TRESCCA

SEVENTH FRAMEWORK PROGRAMME

Project acronym:	**TRESCCA**
Project title:	TRustworthy Embedded systems for Secure Cloud Computing
Project number:	European Commission – 318036
Call identifier:	FP7-ICT-2011.1.4
Start date of project:	01 Oct. 2012 **Duration:** 36 months

Document reference number:	**D2.1**
Document title:	Analysis of state-of-the-art hardware security architectures and their weaknesses
Version:	1.1
Due date of document:	31st of March 2013
Submission date:	5th of May 2014
Lead beneficiary:	CS
Participants:	Andreas HERRHOLZ (CS), Christian STEHNO (CS), Guillaume DUC (IMT), Bernhard KATZMARSKI (OFFIS)
Reviewer:	Henning KLEEN (CS)

Project co-funded by the European Commission within the 7th Framework Programme		
DISSEMINATION LEVEL		
PU	Public	**X**
PCA	Public with confidential annex	
CO	Confidential, only for members of the consortium (including Commission Services)	

EXECUTIVE SUMMARY

This report reviews and analyses the current state-of-the-art of HW security solutions and how they can be used to protect against the security threats addressed by TRESCCA.

Current developments in IT security show that SW security measures are not sufficient to protect against common threats. Even more, successful attacks against typical devices such as mobile phones, tablet PCs, gaming consoles show that hacks can be executed in reasonable time, with low cost equipment and without special knowledge of the internals of a system.

There are several IT security standards and organisations like the ISO/IEC SC27 and Common Criteria which address security management, control and evaluation but which do not yet provide specific guidelines or guidelines for HW security measures (except for tamper proof requirements). Organisations like the Trusted Computing Group and GlobalPlatform and companies like Intel and ARM try to fill this gap by defining proprietary and de-facto standards for HW security architectures and SW APIs. However, their potential is still limited due to lack of transparency and openness.

In the context of cloud computing, there are several aspects to securing the client device and its data. The most difficult issue is to protect the client's data while it is being processed in the cloud. In general all data could be encrypted before it is being transferred to the cloud. However except for homomorphic encryption, no computation could be performed on such data. Due to its complexity and infancy homomorphic encryption is not yet practicable. An alternative for establishing trust between cloud and client is to use Trusted Execution Environments (TEE) on either of both sides.

There are several approaches for TEE, such as TPM, ARM TrustZone and Secure Elements. Though some of these are still vulnerable to external HW attacks, in particular simple memory bus attacks using DMA or bus sniffing. Such attacks can be prevented by encrypting sensitive parts of the communication with the external memory. While there are some solutions available for this already neither of them is mature or fast enough to be suitable for applications envisioned in TRESCCA. Therefore, The Consortium will take these solutions as base for developing an improved and more flexible memory encryption mechanism that is compatible with a wide range of SoC architectures and provides a good and configurable balance between performance and security.

CONTENTS

1 INTRODUCTION

This document is deliverable D2.1 of the TRESCCA project. It briefly reviews existing HW security measures and their weaknesses deriving initial requirements and objectives for the development of the HW security measures of the TRESCCA client platform.

Today, security is one of the most important topics in information technology as it is the main requirements for establishing trustworthiness of any service provided by or through IT devices such as mobile phones, personal computers and or entertainment devices.

Security of IT devices and their services is typically realized through a combination of HW and SW measures. While these measures do not directly depend on each other and can be applied independently they cover different points of attack and breaking one of them can severely impact the security of another. In that sense, the measures together establish a *Chain of Trust* where each of them relies on the dependability of a lower-level measure. HW security measures typically cover the lower levels of such Chain of Trust preventing any attacks to the HW components of a system which, if successful, may open up new points of attack of the SW running on top of it.

This report starts with an overview on typical applications and devices that either incorporate some type of HW security already or would significantly benefit from such measures. In the following chapter existing ICT security standards and standardization organizations are reviewed with respect to their contributions or relevance to HW security. Chapter 4 reviews current solutions to protect the client data either in the cloud or locally. Besides of standard encryption mechanisms solutions for protecting the execution of applications and the processing of data through so-called Trusted Execution Environments are described and analysed. As most of these TEE are still vulnerable HW attacks it is shown what countermeasures would be available.

1.1 Document Versions Sheet

Version	Date	Description, modifications, authors
1.0	2013-11-13	Initial version for Technical Review. A. HERRHOLZ (CS), C.STEHNO (CS), G. DUC (IMT), B. KATZMARSKI (OFFIS)
1.1	2014-05-05	Fix spelling and language. Add comparison overview of HW security solutions following review recommendations. A. HERRHOLZ (CS)

2 APPLICATIONS OF HW SECURITY

There are a wide range of application areas for computing devices with different types of security requirements. The level of security and the effort spend for largely depends on its targeted application area. We will therefore briefly review different application scenarios for computing devices listing their specific security threats and reviewing the currently available and deployed counter measures.

2.1 Personal Computing

Personal Computers (PC) or similar are one of the most common type of computing devices today. According to estimations of the ITU there will be about 410 Million PCs sold alone in 2013 and according to Eurostat 78% of EU27 households had at least one PC in their homes in 2012.

While the use of smart phones for accessing private data like e-mails, personal documents and social networks is becoming more common, PCs are still the most widely used device to store and access such data.This behaviour also makes it the most attractive point of attack. Furthermore PCs are the most common access device to cloud or web based services.

However, the level of security and trust of currently used PCs is almost neglectable. While there have been some efforts in previous years to establish HW and SW mechanisms to improve security and trust between HW and SW providers and users, their success was mostly limited to its use in some closed enterprise environments.

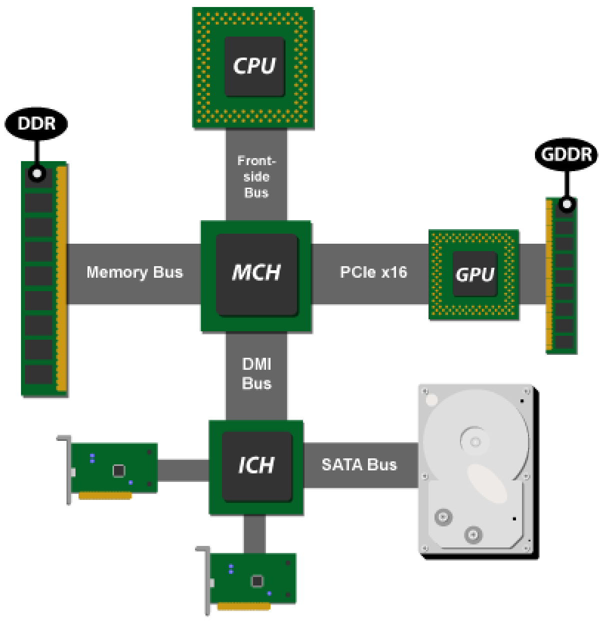

Figure 2.1: PC architecture with CPU, external buses and peripheral components

A typical PC architecture (see Figure 2.1) consists of an ATX compatible mainboard with an Intel x86 compatible CPU, a PCI or PCIexpress peripheral bus and some external DRAM. There are several options for attacking any SW running on such a system. For example, users are able to run any kind of application in parallel to any other application. Typically there are no checks, whether such applications can be trusted or not. While process isolation techniques in modern OS kernels provide direct access to the memory space of other applications, such isolation is only realized in SW (with minor HW assistance). Such mechanisms can easily be compromised if the attacker has access to a privileged account on the system allowing him to circumvent such protection, e.g. through the use of a compromised driver or kernel mode debugger (e.g. SoftICE). Similarly any exploits in SW programs can be used easily to run any untrusted code with highest privileges.

