Journal of Information Assurance and Security 4 (2009) 449-457

Trusted-Computing Technologies for the Protection of Critical Information Systems[†]

Antonio Lioy, Gianluca Ramunno and Davide Vernizzi

Politecnico di Torino
Dip. di Automatica e Informatica
c. Duca degli Abruzzi, 24 – 10129 Torino (Italy)
{ antonio.lioy, gianluca.ramunno, davide.vernizzi } @ polito.it

***Abstract*:** Information systems controlling critical infrastructures are vital elements of our modern society. Purely software-based protection techniques have demonstrated limits in fending off attacks and providing assurance of correct configuration. Trusted computing techniques promise to improve over this situation by using hardware-based security solutions. This paper introduces the foundations of trusted computing and discusses how it can be usefully applied to the protection of critical information systems.

***Keywords*:** Critical infrastructure protection, trusted computing, security, assurance.

1 Introduction

Trusted Computing (TC) technologies have historically been proposed by the TCG (Trusted Computing Group) to protect personal computers from those software attacks that cannot be countered by purely software solutions. However these techniques are now mature enough to spread out to both bigger and smaller systems. Trusted desktop environments are already available and easy to setup. Trusted computing servers and embedded systems are just around the corner, while proof-of-concept trusted environments for mobile devices have been demonstrated and are just waiting for the production of the appropriate hardware anchor (MTM, Mobile Trusted Module).

TC technologies are not easily understood and to many people they immediately evoke the "Big Brother" phantom, mainly due to their initial association with controversial projects from operating system vendors (to lock the owner into using only certified and licensed software components) and from providers of multimedia content (to avoid copyright breaches). However, TC is nowadays increasingly being associated with secure open environments, also thanks to pioneer work performed by various projects around the world, such as OpenTC [1], co-funded by the European Commission.

On the other hand, we have an increasing number of vital control systems (such as electric power distribution, railway traffic, and water supply) that heavily and almost exclusively rely on computer-based infrastructures for their correct operation. In the following we will refer to these infrastructures with the term "Critical Information Systems" (CIS) because their proper behaviour in handling information is critical for the operation of some very important system.

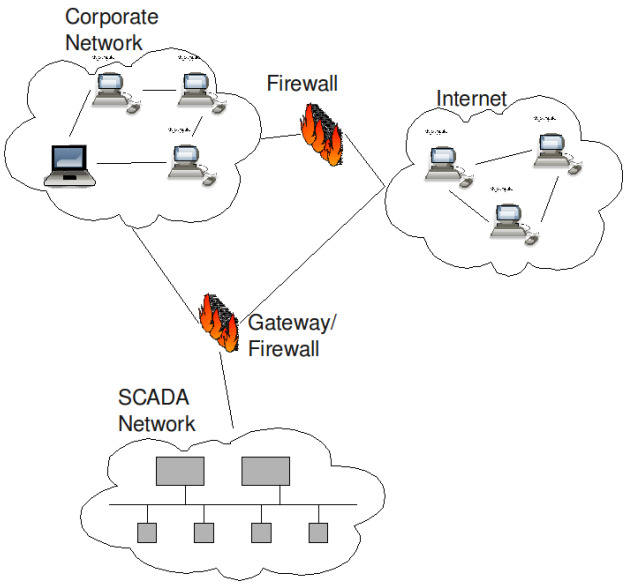

Figure 1: Typical CIS architecture

This paper briefly describes the foundations of TC and shows how they can help in creating more secure and trustworthy CIS.

2 Critical Information Systems (CIS)

CIS are typically characterized by being very highly distributed systems, because the underlying controlled system (e.g. power distribution, railway traffic) is highly distributed itself on a geographic scale (Figure 1).

In turn this bears an important consequence: it is nearly impossible to control physical access to all its components and to the communication channels that must therefore be

[†] This paper is an extended version of the paper by the same title that appeared in the proceedings of the CISIS'08 conference.

Received June 10, 2009

1554-1010 $ 03.50 Dynamic Publishers, Inc.

very trustworthy. In other words, we must consider the likelihood that someone is manipulating the hardware, software, or communication links of the various distributed components. This likelihood is larger than in normal networked systems (e.g. corporate networks) because nodes are typically located outside the company's premises and hosted in shelters easily violated. For example, think of the small boxes containing control equipment alongside railway tracks or attached to electrical power lines.

