top of them are vulnerable and bugs continue to surface. It's important that any software based on blockchain technology undergoes code reviews, penetration testing, and smart contract audits. Additionally, applications should have redundant security measures in place.
- Cryptojacking is the type of malware most associated with blockchain and cryptocurrency. Cryptojacking is the unauthorized and often unnoticeable takeover of a computer's resources to mine cryptocurrency. Although cryptojackers don't directly steal money from their victims, the malware they inject causes performance issues, increases electricity usage, and opens the door for other hostile code.

5.5 Smart Contracts Security Issues

As programs running in the blockchain, smart contracts may have security vulnerabilities caused by program defects. Nicola et al. [159] conduct a systematic investigation of 12 types of vulnerabilities in smart contracts, as shown in Table 2. Loi et al. [160] propose a symbolic execution tool called Oyente to find 4 kinds of potential security bugs. They discovered that 8833 out of 19,366 Ethereum smart contracts are vulnerable. The details of these 4 bugs are as follows[161]:

1. **Transaction-ordering dependence**: Valid transactions can change the state of Ethereum blockchain from σ to σ′: σ T→ σ′. In every epoch, each miner proposes its own block to update the blockchain. Since a block may contain multiple transactions, blockchain state σ may change multiple times within an epoch. When a new block contains two transactions Ti and Tj, which invoke the same smart

contract, it may trigger this vulnerability. Because the execution of the smart contract is associated with state σ, the execution order of Ti and Tj affects the ultimate state. The order of transactions' execution depends entirely on miners. In this case, TOD (Transaction-Ordering Dependent) contracts are vulnerable

2. **Timestamp dependence**: In the blockchain, every block has a timestamp. Some smart contracts' trigger conditions depend on timestamp, which is set by the miner according to its local system time. If an attacker can modify it, timestamp-dependent contracts are vulnerable.
3. **Mishandled exceptions**: This category of vulnerability may occur when different smart contracts are called from each other. When contract A calls contract B, if B runs abnormally, B will stop running and return false. In some invocations, contract A must explicitly check the return value to verify if the call has been executed properly. If A does not correctly check the exception information, it may be vulnerable.
4. **Re-entrancy vulnerability**: During the invocation of the smart contract, the actual state of the contract account is changed after the call is completed. An attacker can use the intermediate state to make repeated calls to the smart contract. If the invoked contract involves Ether transaction, it may result in illegal Ether stealing.

Table 2: Taxonomy of vulnerabilities in smart contract.

Vulnerability	Cause	Level
Call to the unknown	The called function does not exist	Contract source code
Out-of-gas send	Fallback of the callee is executed	
Exception disorder	Irregularity in exception handling	
Type casts	Type-check error in contract execution	
Re-entrancy vulnerability	The function is re-entered before termination	
Field disclosure	Private value is published by the miner	

Immutable bug	Alter a contract after deployment	EVM bytecode
Ether lost	Send ether to an orphan address	
Stack overflow	The number of values in stack exceeds 1024	
Unpredictable state	State of the contract is changed before invoking	Blockchain mechanism
Randomness bug	Seed is biased by malicious miner	
Timestamp dependence	Timestamp of block is changed by malicious miner	

5.6. TEE Assisted Blockchain Technologies & Research

5.6.1. TEE Assisted Blockchain

The Intel SGX solution, used with Blockchain, results in a very high level of trust for decentralized applications, as well as data processing on decentralized nodes. Blockchain technology combined with Intel SGX can protect the decentralized apps by fully securing the data that is not accessible by the execution host. This is essential for the user's input and output data. Also, blockchain-based validation for off-chain computing is provided in order to verify that the decentralized applications are correctly executed in isolation which is neither altered nor interrupted by a decentralized node. A smart-contract signature is signed inside this secure enclave before the verification is done by the blockchain network. Furthermore, assurance is provided that the execution and result is valid and not copied or fabricated by the malicious decentralized node. Also, the end-to-end privacy of the decentralized applications' result is protected, an aspect that can never be inspected by anyone else but the user. [162]

5.6.2. TEE Assisted Smart Contracts

Smart contracts inherit some undesirable blockchain properties. To enable validation of state transitions during consensus, blockchain data is public. In brief, the application complexity of smart contracts today is highly constrained. Without critical performance and confidentiality improvements, smart contracts may fail to deliver on their transformative promise. Researchers have explored cryptographic solutions to these challenges, such as various zero-knowledge proof systems and secure multiparty computation. However, these approaches have significant performance overhead and are only applicable to limited use cases with relatively simple computations. A more performant and general-purpose option is the use of a trusted execution environment (TEE).

A TEE provides a fully isolated environment that prevents other software applications, the operating system, and the host owner from tampering with or even learning the state of an application running in the TEE. For example, Intel Software Guard eXtensions (SGX) provides an implementation of a TEE.

To the best of our knowledge, [163] is the first confidentiality-preserving smart contract system capable of thousands of transactions per second. The key to this achievement is a secure and principled combination of blockchains and trusted hardware.

5.7. TPM Assisted Blockchain Technologies & Research

5.7.1 TPM Assisted Smart Contracts

Several approaches have recently suggested executing blockchain applications and smart contracts within Intel SGX. The most prominent among these is Microsoft's Coco framework [164]. It provides a set of building blocks based on Intel SGX that can be used to secure blockchain systems. Coco integrates consensus algorithms, distributed ledger state, and a runtime environment for executing smart-contract transactions in SGX enclaves. It appears to be derived from Ethereum and mentions the Ethereum Virtual Machine (EVM) for its core data structures and protocols. The components of Coco are described as potentially separate enclaves, but conceptually the entire blockchain node resides in SGX. Since there are limited references available in this area, however, it is difficult to clearly assess the framework [165].