Alternatively attackers can also use HW attacks to compromise PC systems. One common approach is to use Direct Memory Access via the or PCI-Express bus to directly access any memory location of the system. This can be done by either modifying standard PC peripheral cards such as graphics adapter or network cards, or by developing custom cards (e.g. using FPGAs). The overall effort for such attacks is very low, requires only medium expert knowledge and material costs are moderately low.

In a similar HW attack an attacker attaches a sniffing device to the external bus signals (memory bus or PCI bus) on the printed circuit board enabling him to monitor all traffic between the CPU and the memory or other peripheral devices. Even more, depending on the speed and complexity of the bus, it is also possible to inject and manipulate and data going across the bus.

2.2 Digital payment

Digital payment, or synonymously electronic payment, is usually referring to any cash-less payment system which can be used for payment or similar financial transaction within an electronic shop system. Thus, even credit cards are nowadays part of the digital payment system and were even among the first digital payment systems used.

These classical payment cards, besides credit cards also debit cards and pre-paid cards, are commonly used to identify the owing partner of the financial transaction against the delivering one. Thus, they are strongly related to identification systems presented in the next subsection. The rest of the transaction is generally handled within the transaction system issued by the bank. Credit cards can also be issued as virtual credit cards. These are simply some credit cards without the physical card medium. Thus, they are usually not used for standard payment outside e-commerce systems due to a higher risk of fraud. The transactions are still related to some kind of bank account and thus are rather similar to the typical credit card system. The lack of physical items makes this system unrelated to any hardware security measures. Examples are the Wirecard [30] or the paysafecard [23], the latter featuring anonymous payment, but limited to a one-time pre-paid card.

Figure 2.2: Smart card for digital payment (debit card) - (c) Christian Horvat (CC Attribution Share 3.0)

Digital cash extends the classical money system with a virtual representation of money itself. If the virtual representation is lost, so is the amount of money. According to the EC definition of e-money, the digital representation has to be stored on some physical device, is issued by some institution by exchanging money into e-money, and needs to be accepted by other institutions than the issuing one. Thus, typical coupons or other more classical types of cash replacements are not automatically e-cash, either due to the lack of electronic representation in the first place, or by lack of broad acceptance by other companies. E-money systems have become quite popular due to their ability to support micro payments, i.e. minimal amounts of money where account based systems are often inferior due to transaction fees that often have a minimal fixed amount or higher percentages for small charges.

Some examples of card-based e-money are the GeldKarte in Germany and Proton in Belgium. The money is transferred to some smart-card (cf. section 2.2.1) and is protected by the encryption mechanism of the card. Software-based e-money stores the e-money in common data storage, such as the hard drive of a PC. It is most often used for micro payments on websites and internet shops, but also to enhance the privacy level of payments and to avoid abuse of credit card data. The most prominent system that used the direct storage of e-money on PCs was eCash by DigiCash, Inc. This system was shut down in the late 1990s. Some other systems are still active, but have not yet been too successful and widely used. This has changed to some extent with the development of Bitcoin, which added the decentralized acceptance control to software-based e-money. While former systems stored only the money (coins) decentralized on the user's storage, every transaction had to be checked by the central transaction server of the issuing institution. With Bitcoin and other crypto currencies, this test of authenticity can be performed without any central database. Even though Bitcoin and its descendants have established a significant money system, the software-based e-money systems are still not widely adopted.

Software-based e-money systems today usually use a centralized transaction system with dedicated, usually proprietary, interfaces for purchasing or money transfers. No storage on the client computer is needed, since all handling happens at the central institution's systems as some kind of webservice. PayPal is a well-known service of this kind. Due to the centralized software approach, these systems are also not relevant for hardware security considerations.

Due to the massive growth of smart phone usage, the software-based e-money systems have become popular again for mobile payment systems. While possible in principle, none of the existing systems stores the virtual coins on the phone. Thus, they require a connection to the central institution's server and only simplify access to the webservice. None of the services uses any dedicated means of hardware security.

Relevant for the discussion of hardware security are card or otherwise token-based systems, or the security means for the point of sale and communication towards the bank or other transaction institution. These two components are usually tightly related and form a common protocol and security system.

The security means for credit cards are rather low. Only since the end of the 1990s, the cards get equipped with a security chip. Before, only a magnetic strip or the embossed numbers of the card were used for identification, together with a signature. Both imprinted numbers and data on the magnetic strip are identical, up to some checksum used on the magnetic strip for communication integrity. The chip, which is nowadays on almost all cards, is very similar to those on typical smart cards and compatible with those on other debit cards. The chip provides parts of the transaction processing and authentication process on-chip and enables a completely encrypted communication towards the terminal.

Since attacks on credit cards are still rather easy by simply copying the data printed on the card, or stored on the magnetic strip, very few attacks were made against the chip so far. In most cases, either the card or the POS terminal are hacked or disturbed in such a way, that a low-security communication is enforced. Due to the worldwide distribution of credit cards, most systems still allow for such modes as fall-back for older cards. Copying the magnetic data and fishing the PIN as entered on tampered POS terminals allows for card copying up to correct magnetic strips, and thus is the most common attack.

However, also attacks on the software running on the card, or on the protocol between card's chip and terminal have been shown. One attack by Laurie et. al. could override the firmware on the chip. Murdoch et.al.

showed a man-in-the-middle attack on the protocol, which faked the acknowledge messages between terminal and card due to lack of encryption.

The German debit cards 'Electronic Cash' uses the same combination of magnetic strip and smart card chip, and are in parts compatible with credit card systems. However, they have a higher security mode for use with magnetic strips or the chip to allow for secure off-line transactions, i.e. in case the originating bank is not accessible or in order to speed up the whole transaction. Attacks on this system were also made on the card and on the terminal side. Even though the encryption scheme on the card was made weaker by some attacks, no known breach is known so far. However, the POS terminals have been hacked in order to harvest PINs and the encrypted magnetic data, often without even directly hacking any security measures but by mechanical intrusion.

Attacks on non-software based e-money systems are not yet known, supposedly due to the rather low acceptance of these systems and the limitations due to the restrictions to micro-payments.

2.2.1 Smart Cards

Smart cards are pocket-sized cards with embedded integrated circuits. They originated from simple credit and debit cards with read-only magnetic stripes and are now widely used for any kind of secure storing, authentication and transaction application. Most typical use is still banking and electronic payment, though smart cards are also massively emerging as replacements for identification and access cards as well as in health services.

Because of their use for accessing and processing very sensitive and private data, the level of security of smart cards can be considered very high. However, the interest of adversaries to hack smart cards for unauthorized access to private information or fraudulent use is even higher.

Typical attack scenarios for Smart Cards are either using HW or SW attacks, with the latter being the more common one. SW attacks to Smart Cards need to exploit any kind of known or unknown weakness of SW running on the internal CPU. Because access to the actual SW is quite hard (without a successful HW attack) most SW attacks are limited to exploiting weakness of the Smart Card communication interface and protocol.

HW attacks focus on getting access and manipulating the embedded memory (ROM or RAM) or manipulating the processing on the embedded controllers. There are several known HW attacks ranging from deleting EEPROMS through UV lights, removing the passivation of chips up to bugging the communication between the cards and terminals. For most of these attacks there are nowadays effective countermeasures. Remaining HW attacks (side-channel attacks, microprobing) are only available to highly specialized personnel with expensive equipment, making Smart Cards hacking lucrative for only governments and large organizations.

There is though a major drawback of Smart Cards: Their overall performance is very low, limiting their use to applications which do not require high performance computation beside some standard encryption algorithms.

2.3 Electronic devices

Besides PCs and Smart Cards there are a wide a range of other electronic devices incorporating a certain level of computation power and security. Most prominent examples are game consoles, TV set-top boxes and mobile devices.