Furthermore, CIS are usually composed by nodes of many types: for instance sensors that belong to a wireless sensor network (WSN) can be used to read data from the physical environment, while servers based on general purpose hardware collect and analyse those data. Because of the heterogeneous nature of the control networks, different protection mechanisms must be applied according to the type of the node.

An additional problem is posed by the fact that quite often CIS are designed, developed, and deployed by a company (the system developer), owned by a different one (the service provider), and finally maintained by yet another company (the maintainer) on a contract with one of the first two. When a problem occurs and it leads to an accident, it is very important to be able to track the source of the problem: is it due to a design mistake? or to a development bug? or to an incorrect maintenance procedure? The answer has influence over economical matters (costs for fixing the problem, penalties to be paid) and may be also over legal ones, in the case of damage to a third-party.

Even if no damage is produced, it is nonetheless important to be able to quickly identify problems with components of a CIS, being them produced incidentally or by a deliberate act. For example, in real systems many problems are caused by mistakes made by maintainers when upgrading or replacing hardware or software components.

Any technical solution that can help in thwarting attacks and detecting mistakes and breaches is of interest, but nowadays the solutions commonly adopted – such as firewall, VPN, and IDS – heavily rely on correct software configuration of all the nodes. Unfortunately this cannot be guaranteed in a highly distributed and physically insecure system as a CIS. Therefore better techniques should be adopted not only to protect the system against attacks and errors but also to provide assurance that each node is configured and operating as expected. This is exactly one of the possible applications of the TC paradigm, which is introduced in the next section.

3 Trusted Computing principles

In order to protect computer systems and networks from attacks we rely on software tools in the form of security applications (e.g. digital signature libraries), kernel modules (e.g. IPsec) or firmware, as in the case of firewall appliances. However software can be manipulated either locally by privileged and un-privileged users, or remotely via network connections that exploit known vulnerabilities or insecure configurations (e.g. accepting unknown/unsigned Active-X components in your browser). It is therefore clear that it is nearly impossible to protect a computer system from software attacks while relying purely on software defences.

To progress beyond this state, the Trusted Computing Group (TCG)[1], a not-for-profit group of ICT industry players, developed a set of specifications to create a computer system with enhanced security named "trusted platform".

A trusted platform is based on two key components: protected capabilities and shielded memory locations. A protected capability is a basic operation (performed with an appropriate mixture of hardware and firmware) that is vital to trust the whole TCG subsystem. In turn capabilities rely on shielded memory locations, special regions where is safe to store and operate on sensitive data.

From the functional perspective, a trusted platform provides three important features rarely found in other systems: secure storage, integrity measurement and reporting. The integrity of the platform is defined as a set of metrics that identify the software components (e.g. operating system, applications and their configurations) through the use of fingerprints that act as unique identifiers for each component. Considered as a whole, the integrity measures represent the configuration of the platform. A trusted platform must be able to measure its own integrity, locally store the related measurements and report these values to remote entities. In order to trust these operations, the TCG defines three so-called "roots of trust", components that must be trusted because their misbehaviour can not be detected:

- the Root of Trust for Measurements (RTM) that implements an engine capable of performing the integrity measurements;
- the Root of Trust for Storage (RTS) that securely holds integrity measures and protect data and cryptographic keys used by the trusted platform and held in external storages;
- the Root of Trust for Reporting (RTR) capable of reliably reporting to external entities the measures held by the RTS.

The RTM can be implemented by a measurement engine (i.e. the CPU) and code. The latter can be the first software module executed when a computer system is switched on (i.e. a small portion of the BIOS firmware) or directly the hardware itself when using processors of the latest generation. The measurement code, which must be trustworthy, is called Core RTM (CRTM).

The central component of a TCG trusted platform is the Trusted Platform Module (TPM) [2]. This is a low cost chip capable to perform cryptographic operations, securely maintain the integrity measures and report them. Given its functionalities, it is used to implement RTS and RTR, but it can also be used by the operating system and applications for cryptographic operations although its performance is quite low.