The R3 Corda distributed ledger platform has also announced a privacy feature using SGX in a white paper [166]. In Corda, some aspects of the transaction validation are envisaged to take place inside an SGX enclave, potentially running at untrusted nodes. By executing the transaction validation in an enclave, the identities involved in a transaction can be encrypted on the blockchain and are only revealed inside an enclave during validation. Corda strives to port a Java runtime environment (JRE) in order to execute native Corda smart contracts within an enclave.

The IBM Blockchain Platform [167] offers an enterprise blockchain-as-a-service solution allowing for deployment of a Fabric network using Secure Service Container (SSC) technology [168] on IBM Z systems. The platform runs the whole peer and its Linux operating system within a secure container, which is shielded from access of the host and host administrator, similar to SGX. This means that running Fabric within a Secure Service Container requires specialized mainframe hardware and has a large TCB. In contrast, our approach for running Fabric with SGX works with commodity systems and minimizes the TCB for each smart-contract application.

Blockchain oracles and off-chain data. Other works [169][170] realize trusted "oracles" for blockchain smart contracts using SGX. Oracles are data feeds externally to the blockchain. They extend the scope of inputs to which an application can respond and serve as trusted sources and triggers for actions on the blockchain. Leveraging trusted execution technology enhances the trustworthiness of an oracle and allows to verify the correctness of the data source.

6 Conclusion

Trusted computing technologies, although relatively new, have matured in recent years and have managed to provide a secure and credible execution environment that is able to maintain the confidentiality, integrity, and availability of sensitive data and operations. This family of technologies aims to host within their context a safe and completely isolated (physically or virtually) environment on which specific components of an entire application are selected to run. This way, they can enhance the security of sensitive applications by running them in a safely separated from the untrusted device "world".

As evident, due to their detached nature, trusted computing technologies can be used in digital forensics since they cannot be easily manipulated by external actors. In the field of digital forensics, a core factor is the inability of a malicious entity to tamper with the forensic data they leave behind and with the forensic data collection process. If any of those two conditions fail, then the data collected cannot be used in a court of law due to the uncertainty created regarding the validity of the data. Trusted computing can be used to help a "safe zone" which can guarantee that the data collected are valid and they are gathered correctly and thoroughly.

In this deliverable we have presented and analysed the state of the art on trusted computing in the form of trusted execution environments and trusted platform modules. We have analysed their security applications in various context while also focusing on LOCARD by having subsections on secure evidence collection and a section on trusted-computing-backed blockchain and smart contracts. Through our analysis we have identified the shortcomings and strengths of these technologies which will enable LOCARD to properly utilize them and implement confidentiality, integrity and accountability enhancements within the envisioned LOCARD solution and use cases. It is important to underline here that even though trusted computing is a step towards security, it withholds security problems that could lead to vulnerabilities, any implementation should be properly configured in order to uphold the security guarantees that these technologies provide.

In the context of LOCARD, we aim to use trusted computing technologies and especially trusted execution environments to improve the security and correctness of each use case scenario. More specifically, this technology will allow for proper acquisition of evidence, according to international forensic standards, that will be admissible in courts in every jurisdiction across the world with the data contained in the seized digital evidence collected at the crime scene examined in a forensically sound manner.

7 References

[1]. Microsoft Technology Licensing, "Transaction Processing for Consortium Blockchain Network", 15/638213, June 2017.

[2]. Microsoft Technology Licensing, "Establishment of Consortium Blockchain Network", 15/638180, June 2017.

[3]. Dai, Weiqi, et al. "SBLWT: A Secure Blockchain Lightweight Wallet based on Trustzone." IEEE Access (2018).

[4]. Layered Security for Your Next SoC: https://www.arm.com/products/silicon-ip-security

[5]. J. Seibel, K. LaFlamme, F. Koschara, R. Schumak and J. Debate, "Trusted execution environment". US Patent US20170214530A1, 27 01 2016.

[6]. "ADVANCED TRUSTED ENVIRONMENT: OMTP TR1," OMTP Limited, 2009.

[7]. V. Galindo, "Poulpita," 18 2 2014. [Online]. Available: https://poulpita.com/2014/02/18/trusted-execution-environment-do-you-have-yours/. [Accessed 22 5 2018].

[8]. G. Arfaoui, S. Gharout and J. Traore, "Trusted Execution Environments: A Look under the Hood," in 2nd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud), Oxford, 2014.

[9]. "TEE System Architecture v1.2," GlobalPlatform, 2018.

[10]."TEE Client API Specification," GlobalPlatform, 2010.

[11].A. Machiry, E. Gustafson, C. Spensky, C. Salls, N. Stephens, R. Wang, A. Bianchi, Y. R. Choe, C. Kruegel and G. Vigna, "BOOMERANG: Exploiting the Semantic Gap in Trusted Execution Environments," in Proceedings of the Network and Distributed System Security Symposium, San Diego, 2017.

[12].ARM, "ARM Security Technology Building a Secure System using TrustZone® Technology," 2004. [Online]. Available: http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0301h/Chdebaee.html.