2.3.1 Game consoles

In recent years, game consoles have been a very common use case for HW/SW security. In mostly all cases, such security is included to enable Digital Rights Management, i.e. controlling and ensuring that only licensed and original SW can be executed on the device.

Game consoles show what motivates consumers to hack their own devices. While in many cases piracy is the most obvious and visible result of a successful console hack, most of the originators of those hacks declare, that their intention was to get full access to the HW they own. In contrast to PCs and Smart Cards, the amount of private information stored in game consoles is very limited, though stored credit card information on compromised devices could be misused to buy digital assets without consent of the user.

However game consoles are also a good example, how even well designed security architectures fail, if single components of the system are not correctly implemented. Of the three current generation game consoles the Wii was hacked right after marked introduction with a successful SW-only hack only about 1 year later, the XBox 360 about 1 year after its start and the PS3 about 4 years later.

Today's modern game consoles combine different measures in HW and SW to create an effective security architecture. For example, the PS3 uses a combination of encrypted and signed boot loader, encrypted and signed executables, a hypervisor, and eFuses. There have been several successful attempts to use certain exploits of HW and SW components for hacking the PS3, though most of them could be fixed with SW updates. However through these series of attacks, hackers finally noticed a wrong implementation and use of the ECDSA (Elliptic Curve Digital Signature Algorithm) encryption which forms the basis of all PS3 security mechanisms, leading to full access to all private keys of the console. While previous hacks could be compensated by adding HW updates to newly produced models this hack could not be countered without breaking compatibility with existing models.

2.3.2 Mobile devices

Mobile devices are slowly replacing the classic PC as more and more applications and tasks are moving from computers to smart phones and tablet PCs. In that sense, the security risks and threats are quite similar to PCs. However, their architecture is much less standardized and the integration level of peripheral components and CPUs is significantly larger. Also on the SW side, operating systems (typically Android or iOS) are much more restricted compared to traditional PC systems.

On the first look, this brings an improvement to security. Users are not able to gain privileged rights on their devices, thus preventing them of running any kind of untrusted application or even modifying the kernel. Also due to the high level of integration of peripheral HW devices (such as GPU, USB controllers, sound cards, etc.) into the so-called *System-on-Chip*, bus snooping attacks are much more complex. However, such restrictions also increase the motivation to circumvent such measures. Similar to game consoles, users claim the right to fully exploit the HW they own including the right to run any kind of SW on it. In recent years, this led to the common practise of so-called *rooting* or *jailbreaking* of mobile devices. Previous successful attempts to jailbreak Android- and iOS-based devices in shortly after their release have shown that current countermeasures are not effective. In fact, such measures are mostly implemented in SW using either no or only limited HW assistance.

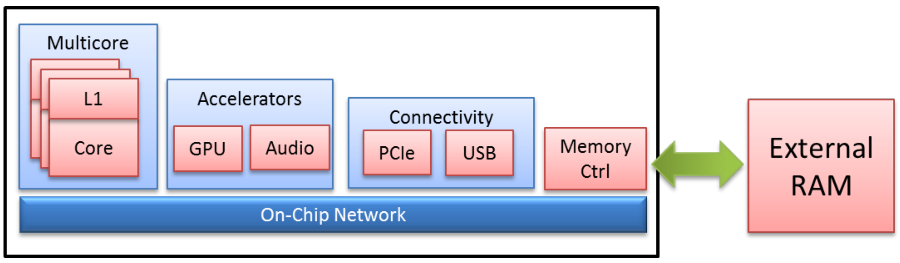

Figure 2.3: System-on-Chip connected via memory bus to external RAM

It can be expected that if manufacturers would include more HW security measures directly into the SoC, attempting to hack or attack the device would be much more complicated. Though manufacturers also have to keep in mind, that any kind of security restriction should not prevent users from fully exploiting the HW they own. Instead, such security measures should also be used to enable parallel execution of trusted and untrusted application without any risks of compromising the trusted part.

3 RELATED IT SECURITY STANDARDS

There are several security-related IT standards, though only very few of them directly address HW security issues. Furthermore, most of these standards define rules, processes and general requirements for designing and assessing security targets, but do not define actual security architectures or implementations.

In the following we will briefly review some of these standards and analyse their contribution to HW-related security.

3.1 ISO SC27 Standards

The ISO/IEC SC27 is a sub-committee of the joint committee of ISO and IEC. It focusses on the development of security standards for information and communication technology (ICT). The SC27 consists of several working groups covering different topics of information security:

WG1 develops standards for information security management systems. These standards (e.g. ISO27001, ISO27005) cover several aspects of security management including requirements, processes, risk management, etc. While it specifies requirements for security mechanisms and measures it does not define actual mechanisms.

WG2 covers standards and guidelines for cryptographic and non-cryptographic techniques and mechanisms including confidentiality, authentication, integrity, key management, etc. While these standards also define general requirements certain parts describe actual implementation that fulfil these requirements, e.g. AES, SEED.

WG3 focuses on standards for specification, evaluation and testing of IT security systems. This includes standards for security requirements of cryptographic modules. Though no actual implementations or architectures are defined or proposed.

WG4 covers standards for security control and services including techniques for network security and application security.

WG5 includes standards for identity management and privacy technologies. This includes for example ISO/IEC 29100 which defines a privacy framework.

While the SC27 maintains and develops a wide range of IT security standards its applicability for HW security and the development of the TRESCCA platform is rather limited. The standards mostly cover requirements for security management, control and assessment of security systems. However they do not define any standard architectures and measures, except for encryption algorithms. Though, from an application and system-level point of view TRESCCA should ensure that any IT system build on top of TRESCCA components should be able to fulfil SC27 standards requirements.

3.2 GlobalPlatform

The GlobalPlatform [2] organization is an industrial standardization group that is trying to standardize the concept of *Trusted Execution Environment*. The TEE is a trade-off between no security and the high security level provided by a secure element (e.g. a smart-card). A Trusted Execution Environment provides some applications with an execution environment isolated from the other applications that run on the system and so it prevents software attacks against the applications running on the trusted environment (but not between applications running inside the trusted environment or between applications running in the normal execution environment). The ARM TrustZone provides an implementation of such a Trusted Execution Environment but the standard allows a certain level of compatibility between different hardware and software implementation.

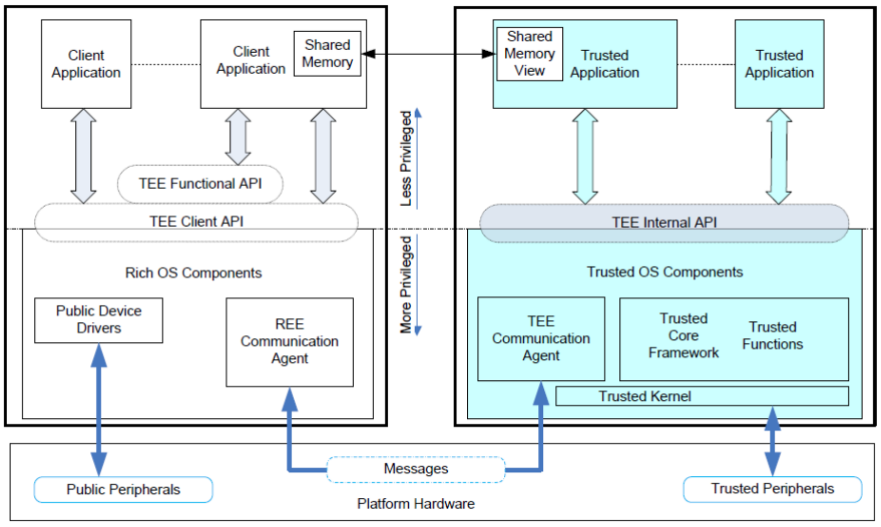

Figure 3.1: GlobalPlatform TEE system architecture

The GlobalPlatform TEE is relevant for TRESCCA, as it might provide a suitable API and hardware abstraction layer for applications to make use of the TRESCCA HW security. However, the origin of the GlobalPlatform group is the Smart Card industry and many of its concepts originate from there. It remains to be seen how far these concepts will be established in the domain of System-on-Chips for mobile and embedded devices.