The TPM is equipped with two special RSA keys, the Endorsement Key (EK) and the Storage Root Key (SRK). The EK is part of the RTR and it is a unique (i.e. each TPM has a different EK) and "non-migratable" key created by the

[1] https://www.trustedcomputinggroup.org

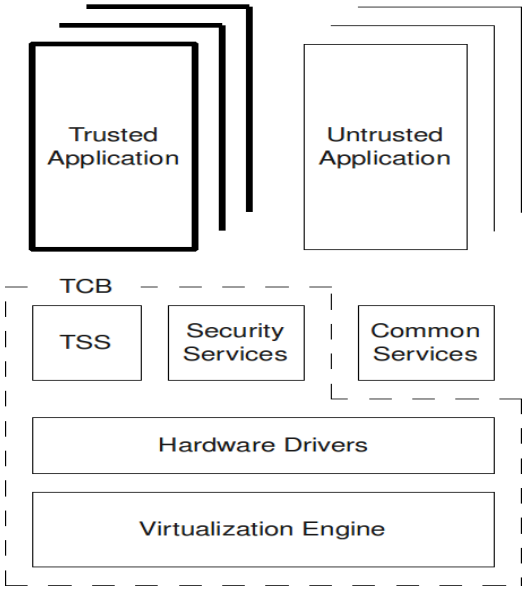

Figure 2: Typical architecture for Trusted Platforms

manufacturer of the TPM and that never leaves this component. Furthermore the specification requires that a certificate must be provided to guarantee that the key belongs to a genuine TPM. The SRK is part of the RTS and it is a "non-migratable" key that protects the other keys used for cryptographic functions[2] and stored outside the TPM. Also SRK never leaves the TPM and it is used to build a key hierarchy. The integrity measures are held into the Platform Configuration Registers (PCR). These are special registers within the TPM acting as accumulators: when the value of a register is updated, the new value depends both on the new measure and on the old value to guarantee that once initialized it is not possible to fake the value of a PCR.

The action of reporting the integrity of the platform is called *Remote Attestation*. A remote attestation is requested by a remote entity that wants evidence about the configuration of the platform. The TPM then makes a digital signature over the values of a subset of PCRs to prove to the remote entity the integrity and authenticity of the platform configuration. For privacy reasons, the EK cannot be used to make the digital signature. Instead, to perform the remote attestation the TPM uses an Attestation Identity Key (AIK), which is an "alias" for the EK. The AIK is a "non-migratable" RSA key created by the TPM whose private part is never released unencrypted outside the chip; this guarantees that the AIK cannot be used by anyone except the TPM itself.

In order to use the AIK for authenticating the attestation data (i.e. the integrity measures) it is necessary to obtain a certificate proving that the key was actually generated by a genuine TPM and it is managed in a correct way. Such certificates are issued by a special certification authority called Privacy CA (PCA). Before creating the certificate, the PCA must verify the genuineness of the TPM. This verification is done through the EK certificate. Many AIKs can be created and, to prevent the traceability of the platform operations, ideally a different AIK should be used for interacting with each different remote attester.

By using trusted computing it is possible to protect data via asymmetric encryption in a way that only the platform's TPM can access them: this operation is called binding. It is however possible to migrate keys and data to another platform, with a controlled procedure, if they were created as "migratable".

The TPM also offers a stronger capability to protect data: sealing. When the user seals some data, he must specify an "unsealing configuration". The TPM assures that sealed data can be only be accessed if the platform is in the "unsealing configuration" that was specified at the sealing time.

The TPM is a passive chip disabled at factory and only the owner of a computer equipped with a TPM may choose to activate this chip. Even when activated, the TPM cannot be remotely controlled by third entities: every operation must be explicitly requested by software running locally and the possible disclosure of local data or the authorisation to perform the operations depend on the software implementation.

In the TCG architecture, the owner of the platform plays a central role because the TPM requires authorisation from the owner for all the most critical operations. Furthermore, the owner can decide at any time to deactivate the TPM, hence disabling the trusted computing features. The identity of the owner largely depends on the scenario where trusted computing is applied: in a corporate environment, the owner is usually the administrator of the IT department, while in a personal scenario normally the end-user is also the owner of the platform.