[13].ARM, "ARM Security Technology Building a Secure System using TrustZone® Technology," 2005. [Online]. Available: http://infocenter.arm.com/help/topic/com.arm.doc.prd29-genc-009492c/PRD29-GENC-009492C_trustzone_security_whitepaper.pdf.

[14].Arm, Layered Security for Your Next SoC.

[15].G. Beniamini, Full TrustZone exploit for MSM8974, 2015.

[16].G. Beniamini, QSEE privilege escalation vulnerability and exploit (CVE-2015-6639), 2016.

[17].G. Beniamini, TrustZone Kernel Privilege Escalation (CVE-2016-2431), 2016.

[18].Y. Chen, Y. Zhang, Z. Wang and T. Wei, "Downgrade attack on trustzone," arXiv preprint arXiv:1707.05082, 2017.

[19].M. Weiss, B. Heinz and F. Stumpf, "A cache timing attack on AES in virtualization environments," in International Conference on Financial Cryptography and Data Security, 2012.

[20].H. Cho, P. Zhang, D. Kim, J. Park, C.-H. Lee, Z. Zhao, A. Doupe and G.-J. Ahn, "Prime+Count: Novel Cross-world Covert Channels on ARM TrustZone," in Proceedings of the 34th Annual Computer Security Applications Conference, New York, ACM, 2018, pp. 441-452.

[21].M. Lipp, D. Gruss, R. Spreitzer, C. Maurice and S. Mangard, "ARMageddon: Cache attacks on mobile devices," in 25th {USENIX} Security Symposium ({USENIX} Security 16), 2016.

[22].N. Zhang, K. Sun, D. Shands, W. Lou and Y. T. Hou, "TruSpy: Cache Side-Channel Information Leakage from the Secure World on ARM Devices.," IACR Cryptology ePrint Archive, vol. 2016, p. 980, 2016.

[23].A. Tang, S. Sethumadhavan and S. Stolfo, "{CLKSCREW}: Exposing the Perils of Security-Oblivious Energy Management," in 26th {USENIX} Security Symposium ({USENIX} Security 17), 2017.

[24].A. Cui and R. Housley, "{BADFET}: Defeating Modern Secure Boot Using Second-Order Pulsed Electromagnetic Fault Injection," in 11th {USENIX} Workshop on Offensive Technologies ({WOOT} 17), 2017.

[25].INTEL. Product Change Notification. https://qdms.intel.com/dm/i.aspx/5A160770-FC47-47A0-BF8A-062540456F0A/PCN114074-00.pdf, October 2015.

[26].INTEL CORP. Software Guard Extensions Programming Reference, Ref. 329298-002US.https://software.intel.com/sites/default/files/managed/48/88/329298-002.pdf, 2014.

[27].COSTAN, V., AND DEVADAS, S. Intel SGX Explained. Tech. rep., Cryptology ePrint Archive, 2016

[28].V. Costan and S. Devadas. Intel SGX Explained. https://eprint.iacr.org/2016/086.pdf, 2016.

[29].Intel. ISCA 2015 SGX Tutorial. https://software.intel.com/sites/default/les/332680-002.pdf,2015.

[30].J. Rutkowska. Thoughts on Intel's upcoming SoftwareGuard Extensions (Part 1).http://blog.invisiblethings.org/ 2013/08/30/thoughts-on-intels-upcoming-software.html, August 2013.

[31].P. Jain, S. Desai, S. Kim, M.-W. Shih, J. Lee, C. Choi,Y. Shin, T. Kim, B. B. Kang, and D. Han. OpenSGX: An Open Platform for SGX Research. In Proceedings of the 2016 Annual Network and Distributed System Security Symposium (NDSS'16), San Diego, CA, Feb. 2016.

[32]. SoK: A Study of Using Hardware-assisted Isolated Execution Environments for Security, Fengwei Zhang, Hongwei Zhang, HASP 2016, June 18 2016

[33]. P. Jain, S. Desai, S. Kim, M.-W. Shih, J. Lee, C. Choi, Y. Shin, T. Kim, B. B. Kang, and D. Han. OpenSGX: An Open Platform for SGX Research. In Proceedings of the 2016 Annual Network and Distributed System Security Symposium (NDSS'16), San Diego, CA, Feb. 2016.

[34]. X. Ruan. Platform Embedded Security Technology Revealed: Safeguarding the Future of Computing with Intel Embedded Security and Management Engine. Apress, 2014.

[35]. Full security solutions that locks you down, not in: https://www.amd.com/en/technologies/security

[36]. Protect enterprise workloads with Secure Service Container: https://www.ibm.com/us-en/marketplace/secure-service-container

[37]. MultiZone: https://hex-five.com/

[38]. OP-TEE Overview: https://wiki.st.com/stm32mpu/wiki/OP-TEE_overview

[39]. GlobalPlatform: https://globalplatform.org/

[40]. OP-TEE GitHub: https://github.com/OP-TEE/

[41]. Open Portable Trusted Execution Environment: https://www.op-tee.org/

[42]. Paverd, Andrew J., and Andrew P. Martin. "Hardware security for device authentication in the smart grid." International Workshop on Smart Grid Security. Springer, Berlin, Heidelberg, 2012.

[43]. Bhargav-Spantzel, Abhilasha. "TRUSTED EXECUTION ENVIRONMENT FOR PRIVACY PRESERVING BIOMETRIC AUTHENTICATION." Intel Technology Journal 18.4 (2014).