3.3 Trusted Computing Group

The Trusted Computing Group [29] (TCG) is a consortium which aims at establishing specifications to increase the trust in computing devices (from mobile phones to servers). The trust, according to the definition of the TCG, is the expectation that a device will behave in a particular manner for a specific purpose. One of its main milestones was the publication of the specifications of the Trusted Platform Module (TPM). The TPM is a tamper resistant chip connected to the PC and provides services to securely store and manage private keys as well as to enable integrity measurement and attestation of PC systems.

The TPM is well established in today's PC systems. Nevertheless, due to privacy concerns its actual use is still limited. The TPM 2.0 standard which is currently under development includes interesting concepts, such as integrating a TPM into a System-on-Chip, which could be of relevance for TRESCCA. One option for TRESCCA is to investigate and demonstrate, how the open TRESCCA technology can be used to implement and integrate a TPM 2.0 conform element inside a chip.

3.4 Common Criteria for Information Technology Security Evaluation

The *Common Criteria for Information Technology Security Evaluation* or short *Common Criteria* (CC) are an international standard (ISO/IEC 15408) for certification of security systems. The current version is 3.1. The Common Criteria define a framework in which a system designer and/or users can specify functional and assurance security requirements for through *Protection Profiles*. Such profiles define a set of evaluation criteria a system has to fulfil. Most common approach is the definition of a so-called Evaluation Assurance Level (EAL), which defines a numerical grade for the confidence that a certain security feature has been implemented reliable. The CC standard defines EALs from 1 to 7, with 1 being the lowest grade (functionally tested) and 7 the highest (formally verified designed and tested). Typical EALs for security functions in software systems, e.g. user-based security in operating systems, are EAL4 (Linux, MS Windows XP/Vista/7) or EAL4+ (Solaris). Smart Cards and certain specialized SW products are certified to be EAL5 or even higher.

While the EAL implicates a specific set of procedures and processes which one has to follow and document when developing and testing the product, it does not directly define how the security feature is implemented.

There are several cases known, where EAL4 or EAL4+ certified products could be hacked successfully including certain products, such as Smart Cards and encrypted USB flash drives. This shows, that at least on those levels, the CC evaluation does not guarantee that the chosen implementation for the security feature is fully secure.

From the point of view of TRESCCA it could be an option to investigate, which CC related security features can be realized and what kind of proves it can provide to help certifying the reliability of that features. For example, as most of the TRESCCA components will be fully documented and open-source, the effort for EAL assessment could be reduced if TRESCCA components are reused.

4 SECURITY OF DATA IN THE CLOUD AND ON LOCAL DEVICES

This chapter describes the different solutions to protect the confidentiality and integrity of the client's data inside the cloud. All these solutions can be applied to protect the data of the cloud service provider when they are manipulated on the client side.

4.1 Security of communication between client and cloud

The security of the communications between the client's network or devices and the cloud provider's network can easily be protected using well known protocols such as SSL/TLS [9] or IPsec [18]. These protocols ensure end-to-end confidentiality, integrity and authentication.

Several attacks have targeted some versions or some implementations of these protocols (such as the BEAST [25] or CRIME [26] attacks).

One of the major problems of these protocols is that the authentication often relies on digital certificates delivered by certification authorities that have to be trusted. Internet browsers for instance, by default, trust tens of certification authorities from all around the world. If only one of these certification authorities delivers a valid certificate for address A to the wrong entity B, B can perform man-in-the-middle attacks and retrieve or modify all the data between the client and A without being detected. However, this type of incident happened several time during the past few years (in 2001, VeriSign delivered code signing certificates in the name of Microsoft to fake employees; in 2011, DigiNotar was hacked and valid fake certificates were delivered in the name of several high-value websites such as Google; in 2011, TurkTrust accidentally delivers intermediate CA certificate to two of its clients and one of them was used later for signing valid fake certificate for Google website, etc.).

Authentication of clients often relies on passwords. However, passwords are vulnerable to brute-force attacks, dictionary attacks or phishing. This is why several cloud based services are migrating toward two-factors authentication techniques that often combine a traditional password and a one-time code generated by either a secure token or delivered to the client via another channel (for instance a mobile phone).

4.2 Encryption on the client's side

The first way for the client to be sure that the confidentiality and integrity of its data are protected inside the cloud is to encrypt them before sending them to the cloud. If the encryption key is not known by the cloud provider and if the encryption algorithm is strong enough, the confidentiality of the data cannot be violated, either during the transport to the servers or during their storage. There are several applications making the process of encryption and decryption transparent for the user, such as BoxCryptor [1].

Strong encryption prevents adversaries from accessing the sensitive data but the access patterns of the data owners can however be observed and analysed by the cloud provider, the end user or eavesdroppers. These access patterns (which data are accessed, when, how many times, in what order) can sometimes be exploited to expose sensitive information or to compromise privacy. Oblivious transfer[24] is an active research field aiming at masking the access patterns. The TRESCCA project deals with lower level hardware and software security features. The TRESCCA platform is thus perfectly compatible with the oblivious transfer protocols. If they are needed for a particular application, these transfer protocols can be implemented on top of the TRESCCA infrastructure, both on the client and cloud sides.

One of the major drawbacks of data strong encryption is that the cloud end user can only take benefit from the storage capacity of the cloud and not the computing power of it because the servers, without the encryption keys, cannot perform computations on the encrypted data. Symmetrically, the cloud operators cannot delegate computations involving sensitive data to the large farm of light clients located at the end users.

4.3 Homomorphic encryption

One solution to this problem would be the homomorphic encryption. Homomorphic encryption is a class of encryption algorithms and protocols that allow an entity to perform useful computations directly on encrypted data without having to decrypt them before and so, without having the encryption key. Applied to the cloud scenario, homomorphic encryption would allow the client to encrypt its data before sending them to the cloud and the servers to perform useful computations on these encrypted data. Thus, the client would have the guarantee that the confidentiality of its data is protected inside the cloud (the cloud service provider does not have the key to decrypt the client's data) but would also be able to use the computing power provided by the cloud.

There are two kinds of homomorphic encryption. The first one, usually called partially homomorphic encryption, can perform in encrypted form only one of two possible operations defined on a field (e.g. addition or multiplication, but not both). Several encryption schemes are naturally homomorphic, like, for instance, the modular exponentiation with which the product of two ciphertexts is the ciphertext of the product of the two plaintexts. The partially homomorphic encryption is used in some very specific cases for which it is efficient and simple. Unfortunately, because it covers only one operation, it is not usable for arbitrary computations. The second kind, usually called fully homomorphic encryption, or homomorphic encryption for short, can perform in encrypted form the two operations defined on a field. It is thus suitable for arbitrary computations.

The first fully homomorphic scheme was presented in 2009 by Craig Gentry [13, 12]. Several other schemes have been presented since but they are not sufficiently efficient (in term of computing time or memory space requirement) to solve the issue of cloud security. The most promising current implementations are still orders of magnitude from what would be considered as a reasonable overhead. In [14], for instance, the authors implemented an optimized fully homomorphic scheme on a high end desktop computer (64-bits, quad-core Intel Xeon E5450 processor at 3 GHz with 12MB L2 cache and 24GB of RAM). The public key size was 2.25 GBytes, the time taken for the key generation was 2.2 hours and the recryption operation, which is required periodically during computations to reduce the computation noise, took 31 minutes. All in all this implementation is orders of magnitudes less powerful than the old ENIAC.