A Trusted Platform (TP) is a computing system which exploits the features offered by the TPM and the Core RTM to build advanced security properties. Usually a Trusted Platform is composed of different layers (Figure 2):

- the Trusted Computing Base (TCB) is the lower layer that is made up by the TPM, optionally a virtualization engine, the driver of the hardware components, some software service needed to manage the security features and a software layer (TSS, the TCG Software Stack) that allows the interaction with the TPM.
- the trusted applications are in charge of performing critical operations. These applications take advantage of the security services offered by the TCB (for instance, by using the sealing to protect important data such as the cryptographic keys).
- the untrusted applications are common applications whose malfunctioning is not harmful for the system. For these applications there is no need to provide strong security features.

Run-time isolation between software modules with different security requirement (i.e. isolating trusted from untrusted applications) is an interesting complementary requirement for a trusted platform. Given that memory areas of different modules are isolated and inter-module communication can occur only under well specified control flow policies, then if a specific module of the system is

[2] In order to minimize attacks, the SRK is never used for any cryptographic function but only to protect other keys.

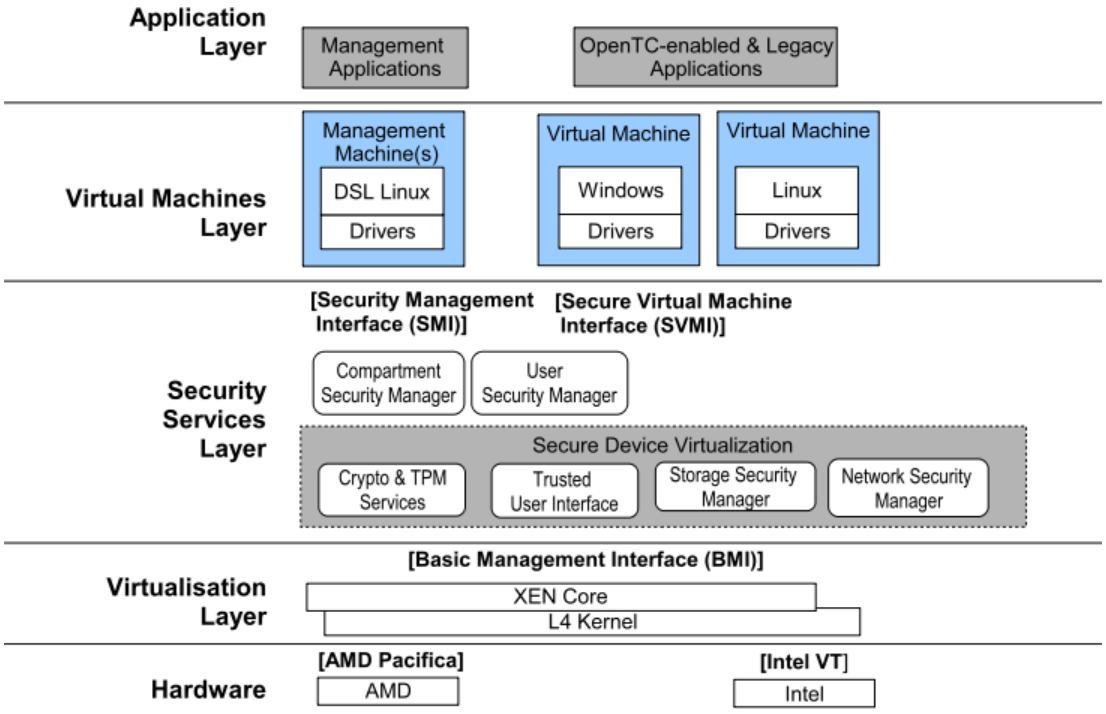

Figure 3: The OpenTC architecture.

compromised (e.g. due to a bug or a virus), the other modules that are effectively isolated from that one are not affected at all. Today virtualization is an emerging technology for PC class platforms to achieve run-time isolation and hence is a perfect partner for a TPM-based trusted platform.

The current TCG specifications essentially focus on protecting a platform against software attacks. The AMD Virtualization (AMD-V)[3] [3] and the Intel Trusted Execution Technology (TXT)[4] [4] initiatives, besides providing hardware assistance for virtualization, increase the robustness against software attacks and the latter also starts dealing with some basic hardware attacks. In order to protect the platforms also from physical attacks, memory curtaining and secure input/output should be provided: memory curtaining extends memory protection in a way that sensitive areas are fully isolated while secure input/output protects communication paths (such as the buses and input/output channels) among the various components of a computer system. Intel TXT focuses only on some so called "open box" attacks, by protecting the slow buses and by guaranteeing the integrity verification of the main hardware components on the platform.