[44]. Shepherd, Carlton, Raja Naeem Akram, and Konstantinos Markantonakis. "Towards trusted execution of multi-modal continuous authentication schemes." Proceedings of the Symposium on Applied Computing. ACM, 2017.

[45]. Marforio, Claudio, et al. "Secure enrollment and practical migration for mobile trusted execution environments." Proceedings of the Third ACM workshop on Security and privacy in smartphones & mobile devices. ACM, 2013.

[46]. McGillion, B., Dettenborn, T., Nyman, T., & Asokan, N. (2015, August). Open-TEE--An Open Virtual Trusted Execution Environment. In 2015 IEEE Trustcom/BigDataSE/ISPA (Vol. 1, pp. 400-407). IEEE.

[47]. FIDO UAF Authenticator Commands: https://fidoalliance.org/specs/fido-uaf-v1.1-id-20170202/fido-uaf-authnr-cmds-v1.1-id-20170202.html

[48]. FIDO UAF APDU: https://fidoalliance.org/specs/fido-uaf-v1.1-id-20170202/fido-uaf-apdu-v1.1-id-20170202.html

[49]. Y. Gicquel, “Secure boot Secure software update,” 2016. Online: https://iot.bzh/download/public/2016/publications/SecureBoot-SecureSoftwareUpdates.pdf

[50]. C. Mune, E. Sanfelix, “Secure Initialization of TEEs,” 2017. Online: https://www.riscure.com/uploads/2017/08/euskalhack_2017_-_secure_initialization_of_tees_when_secure_boot_falls_short.pdf

[51]. T. Ceresini, “Maintaining the forensic viability of logfiles,” SANS Institute, As part of GIAC practical repository, 5 2001.

[52]. P. D. Dixon, “An overview of computer forensics,” Potentials, IEEE, vol. 24, no. 5, pp. 7–10, Dec. 2005.

[53]. T. Ceresini, “Maintaining the forensic viability of logfiles,” SANS Institute, As part of GIAC practical repository, 5 2001.

[54]. Anderson, R.: Cryptography and competition policy: issues with \trusted computing". In: Proceedings of the twenty-second annual symposium on Principles of distributed computing, ACM Press (2003)

[55]. TCG: Trusted platform module summary. Technical report, Trusted Computing Group (2008)

[56]. 00.96, March 15 (2013) TPM Design Principles, https://trustedcomputinggroup.org/wp-content/uploads/TPM-Main-Part-1-Design-Principles_v1.2_rev116_01032011.pdf

[57]. M. Eriksson, M. Pourzandi, B. Smeets, “Trusted computing for infrastructure”, Ericsson Review, October 24, 2014, 2-8.

[58]. Tomilson, A.: Introduction to the tpm. In: Smart Cards, Tokens, Security and Applications. Springer (2008) 155-172

[59]. TCG: TCG Specification Architecture Overview Revision 1.4. Trusted Computing Group, Portland, OR. (August 2007)

[60]. A. Carroll,M., Juarez, I., Polk, T., Leininger, , Microsoft "Palladium": A business overview. Technical report, Microsoft Content Security Business Unit (August 2002)

[61]. Ultimaco Safeware AG. “SGCE Cryptographic Library, FIPS 140-2 Level 1 Vendor Evidence Documentation”, Version 1.10, August 2010.

[62]. Thales HSM, https://www.thalesesecurity.com/products/hsm-management-and-monitoring

[63]. R. Sailer, X. Zhang, T. Jaeger, L. van Doorn.. Design and implementation of a TCG-based Integrity Measurement Architecture. In SSYM’04: Proceedings of the 13th conference on USENIX Security Symposium, USENIX Association: Berkeley, CA, USA; 2004.

[64]. T. Jaeger, R., Sailer, U. Shankar. PRIMA: Policy-reduced Integrity Measurement Architecture. In SACMAT ’06: Proceedings of the eleventh ACM Symposium on Access Control Models and Technologies. ACM Press: New York, NY, USA, 2006; 19–28.

[65]. A. R. Sadeghi, C. Stuble. Property-based attestation for computing platforms: caring about properties, not mechanisms. In NSPW ’04: Proceedings of the 2004 Workshop on New Security Paradigms, ACM Press: New York, NY, USA, 2004; 67–77.

[66]. K. Kostiainen, N. Asokan, J. E.. Ekberg: Practical property-based attestation on mobile devices. In: McCune, J.M., Balacheff, B., Perrig, A., Sadeghi, A.R., Sasse, Y. Beres. (eds.) Trust 2011. LNCS, vol. 6740, pp. 78–92. Springer, Heidelberg (2011)

[67]. V. Haldar, D. Chandra, M. Franz. Semantic Remote Attestation—a virtual machine directed approach to Trusted Computing. In. Proc. of the Third Virtual Macine Research and Technology Symposium USENIX 2004.

[68]. L. Gu, X. Ding, R. Deng, B. Xie, H. Mei. Remote Attestation on Program Execution. In STC ’08: Proceedings of the 2008 ACM Workshop on Scalable Trusted Computing, ACM: New York, NY, USA, 2008.

[69]. J. Lyle. 2009. Trustable Remote Verification of Web Services. In Proceedings of the 2nd International Conference on Trusted Computing (Trust '09), Liqun Chen, Chris J. Mitchell, and Andrew Martin (Eds.). Springer-Verlag, Berlin, Heidelberg, 153-168. DOI=http://dx.doi.org/10.1007/978-3-642-00587-9_10

[70]. M. Alam, X. Zhang, M. Nauman, T. Ali, J. P. Seifert. Model-based Behavioral Attestation. In SACMAT ’08: Proceedings of the thirteenth ACM symposium on Access control models and technologies ACM Press: New York, NY, USA, 2008.