However, this research topic is very active and may provide an efficient homomorphic scheme in the long term.

4.4 Trusted execution environments

As for the moment, the homomorphic encryption schemes are not practical, if the client wants its data to be manipulated inside the cloud while guaranteeing the confidentiality. Also a service provider that wants to execute applications on the client device needs a way to attest its trustworthiness before he delegates any processing to the device. Both cases require a *trusted execution environment* which could be available on the client as well as on the cloud server.

These trusted execution environments can be divided into two categories depending on the threats they protect the client's applications and data from. They can prevent software threats (*i.e.* properly isolate the client's applications from other applications, including the operating system, running on the same machine, or prove to the client that the software execution environment can be trusted), or hardware threats (*i.e.* protect client's applications from adversary, such as a rogue system administrator, that has a physical access to the machine).

4.4.1 Protection against software threats

This section presents the different solutions against software threats. All these solutions are vulnerable to some hardware attacks.

ARM TrustZone

The ARM TrustZone technology [3, 15] prevents software attacks against sensitive applications on embedded systems by partitioning all the hardware and software resources into two parts: the normal world and the secure world. Software or hardware components of the normal world are prevented from communicating or interfering with the secure world. So a malware running in the normal world would be unable to access or disturb a critical application running in the secure world. The partitioning between the two worlds impacts the processor (a secure execution mode is added to the traditional execution modes in order for the two worlds to be able to share the same processor), the main interconnect bus of the SoC controls the communications depending on the currently active world, the IPs which are allocated to one of the two worlds and the memory controller which can divide the memory space between the two worlds.

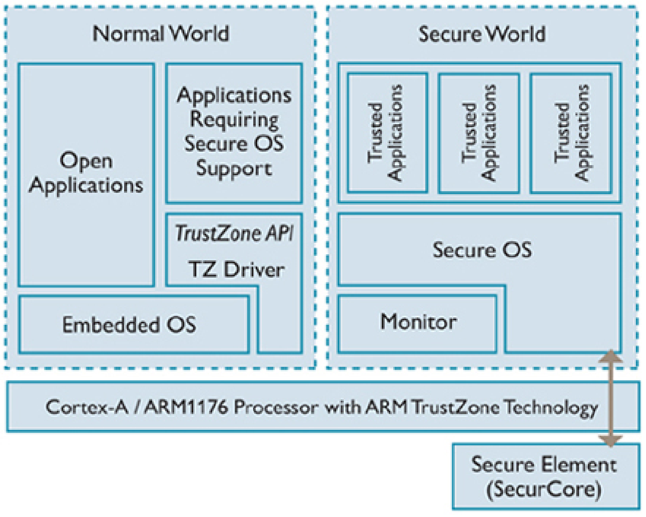

Figure 4.1: TrustZone SW architecture (ARM)

TPM

The Trusted Platform Module is a tamper-resistant chip (usually using the same technologies as smart-card chips) connected to the motherboard of PC or servers and has three main features: integrity measurement and storage, remote attestation and secure storage.

Integrity measurement and attestation During the boot of the PC, measurements of the elements that may affect the trustworthiness of the platform (mainly the pieces of software that are run during the boot: BIOS, PCI boards BIOS ROM, bootloader, kernel of the operating system, etc.) are performed and sent to the TPM for storage, where they are stored into special registers named Platform Configuration Registers (PCR). During the life of the platform, the TPM can attest, using digital signature with cryptographic keys embedded inside the TPM, of the content of the PCR, and so, it can attest of the current state of the platform. The secure storage

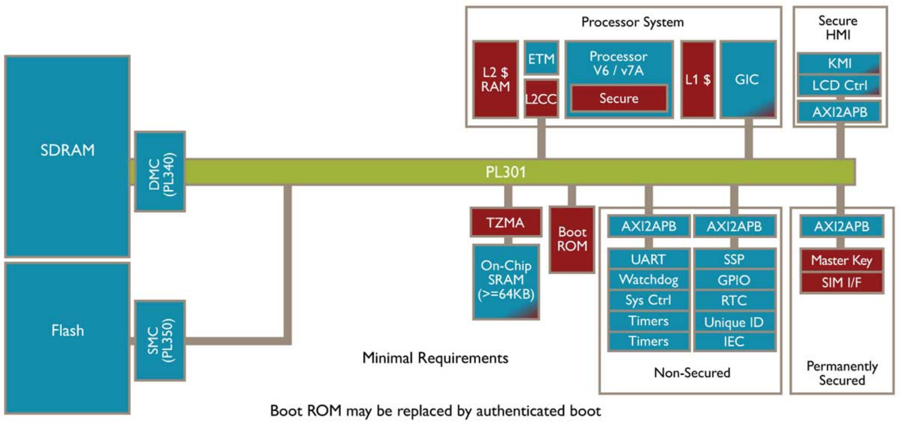

Figure 4.2: Example of TrustZone system architecture (ARM). Red = protected components, Blue = unprotected components.

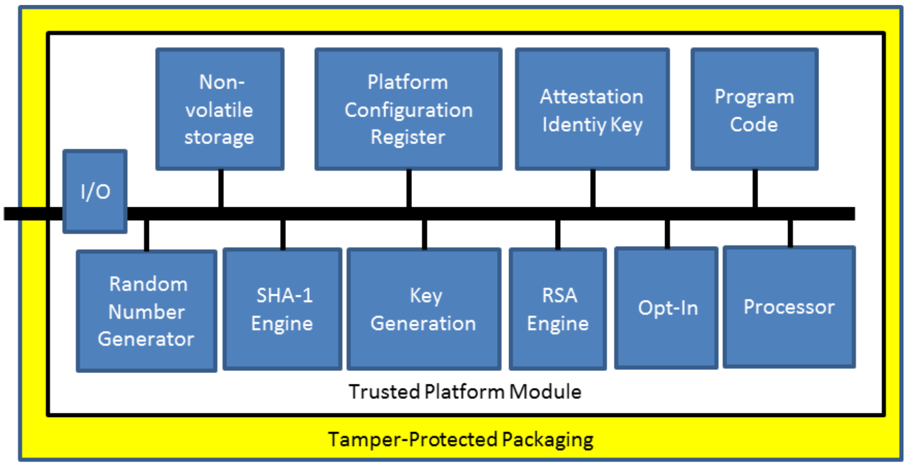

Figure 4.3: TPM architecture overview

feature allows an application to store data that will be recoverable only if the state of the platform (values contained in the PCR) matches a state specified by the application.

Key management The TPM offers a secure storage location for RSA based encryption keys. The private part of a stored key resides within the TPM and is never revealed. The TPM has a limited number of key slots. To be able to handle more keys the TPM is using the concept of a key hierarchy. Private parts of newly created keys are encrypted with a parent key building up a key chain to the storage root key which is created if a user declares ownership over a certain TPM. The storage root key (SRK) therefore is the root node in the key chain hierarchy. In this way keys can be securely offloaded to external storage e.g.to file system without exposing them.

Several different types of keys are envisaged in the specification of TCG:

- Signing Key
- Storage Key
- Identity Key

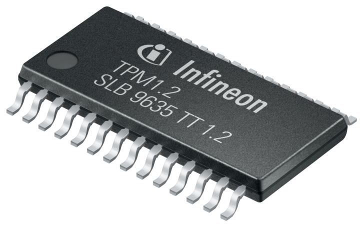

Figure 4.4: Infineon SLB9635 TPM chip

- Authchange Key
- Bind Keys
- Legacy Key

The TPM has an Endorsement Key(EK) embedded. The private key PrivEK cannot leave the TPM. The manufactor has the responsibility to create and insert this key into every TPM. The EK plays a very important role, as it is the only way to identify a valid TPM. The public portion is called PubEK and created a lot of privacy concerned discussions. As the EK stays the same the whole lifetime of the TPM and would be used for identification a user could be reliably tracked which creates an enormous thread to privacy. Sure, if no sensible information is mapped to a PubEK privacy is not in danger. It's first of all just a random number but it offers the possibility to threaten privacy.