[71]. S. Berger, R. Cáceres, K. A. Goldman, R. Perez, R. Sailer, and L. van Doorn. 2006. vTPM: virtualizing the trusted platform module. In Proceedings of the 15th conference on USENIX Security Symposium - Volume 15 (USENIX-SS'06), Vol. 15. USENIX Association, Berkeley, CA, USA, pages.

[72]. England, P., Loeser, J.: Para-Virtualized TPM Sharing. In: Lipp, P., Sadeghi, A.-R., Koch, K.-M. (eds.) TRUST 2008. LNCS, vol. 4968, pp. 119–132. Springer, Heidelberg (2008)

[73]. Stumpf, F., Eckert, C.: Enhancing Trusted Platform Modules with Hardware-Based Virtualization Techniques. In: Cotton, A., Dini, O., Skarmeta, A.F.G., Ion, M., Popescu, M., Takesue, M. (eds.) Proceedings of the Second International Conference on Emerging Security Information, Systems and Technologies, SECURWARE 2008, pp. 1–9. IEEE Computer Society (2008)

[74]. Chen, L., Ryan, M.: Offline dictionary attack on TCG TPM weak authorisation data, and solution. In: Future of Trust in Computing. Vieweg+Teubner (2009) 193-196

[75]. Chen, L., Ryan, M.: Attack, solution and veri_cation for shared authorisation data in TCG TPM. In: Proceedings of the 6th international conference on Formal Aspects in Security and Trust. FAST '09 (2010) 201-216

[76]. Blanchet, B.: An efficient cryptographic protocol verifier based on prolog rules. In: Proceedings of the 14th IEEE workshop on Computer Security Foundations. CSFW '01 (2001)

[77]. Kinkelin, Holger, et al. "On using tpm for secure identities in future home networks." *Future Internet* 3.1 (2011): 1-13.

[78]. FIDO UAF Authenticator Commands: https://fidoalliance.org/specs/fido-uaf-v1.1-id-20170202/fido-uaf-authnr-cmds-v1.1-id-20170202.html

[79]. FIDO ECDAA Algorithm: https://fidoalliance.org/specs/fido-uaf-v1.1-id-20170202/fido-ecdaa-algorithm-v1.1-id-20170202.html

[80]. FIDO UAF Protocol Specification: https://fidoalliance.org/specs/fido-uaf-v1.1-id-20170202/fido-uaf-protocol-v1.1-id-20170202.html

[81]. UEFI and the TPM: Building a foundation for platform trust: https://resources.infosecinstitute.com/uefi-and-tpm/

[82]. Spafford, E.: Some Challenges in Digital Forensics. In: Advances in Digital Forensics II (2006)

[83]. Adams, C.W.: Legal Issues Pertaining to the Development of Digital Forensic Tools. In: 3rd International Workshop on Systematic Approaches to Digital Forensic Engineering, Oakland, California, USA (2008)

[84]. Caloyannides, M.A.: Forensics Is So "Yesterday". IEEE Security & Privacy 7(2), 18–25 (2009)

[85]. Carrier, B.D.: Digital Forensics Works. Computing in Science and Engineering 7(2), 26– (2009)

[86]. Liles,S.,Rogers,M.,et al.:Survey of the Legal Issues Facing Digital Forensic Experts.In: Advances in Digital Forensics V (2009)

[87]. Antoniou, G., Wilson, C., Geneiatakis, D.: PPINA - A Forensic Investigation Protocol for Privacy Enhancing Technologies. In: Leitold, H., Markatos, E.P. (eds.) CMS 2006. LNCS, vol. 4237, pp. 185–195. Springer, Heidelberg (2006)

[88]. Antoniou, G., Gritzalis, S.: RPINA- Network Forensics Protocol Embedding Privacy Enhancing Technologies. In: IEEE International Symposium on Communications and Information Technology (ISCIT), Bangkok, Thailand (2006)

[89]. Antoniou, G., Sterling, L., et al.: Privacy and forensics investigation process: The ERPINA protocol. Computer Standards & Interfaces 30, 229–236 (2008)

[90]. Olivier, M.: Forensics and Privacy-Enhancing Technologies. In: Advances in Digital Forensic, ch. 2 (2008)

[91]. Muththolib Sidheeq, Ali Dehghantanha, Geetha Kananparan, ―Utilizing Trusted Platform Module to Mitigate Botnet Attacks‖ in International Journal of Advancements in Computing Technology (IJACT), Volume 2 Issue 5, pp. 111-117, 2010- Korea

[92]. A Framework of TPM, SVM and Boot Control for Securing Forensic Logs, Nazanin Borhan, Ramlan Mahmod, Ali Dehghantanha

[93]. X. Zhao, K. Borders, and A. Prakash. Svgrid: A secure virtual environment for untrusted grid applications. In CM/IFIP/USENIX 6th International Middleware Conference, France. 2005

[94]. Intel. 64 and IA-32 Architectures Software Developer's Manual. http://www.intel.com/content/www/us/en/processors/architectures-software-developer-manuals.html.