Issues The TPM does not prevent hardware attacks. It is even quite easy to perform a man-in-the-middle attack on the Low-Count Pin (LCP) bus of the motherboard on which the TPM is connected. This attack allows an adversary to store false values in the PCR of the TPM and so make believe that the platform is in a different state than it is. Another problem with the TPM is that, due to the high number of possible versions of the pieces of software measured, the creation of the list of "trusted" values can be difficult.

Another issue also related to the nature of the TPM of being an external component is its limited processing performance and capabilities, while it includes specific crypto processors for executing cipher algorithms its overall performance is more comparable to microcontrollers than state-of-the-art CPUs. Furthermore, the internal memory of the TPM is very limited only allowing storage of a small set of key.

Finally, though the TPM is an industry standard its internals are not fully documented. Also until recent, there was no official certification program by the TCG testing and certifying the compliance of any TPM device to the standard. Several of such issues including doubts about the protection of privacy lead to a general mistrustfulness against the TPM, though most of its general concepts could be considered a benefit for PC security.

Intel Trusted Execution Technology

The Intel Trusted Execution Technology [17] allows the dynamic creation of trusted execution environments (virtual machines) whose state can be cryptographicaly attested and which are protected against software attacks originated from the outside of these environments. In addition to modifications to the chipset and to the processor, this technology relies on a TPM (to store integrity measurement and to attest them) and virtualization technologies (VT-x – hardware-assisted virtualization – and VT-d – IO MMU – to properly isolate virtual machines).

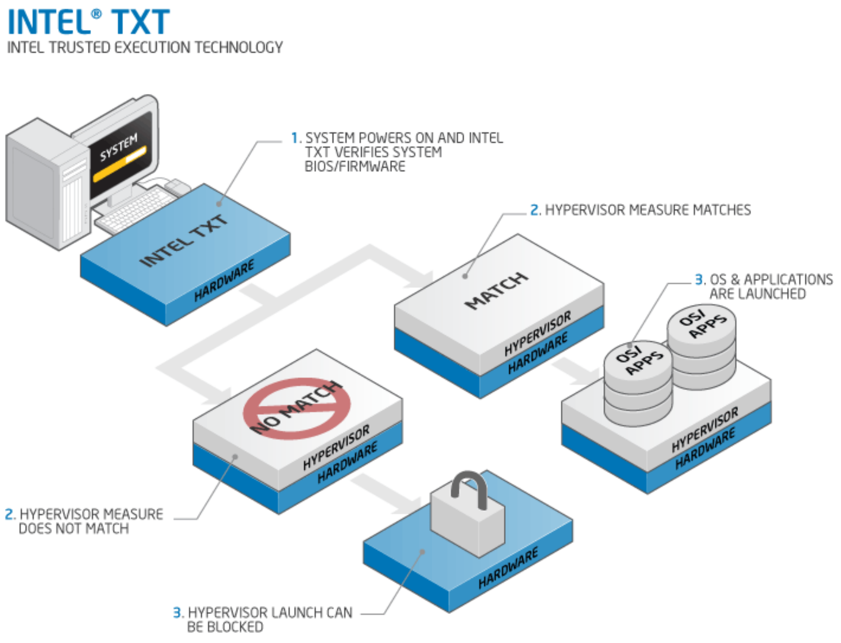
INTEL® TXT
INTEL TRUSTED EXECUTION TECHNOLOGY
SYSTEM
INTEL TXT
HARDWARE
1. SYSTEM POWERS ON AND INTEL TXT VERIFIES SYSTEM BIOS/FIRMWARE
2. HYPERVISOR MEASURE MATCHES
MATCH
HYPERVISOR
HARDWARE
3. OS & APPLICATIONS ARE LAUNCHED
OS/ APPS
OS/ APPS
HYPERVISOR
HARDWARE
NO MATCH
HYPERVISOR
HARDWARE
2. HYPERVISOR MEASURE DOES NOT MATCH
HARDWARE
3. HYPERVISOR LAUNCH CAN BE BLOCKED

Figure 4.5: Intel TXT SW architecture

During the boot of the server, its state (BIOS, PCI ROM, bootloader, hypervisor...) is measured and the integrity measurements are stored inside the TPM. These measurements, which can be remotely attested by the TPM, in combination with a list of known trusted values, allow to know whether the platform (usually up to the hypervisor or the kernel of the OS because after, too many drivers, applications, etc. lead to too many different integrity measurements) is in a trusted state or not. During the life of the platform, new trusted virtual machines can be created: when the system needs to create a virtual machine, a special instruction of the processor executes a small piece of code (called Authenticated Code) directly inside the processor, which is responsible for configuring the hardware so the new virtual machine is totally protected from the other applications running on the system (thanks to VT-x and VT-d technologies). The processor also provides an attestation that the virtual machine was properly launched and that it can be trusted.

However, a bug has been found [31] in the first versions of the secure code that is supposed to configure properly the virtual machine, that leads to a bypass of the TXT security.

4.4.2 Protection against hardware threats

The protection against an adversary that has a physical access to the machine is more difficult and imposes specialized hardware components.

All solutions presented in this section make the hypothesis that the adversary cannot perform attacks directly against a chip (using passive techniques such as Side-Channel Analysis or active techniques such as micro-probing or fault injections) but only board-level attacks (such as spying on the memory bus).

Under this assumption, everything can be protected in software except the communications between the processor and its main volatile memory (RAM) because all applications are executed from this main memory so their code and data must be transferred on the memory bus where they can be spied or tampered. Software obfuscation techniques can make these bus probing attacks more difficult but not impossible.

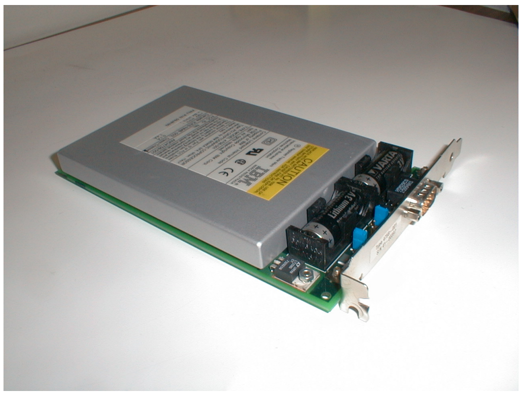

Figure 4.6: Photo of IBM 4758 Cryptographic Coprocessor (courtesy of Steve Weingart)

Secure Element

The first solution is to perform the decryption of the data, the computations on decrypted data and the encryption of the results inside a inaccessible tamper-resistant computing element. With such a solution the memory bus is not exposed to attackers and bus probing becomes as difficult as on-chip attacks. This computing element can be for instance a smart-card.

Typical example of a Secure Element is a smart card as described in section 2.2.1 embedding a processor, a RAM, a non-volatile memory (Flash, EEPROM) and sometimes dedicated IPs (for instance cryptographic accelerators) inside a single chip. Smart-card chips also often include hardware and software counter-measures against different types of physical attacks that make them highly tamper-resistant devices. They are used for sensitive applications such as banking applications or mobile phone communications (the identification of the subscriber is perform using a smart-card called the Subscriber Identity Module, SIM). The major drawback of smart-cards, that make them unsuitable for cloud applications, is their limited computing power.