[95]. SoK: A Study of Using Hardware-assisted Isolated Execution Environments for Security/ Fengwei Zhang, HASP 2016, June 18 2016

[96]. Trusted Computing Group. TCG D-RTM Architecture Document Version 1.0.0. http://www.trustedcomputinggroup.org/resources/drtmarchitecture specication, June 2013.

[97]. Trusted Computing Group. TCG PC Client Specific Implementation Specification For Conventional BIOS, Version 1.20, Revision 1.00, For TPM Family 1.2.

[98]. Intel. Trusted Execution Technology. http://www.intel.com/content/www/us/en/trusted-execution-technology/trusted-execution-technology-security-paper.html.

[99]. Advanced Micro Devices, Inc. AMD64 ArchitectureProgrammer's Manual Volume 2: SystemProgramming. http://support.amd.com/TechDocs/24593.pdf, June2015.

[100]. J. McCune, B. Parno, A. Perrig, M. Reiter, and H. Isozaki. Flicker: An Execution Infrastructure for TCB Minimization. In Proceedings of the 3rd ACM SIGOPS/EuroSys European Conference on Computer Systems, 2008.

[101]. J. M. McCune, Y. Li, N. Qu, Z. Zhou, A. Datta, V. Gligor, and A. Perrig. TrustVisor: Ecient TCB reduction and attestation. In Proceedings of the 31st IEEE Symposium on Security and Privacy, 2010.

[102]. J. M. McCune, A. Perrig, and M. K. Reiter. Safe passage for passwords and other sensitive data. In NDSS, 2009.

[103]. SoK: A Study of Using Hardware-assisted Isolated Execution Environments for Security/ Fengwei Zhang

[104]. https://www.blockchain.com/

[105]. G. Zyskind, O. Nathan and A. '. Pentland, "Decentralizing Privacy: Using Blockchain to Protect Personal Data," 2015 IEEE Security and Privacy Workshops, San Jose, CA, 2015, pp. 180-184

[106]. A Semantic Framework for the Security Analysis of Ethereum smart contracts, Ilya Grishchenko, Matteo Maffei, and Clara Schneidewind

[107]. An Empirical Analysis of Smart Contracts: Platforms, Applications, and Design Patterns, Massimo Bartoletti, Livio Pompianu, International Financial Cryptography Association 2017

[108]. Colchester, J. (2018) Blockchain: An overview. Available at http://complex-itylabs.io/blockchain-overview/

[109]. Bitcoin: A Peer-to-Peer Electronic Cash System, Satoshi Nakamoto, 2008

[110]. Crosby, M., Pattanayak, P., Verma, S., Kalyanaraman, V., 2016. Blockchain technology: beyond bitcoin. Appl. Innovation 2, 6–10.

[111]. Zheng, Z., Xie, S., Dai, H.-N., Wang, H., 2016. Blockchain challenges and opportunities: a survey. Work Pap

[112]. Haferkorn, M., Quintana Diaz, J.M., 2015. Seasonality and Interconnectivity Within Cryptocurrencies – An Analysis on the Basis of Bitcoin, Litecoin and Namecoin. Springer International Publishing, Cham (pp. 106–120).

[113]. Nguyen, Q.K., 2016. Blockchain-A Financial Technology for Future Sustainable Development. In: Proceedings – 3rd International Conference on Green Technology and Sustainable Development, GTSD 2016, pp. 51–54.

[114]. Van de Velde, J., Scott, A., Sartorius, K., Dalton, I., Shepherd, B., Allchin, C., Dougherty, M., Ryan, P., Rennick, E., 2016. Blockchain in capital markets—The prize and the journey.

[115]. Wu, T., Liang, X., 2017. Exploration and practice of inter-bank application based on blockchain. In: ICCSE 2017–12th International Conference on Computer Science and Educationpp. 219–224.

[116]. Papadopoulos, G., 2015. Blockchain and Digital Payments: An Institutionalist Analysis of Cryptocurrencies, 153–172.

[117]. Beck, R., Stenum Czepluch, J., Lollike, N., Malone, S., 2016. Blockchain – The gateway to trust-free cryptographic transactions. In: 24th European Conference on Information Systems, ECIS 2016.

[118]. Gazali, H.M., Hassan, R., Nor, R.M., Rahman, H.M.M., 2017. Re-inventing PTPTN study loan with blockchain and smart contracts. In: ICIT 2017–8th International Conference on Information Technology, Proceedings 751–754.

[119]. Cocco, L., Pinna, A., Marchesi, M., 2017. Banking on blockchain: costs savings thanks to the blockchain technology. Future Internet 9 (3), 25.

[120]. Dai, J., Vasarhelyi, M.A., 2017. Toward blockchain-based accounting and assurance. J. Inf. Syst. 31 (3), 5–21.

[121]. Cawrey, D., 2014. 37Coins Plans Worldwide Bitcoin Access with SMS-Based Wallet, http://www.coindesk.com/37coins-plans-worldwide-bitcoin-access-sms-basedwallet/.

[122]. Rizzo, P., 2014. How Kipochi Is Taking Bitcoin into Africa, http://www.coindesk.com/kipochi-taking-bitcoin-africa/.

[123]. Bhowmik, D., Feng, T., 2017. The multimedia blockchain: a distributed and tamper-proof media transaction framework. In: International Conference on Digital Signal Processing DSP.

[124]. Dupont, Q., 2017. Blockchain identities: notational technologies for control and management of abstracted entities. Metaphilosophy 48 (5), 634–653.