Another approach, similar to smart-card but at a larger scale is Hardware Security Module, such as the IBM Cryptographic Coprocessors [16] (see Figure 4.6). They are extension cards, that can be plugged into a server, and that contain an execution environment (processor, volatile and non-volatile memory) inside a tamper-proof shield. They embed also some sensors that are design to erase the stored secrets if an attempt to break the shield is detected. They are often used to store the private keys for secure web servers of certification authorities but less to perform secure computations.

Memory bus-encryption architectures

With the second solution the memory bus is exposed to attackers, on board, but encryption and integrity checking are used to protect the data that are transported on this bus (and so that are stored in the main memory).

Outside the military world, the first architecture that uses memory encryption was proposed in 1979 by Robert Best [4, 5, 6, 7] in order to prevent software piracy. He describes a micro-processor which embeds a cryptographic unit located on the chip, between the core of the processor and the pins of the memory bus. This unit is in charge of encrypting all data that are sent outside of the CPU and decrypting data that are read by the CPU.

The Dallas Semiconductor DS5002FP [8] micro-controller is an example of such an architecture. However, it does not protect the integrity of data and using this flaw, Markus Kuhn succeeded to break this micro-controller [19].

Since 2000, several architectures have been proposed by the academic world that combines encryption and integrity protection (including the difficult and costly protection against replay attacks), such as XOM [21, 20, 22], Aegis [27, 11, 28] and CryptoPage [10]. These architectures impose modifications to the CPU core that may not be economically feasible.

4.5 Comparison of HW security solutions and their coverage of TRESCCA security objectives

Comparing different HW security measures and their effectiveness against security threats is difficult because their direct functionalities and implementations can differ significantly. The security solutions described in this report can be roughly categorised into two different types:

1. **Isolated HW security elements**: These elements provide security functions and services in an isolated component and environment. All functionality is encapsulated within this element. Any security is limited to the component itself, e.g. any data or processing is only protected as long as it resides inside the component. Solutions that fall into this category are e.g. Trusted Platform Module and Secure Element.

2. **Complementary HW security elements**: These elements provide security base services that can be used by other HW and SW components to implement more complex higher level security functions. Solutions that fall into this category are e.g. ARM TrustZone, Intel TXT, Memory Encryption and NoC Firewalls.

The security threats and TRESCCA security objectives defined in deliverables D1.2 and D1.3 do not directly address HW security but instead higher level software security. Nevertheless any security service implemented on the software layer which is not protected against threats and attacks on the HW level is likely to fail. Table 4.5 compares the previously described security solutions wrt. to their coverage of the security objectives of TRESCCA which were introduced in deliverable D1.3. In addition to the security objectives of D1.3 two additional objectives have been added:

Tamper resistance is the resistance against any type of physical attack to the chip itself, e.g. trying to read the content of an on-chip memory. Such attacks typically require expensive equipment, typically not available to hobbyists but which may be available to larger organisations or governments.

Robustness against bus sniffing is the ability to protect the integrity and confidentiality of any data that is transmitted via any chip interfaces and signals directly accessible on the board of an electronic system. Bus sniffing does not require expensive equipment and can easily be executed by hobbyists and hackers.

ARM TrustZone is a complimentary security solutions that enables separation of software artefacts (virtual machines, operating systems, applications, processes) by providing a secure and a non-secure world that is separated on the HW level. Given that other HW components in the System-on-Chip respect the HW signals to recognize and follow the policies for the two worlds, TrustZone can provide a basis to build up and secure a range of more sophisticated security services (incl. realization of a software TPM, isolation of SW components, attestation support). However, as the memory bus is not encrypted a HW attack like bus sniffing can easily compromise any of the realized security services.

The Trusted Platform Module and the Secure Element are isolated HW security components that provide a secure and tamper resistant compartment in which several security services can be implemented. However due to their isolation, protection is limited to any functionality implemented within that compartment. This limits the applicability for other more sophisticated SW security services, like isolation and robustness against SW exploits. While it provides a high level of tamper resistance any data going to or leaving the device is not protected and could be compromised.

Table 4.1: Comparison of HW security solutions wrt. to their coverage of TRESCCA security objectives.

	HW Security Technologies				
TRESCCA Security Objectives	ARM TrustZone	Trusted Platform Module/Secure Element	Intel Trusted Execution Technology	Memory bus encryption (TRESCCA)	Network-on-Chip Firewalls (TRESCCA)
Trusted Compartment (existence, confidentiality and integrity of TC)	o	++	+	++	++
Robustness against software denial of service	+	NA	NA	NA	++
Attestation (remote attestation of TC)	o	+	+	++	NA
Remote authentication of delegated processing	o	NA	+	+	NA
Software update and boot-to-boot integrity	+	+	++	+	NA
Isolation between applications (and hypervisor)	+	NA	+	++	++
Robustness against exploits and other logical attacks	+	NA	o	NA	++
Tamper resistance (robustness against physical attacks)	o	++	+	o	o
Robustness against bus sniffing	-	-	o	++	NA

Intel Trust Execution Technology combines and integrates technology like a TPM, an integrated and protected memory area and IOMMU to provide secure boot, VM isolation and dynamic attestation/measurements services. In that sense it is more capable than a single TPM and TrustZone. Though most of its advantages come due to the fact that it is a proprietary and fixed technology that is specifically tailored for Intel CPUs. It also vulnerable against memory bus sniffing attacks but provides a bit more security due to its internal memory for first stage boot loaders. It also provides a higher level of tamper resistance due to its integrated memory though it is not as resistant as a TPM itself.

The memory encryption module proposed by TRESCCA is a very elementary and complementary security element that enables a wide range of SW based security services, as all code of that SW can be secured within the external RAM. However its robustness against denial of service attacks and SW exploits is limited to protecting the memory of applications against each others. Its tamper resistance is comparable with Intel TXT as it is integrated into the System-on-Chip but not not specifically protected against physical attacks on chip level.

The Network-on-Chip firewalls also proposed by TRESCCA are complementary to the memory encryption by covering those security objectives that cannot be realized just by memory bus encryption. The NoC firewalls enable the definition of fine grained access control rights to internal chip peripherals and interfaces on the level of virtual machines. Due to its fine granularity and its support for multiple VM instances it goes far beyond the concept of TrustZone which only enables the separation of the SW architecture into two worlds.

5 CONCLUSION

This report reviewed and analysed the current state-of-the-art of HW security solutions and how they can be used to protect against the security threats defined by TRESCCA.

Current developments in IT security have shown that SW security measures are not sufficient to protect against common threats. Even more, successful attacks against typical devices such as mobile phones, tablet PCs, gaming consoles show that hacks can be executed with low cost equipment and without special knowledge on the internals of a system.

Though there are several IT security standards from official standardization organisations like ISO/IEC and from industry groups, such standard either do not address HW security measures specifically or are not effective. Standards like the Trusted Platform Module and the GlobalPlatform Trusted Execution Environment provide a base for establishing HW security measures in a wide range of devices, though their potential is currently limited through lack of transparency and openness. TRESCCA will continue to monitor such standards and investigate how far the TRESCCA solutions can comply to or extend such standards. One promising candidate is the GlobalPlatform TEE which could provide a suitable API and hardware abstraction layer for the TRESCCA HSM components and its lower level SW drivers.

There are several aspects to securing the client device and its data when connected to the cloud. First of all, communication between the client and cloud needs to be secure. As long as the underlying base of HW and SW can be trusted and has not been compromised, any trusted SW service for communication protection is sufficient. Any HW support can be limited to encryption processing acceleration.