[125]. Reijers, W., O'Brolcháin, F., Haynes, P., 2016. Governance in blockchain technologies & social contract theories. Ledger 1, 134–151.

[126]. S., Pinna, A., Seu, M., Pani, F.E., 2017. CitySense: Blockchain-oriented Smart Cities. In: ACM International Conference Proceeding Series, vol. Part F129907

[127]. Lee, J.-H., 2018. BIDaaS: blockchain based ID as a service. IEEE Access 6, 2274–2278.

[128]. Lemieux, V.L., 2016. Trusting records: is blockchain technology the answer? Records Manage. J. 26 (2), 110–139.

[129]. Roberts, J.J., Microsoft and Accenture Unveil Global ID System for Refugees, http://fortune.com/2017/06/19/id2020-blockchain-microsoft/.

[130]. Boucher, P., 2016. What if blockchain technology revolutionised voting? Scientific Foresight Unit (STOA), European Parliamentary Research Service.

[131]. Moura, T., Gomes, A., 2017. Blockchain Voting and its effects on Election Transparency and Voter Confidence. In: Proceedings of the 18th Annual International Conference on Digital Government Research, ACM, pp. 574–575.

[132]. IBM, 2017. Three ways blockchain Explorers chart a new direction,https://www-935.ibm.com/services/studies/csuite/pdf/GBE03835USEN-00.pdf.

[133]. Stats, I.W., 2017. Internet usage statistics, The internet big picture, http://www.internetworldstats.com/stats.htm.

[134]. R.D., Guarino, S., Verde, N., Domingo-Ferrer, J., 2014. Security in wireless ad-hoc networks – A survey. Comput. Commun. 51 (Supplement C), 1–20.

[135]. Novo, O., 2018. Blockchain meets IoT: an architecture for scalable access management in IoT. IEEE Internet Things J. 5 (2), 1184–1195.

[136]. Mettler, M., 2016. Blockchain technology in healthcare: The revolution starts here. In: 2016 IEEE 18th International Conference on e-Health Networking, Applications and Services (Healthcom) IEEE, pp. 1–3.

[137]. Liu, P.T.S., 2016. Medical record system using blockchain, big data and tokenization. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 9977 LNCS, pp. 254–261.

[138]. Angraal, S., Krumholz, H.M., Schulz, W.L., 2017. Blockchain technology: Applications in health care, Circulation: Cardiovascular Quality and Outcomes 10(9).

[139]. Hoy, M.B., 2017. An introduction to the blockchain and its implications for libraries and medicine. Med. Ref. Services Q. 36 (3), 273–279.

[140]. Azaria, A., Ekblaw, A., Vieira, T., Lippman, A., 2016. MedRec: Using blockchain for medical data access and permission management. In: Proceedings – 2016 2nd International Conference on Open and Big Data, OBD 2016, pp. 25–30.

[141]. Sullivan, Frost, 2017. Future of Blockchain Technology in Connected Health Ecosystem. Convergence of Blockchain with AI and IoMT set to re-architect Healthcaresector, http://innovationsprint.eu/wp-content/uploads/2017/08/Briefing-Deck-Future-of-Blockchain-Technology-_1 h-Sept.pdf.

[142]. Parliament, E., 2016. General Data Protection Regulation, http://data.consilium.europa.eu/doc/document/ST-5419-2016-INIT/en/pdf.

[143]. Politou, E., Alepis, E., Patsakis, C., 2018. Forgetting personal data and revoking consent under the GDPR: challenges and proposed solutions. J. Cybersecurity 1–20.

[144]. Puthal, D., Malik, N., Mohanty, S.P., Kougianos, E., Yang, C., 2018. The blockchain as a decentralized security framework [Future Directions]. IEEE Consumer Electron. Magazine 7 (2), 18–21.

[145]. Bozic, N., Pujolle, G., Secci, S., 2016. A tutorial on blockchain and applications to secure network control-planes. In: 3rd Smart Cloud Networks and Systems, SCNS 2016.

[146]. Di Francesco Maesa, D., Mori, P., Ricci, L., 2017. Blockchain based access control. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 10320 LNCS, pp. 206–220.

[147]. Tang, C.M., Zhang, Y.L., Yu, X., 2018. Design of vehicle networking data exchange system based on BlockChain. J. Tianjin Polytech. Univ. 37 (2), 84–88.

[148]. Dorri, A., Steger, M., Kanhere, S.S., Jurdak, R., 2017c. BlockChain: a distributed solution to automotive security and privacy. IEEE Commun. Mag. 55 (12), 119–125.

[149]. Bilal, K., Malik, S.U.R., Khalid, O., Hameed, A., Alvarez, E., Wijaysekara, V., Irfan, R., Shrestha, S., Dwivedy, D., Ali, M., Shahid Khan, U., Abbas, A., Jalil, N., Khan,S.U., 2014. A taxonomy and survey on green data center networks. Future Generation Comput. Syst. 36, 189–208.

[150]. Mengelkamp, E., Notheisen, B., Beer, C., Dauer, D., Weinhardt, C., 2018. A blockchain-based smart grid: towards sustainable local energy markets. Comput. Sci.-Res.Dev. 33 (1–2), 207–214.

[151]. Wu, L., Meng, K., Xu, S., Li, S.Q., Ding, M., Suo, Y., 2017. In: Democratic Centralism: A Hybrid Blockchain Architecture and Its Applications in Energy Internet, in:Proceedings – 1st IEEE International Conference on Energy Internet 2017, pp. 176–181.