It is more difficult if the client's data needs to be protected when it is processed in the cloud. In general all data could be encrypted before it being transferred to the cloud. Except for homomorphic encryption, no computation could be performed on such data. However, homomorphic encryption is still in its infancy, computationally complex and therefore currently not practicable. Alternatively computation on client's data can be protected by Trusted Execution Environments (TEE), i.e. a HW/SW system whose trustworthiness attested. The same can be applied for enabling the execution of trusted services on client devices. There are several approaches for TEE, such as TPM, ARM TrustZone and Secure Elements. Some of these are still vulnerable to external HW attacks.

With many peripheral devices now being integrated into the System-on-Chip it becomes more difficult to attack communication between those devices and the CPU. Although the most obvious attack point is still available: the bus to the external memory. One benefit of encrypting the external memory bus is that almost any other security function can be implemented in SW as long as its memory area is protected. Currently, there are some solutions available for protecting the memory bus. However most of them are not mature enough or suffer performance issues. TRESCCA will take these solutions as base for developing an improved and more flexible memory encryption mechanism that is compatible with a wide range of SoC architectures and provides a good and configurable balance between performance and security.

6 ACRONYMS

AES	Advanced Encryption Standard
API	Application Programming Interface
BIOS	Basic Input Output System
CC	Common Criteria
CPU	Central Processing Unit
DMA	Direct Memory Access
EAL	Evaluation Assurance Level
ECDSA	Elliptic Curve Signature Algorithm
EEPROM	Electronically Erasable Programmable Read-Only Memory
EK	Endorsement Key
FPGA	Field Programmable Gate Array
GPU	Graphics Processing Unit
HW	Hardware
IC	Integrated Circuit
ICT	Information and Communication Technology
IOMMU	Input/Output Memory Mapping Unit
ISO	International Standardization Organization
IT	Information Technology
ITU	International Telecommunications Union
LPC	Low Pin Count
PC	Personal Computer
PCI	Peripheral Component Interconnect
PCR	Platform Configuration Register
POS	Point of Sale
RAM	Random Access Memory
ROM	Read-Only Memory
RSA	Rivest, Shamir, Adleman
SIM	Subscriber Identity Module
SRK	Storage Root Key
SSL	Secure Sockets Layer
SW	Software
TCG	Trusted Computing Group
TEE	Trusted Execution Environment
TPM	Trusted Platform Module

Continued on next page –

– continued from previous page	
TXT	Trusted Execution Technology
USB	Universal Serial Bus

BIBLIOGRAPHY

[1] Boxcryptor, January 2013. `https://www.boxcryptor.com`.

[2] Globalplatform, January 2013. `http://www.globalplatform.org/`.

[3] ARM. Trustzone, January 2013. `http://www.arm.com/products/processors/technologies/trustzone.php`.

[4] Robert M. Best. Microprocessor for executing enciphered programs. Technical Report US4168396, United States Patent, September 1979.

[5] Robert M. Best. Preventing software piracy with crypto-microprocessors. In *IEEE Spring CompCon'80*, pages 466–469. IEEE Computer Society, February 1980.

[6] Robert M. Best. Crypto microprocessor for executing enciphered programs. Technical Report US4278837, United States Patent, July 1981.

[7] Robert M. Best. Crypto microprocessor that executes enciphered programs. Technical Report US4465901, United States Patent, August 1984.

[8] Dallas Semiconductor. *DS5002FP Secure Microprocessor Chip*, July 2006. `http://datasheets.maxim-ic.com/en/ds/DS5002FP.pdf`.

[9] T. Dierks and C. Allen. The TLS protocol — version 1.0. RFC 2246, Internet Engineering Task Force, January 1999. `http://tools.ietf.org/html/rfc2246`.

[10] Guillaume Duc and Ronan Keryell. CryptoPage: an efficient secure architecture with memory encryption, integrity and information leakage protection. In *Proceedings of the 22th Annual Computer Security Applications Conference (ACSAC'06)*, pages 483–492. IEEE Computer Society, December 2006.

[11] Blaise Gassend, G. Edward Suh, Dwaine Clarke, Marten van Dijk, and Srinivas Devadas. Caches and hash trees for efficient memory integrity verification. In *Proceedings of the 9th International Symposium on High-Performance Computer Architecture (HPCA'03)*, pages 295–306, February 2003.

[12] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. `http://crypto.stanford.edu/craig`.

[13] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC)*, pages 169–178. ACM, 2009.

[14] Craig Gentry and Shai Halevi. Implementing gentry's fully-homomorphic encryption scheme. Cryptology ePrint Archive, Report 2010/520, 2010. `http://eprint.iacr.org/`.

[15] Tom R. Halfhill. ARM dons armor: Trustzone security extensions strengthen ARMv6 architecture. *Microprocessor Report*, 8/25/03-01, August 2004.

[16] IBM. Ibm pcie cryptographic coprocessor, January 2013. `http://www-03.ibm.com/security/cryptocards/pciecc/overview.shtml`.

[17] Intel. Trusted compute pools with intel(r) trusted execution technology (intel(r) txt), January 2013. `http://www.intel.com/content/www/us/en/architecture-and-technology/trusted-execution-technology/malware-reduction-general-technology.html`.

[18] S. Kent and K. Seo. Security architecture for the internet protocol. RFC 4301, Internet Engineering Task Force, December 2005. `http://tools.ietf.org/html/rfc4301`.

[19] Markus G. Kuhn. Cipher instruction search attack on the bus-encryption security microcontroller DS5002FP. In *IEEE Transaction on Computers*, volume 47, pages 1153–1157. IEEE Computer Society, October 1998.

[20] David Lie, John Mitchell, Chandramohan A. Thekkath, and Mark Horowitz. Specifying and verifying hardware for tamper-resistant software. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, pages 166–177, May 2003.

[21] David Lie, Chandramohan Thekkath, Mark Mitchell, Patrick Lincoln, Dan Boneh, John Mitchell, and Mark Horowitz. Architectural support for copy and tamper resistant software. In *Proceedings of the Ninth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS IX)*, pages 168–177, October 2000.

[22] David Lie, Chandramohan A. Trekkath, and Mark Horowitz. Implementing an untrusted operating system on trusted hardware. In *Proceedings of the 9th ACM Symposium on Operating Systems Principles (SOSP'03)*, pages 178–192, October 2003.

[23] Paysafecard. Website, November 2013. `https://www.paysafecard.com`.

[24] Michael Rabin. How to exchange secrets by oblivious transfer. Technical report, Aiken Computation Laboratory, Harvard University, 1981.

[25] Juliano Rizzo and Thai Duong. BEAST: Surprising crypto attack against HTTPS. In *Ekoparty Security Conference*, September 2011.

[26] Juliano Rizzo and Thai Duong. The CRIME attack. In *Ekoparty Security Conference*, September 2012.

[27] G. Edward Suh, Dwaine Clarke, Blaise Gassend, Marten van Dijk, and Srinivas Devadas. AEGIS: Architecture for tamper-evident and tamper-resistant processing. In *Proceedings of the 17th International Conference on Supercomputing (ICS'03)*, pages 160–171, June 2003.

[28] G. Edward Suh, Charles W. O'Donnell, Ishan Sachdev, and Srinivas Devadas. Design and implementation of the AEGIS single-chip secure processor using physical random functions. In *Proceedings of the 32nd Annual International Symposium on Computer Architecture (ISCA'05)*, pages 25–36. IEEE Computer Society, June 2005.

[29] TCG. Trusted computing group, January 2013. `http://www.trustedcomputinggroup.org/`.

[30] Wirecard. Company website, November 2013. `http://www.wirecard.com`.

[31] Rafal Wojtczuk and Joanna Rutkowska. Attacking intel TXT(r) via SINIT code execution hijacking. Technical report, Invisible Things Lab, November 2011. `http://www.invisiblethingslab.com/resources/2011/Attacking_Intel_TXT_via_SINIT_hijacking.pdf`.