[152]. Deutsche Energie-Agentur GmbH, Blockchain in the energy transition. A survey among decision-makers in the German energy industry.https://www.dena.de/ fileadmin/dena/Dokumente/Meldungen/dena_ESMT_Studie_blockchain_englisch.pdf, 2016.

[153]. Kyriakarakos, G., Papadakis, G., 2018. Microgrids for productive uses of energy in the developing world and blockchain: a promising future. Appl. Sci. (Switzerland)8, (4).

[154]. Smart Contracts Application in Identity Security: https://dzone.com/articles/smart-contracts-application-in-identity-security

[155]. Fotiou, Nikos, and George C. Polyzos. "Smart contracts for the internet of things: Opportunities and challenges." 2018 European Conference on Networks and Communications (EuCNC). IEEE, 2018.

[156]. H. Ritzdorf, K. Wst, A. Gervais, G. Felley, and S. Capkun, “TLS-N:Non-repudiation over TLS Enabling - Ubiquitous Content Signingfor Disintermediation,” Cryptology ePrint Archive, Report 2017/578,2017. [Online]. Available: https://eprint.iacr.org/2017/578

[157]. Park, J.H.; Park, J.H. Blockchain Security in Cloud Computing: Use Cases, Challenges, and Solutions. Symmetry 2017

[158]. Lin, I.-C & Liao, T.-C. (2017). A survey of blockchain security issues and challenges. International Journal of Network Security.

[159]. A survey of attacks on Ethereum smart contracts, Nicola Atzei, Massimo Bartoletti, and Tiziana Cimoli, Proceedings of the 6th International Conference on Principles of Security and Trust - Volume 10204 Pages 164-186

[160]. L. Luu, D.-H. Chu, H. Olickel, P. Saxena, A. Hobor, Making smart contracts smarter, in: The ACM SIGSAC Conference on Computer and Communications Security, 2016, pp. 254-269.

[161]. A survey on the security of blockchain systems, Xiaoqi Li a, Peng Jiang, Ting Chen, Xiapu Luo , Qiaoyan Wenc

[162]. Intel Software Guard Extensions: https://software.intel.com/en-us/sgx

[163]. Ekiden: A Platform for Confidentiality-Preserving, Trustworthy, and Performant Smart Contracts,Raymond Cheng, Fan Zhang, Jernej Kos, Warren He, Nicholas Hynes∗§ Noah Johnson, Ari Juels† Andrew Miller, Dawn Song

[164]. Microsoft. 2017. The Coco Framework. (2017). Whitepaper, https://github.com/ Azure/coco-framework.

[165]. Blockchain and Trusted Computing: Problems, Pitfalls, and a Solution for Hyperledger Fabric, Marcus Brandenburger, Christian Cachin, Rüdiger Kapitza, Alessandro Sorniotti

[166]. Mike Hearn. 2016. Corda: A distributed ledger. (2016). Whitepaper,https://docs.corda.net/_static/corda-technical-whitepaper.pdf.

[167]. IBM. 2018. IBM Blockchain Platform. (2018).https://console.bluemix.net/docs/services/blockchain/index.html.

[168]. IBM. 2017. Secure Service Container User's Guide SC28-6971-01.(2017).https://www-01.ibm.com/support/docview.wss?uid=isg2bb79df265313634d85258088005188e3&aid=1.

[169]. Microsoft. 2017. Introducing Project "Bletchley". (2017). Whitepaper,https://github.com/Azure/azure-blockchain-projects/blob/master/bletchley/bletchley-whitepaper.md.

[170]. Fan Zhang, Ethan Cecchetti, Kyle Croman, Ari Juels, and Elaine Shi. 2016. Town Crier: An Authenticated Data Feed for Smart Contracts. In Proc. ACM SIGSAC Conference on Computer and Communications Security (CCS).

[171]. Abera, Tigist, et al. "C-FLAT: control-flow attestation for embedded systems software." *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016.

[172]. Dessouky, Ghada, et al. "Lo-fat: Low-overhead control flow attestation in hardware." *Proceedings of the 54th Annual Design Automation Conference 2017*. ACM, 2017.

[173]. Dessouky, Ghada, et al. "Litehax: Lightweight hardware-assisted attestation of program execution." *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2018.

[174]. Ambrosin, Moreno, et al. "SANA: secure and scalable aggregate network attestation." *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016.

[175]. Asokan, Nadarajah, et al. "Seda: Scalable embedded device attestation." *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2015.

[176]. Ibrahim, Ahmad, et al. "DARPA: Device attestation resilient to physical attacks." *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*. ACM, 2016.

[177]. Koutroumpouchos, Nikos, et al. "Secure Edge Computing with Lightweight Control-Flow Property-based Attestation." *2019 IEEE Conference on Network Softwarization (NetSoft)*. IEEE, 2019.

[178]. ARM, " The AMBA3 AXI system bus," 2009. [Online]. Available: http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.prd29-genc-009492c/ch03s02s01.html

[179]. ARM, " The AMBA3 APB peripheral bus," 2009. [Online]. Available: http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.prd29-genc-009492c/ch03s02s01.html

[180]. Fran Casino, Thomas K. Dasaklis, Constantinos Patsakis, A systematic literature review of blockchain-based applications: Current status, classification and open issues, Telematics and Informatics, Volume 36, 2019, Pages 55-81, ISSN 0736-5853, https://doi.org/10.1016/j.tele.2018.11.006.