4 The Open Trusted Computing Project

OpenTC[5] is a FP6 EC co-funded project that applied TC techniques to the creation of an open and secure computing environment by coupling them with advanced virtualization techniques. In this way it is possible to create on the same computer different execution environments mutually protected and with different security properties. The overall architecture of the OpenTC environment is shown in Figure 3.

OpenTC uses virtualization layers – also called Virtual Machine Monitors (VMM) or hypervisors – and supports two different implementations: Xen [5] and L4/Fiasco [6]. This layer hosts compartments, also called virtual machines (VM), domains or tasks, depending on the VMM being used. Some domains host trust services that are available to authorised user compartments. Various system components make use of TPM capabilities, e.g. in order to measure other components they depend on or to prove the system integrity to remote challengers. Each VM can host an open or proprietary operating environment (e.g. Linux or Windows) or just a minimal library-based execution support for a single application.

The viability of the OpenTC approach has been demonstrated by creating three proof-of-concept prototypes, the so-called PET, CC@H and VDC ones, that are publicly available at the project's web site.

The PET (for "Private Electronic Transactions") scenario [7] aims to improve the trustworthiness of interactions with remote servers. Transactions are simply performed by accessing a web server through a standard web browser running in a dedicated trusted compartment. The server is assumed to host web pages related to a critical financial service, such as Internet banking or another e-commerce service. The communication setup between the browser compartment and the web server is extended by a protocol for mutual remote attestation tunnelled through an SSL/TLS channel. During the attestation phase, each side assesses the trustworthiness of the other. If this assessment is negative on either side, the SSL/TLS tunnel is closed, preventing further end-to-end communication. If the

[3] http://www.amd.com/virtualization
[4] http://www.intel.com/technology/security/
[5] http://www.opentc.net

assessment is positive, end-to-end communication between browser and server is enabled via standard HTTPS tunnelled over SSL/TLS. In this way the user is reassured about having connected to the genuine server of its business provider and about its integrity, and the provider knows that the user has connected by using a specific browser and that the hosting environment (i.e. operating system and drivers) has not been tampered with, for example by inserting a key-logger.

The CC@H (for "Corporate Computing at Home") scenario [8, 9] demonstrates the usage of the OpenTC solution to run simultaneously on the same computer different non-interfering applications. It reflects the situation where employers tolerate, within reasonable limits, the utilization of corporate equipment (in particular notebooks) for private purposes but want assurance that the compartment dedicated to corporate applications is not manipulated by the user. In turn the user has a dedicated compartment for his personal matters, included free Internet surfing which, on the contrary, is not allowed from the corporate compartment. The CC@H scenario is based on the following main functional components:

- boot-loaders capable of producing cryptographic digests for lists of partitions and arbitrary files that are logged into PCRs of the TPM prior to passing on control of the execution flow to the virtual machine monitor (VMM) or kernel it has loaded into memory;
- virtualization layers with virtual machine loaders that calculate and log cryptographic digests for virtual machines prior to launching them;
- a graphical user interface enabling the user to launch, stop and switch between different compartments with a simple mouse click;
- a virtual network device for forwarding network packets from and to virtual machine domains;
- support for binding the release of keys for encrypted files and partitions to defined platform integrity metrics;
- a trusted channel built upon TLS extensions [10] to let the Corporation remotely attest platform's TCB and the Corporate Compartment.

The last scenario called VDC (for "Virtual Data Centre") extends the concepts of the CC@H scenario to cover the data centre's use cases. In this context, two main stakeholders come into play: data centre customers and administrators.

Data centre customers want to run applications in a remote execution environment while having guarantees that they are properly isolated. Additionally, customers require management support to start, stop and reconfigure their execution environments and allocate (virtual) resources. They also may desire to split security critical and non-critical parts between different virtual domains, with dedicated gateways between them. Finally they may want to provide services to external users (e.g. partners of data centre customers) and therefore it must be possible to leave open access to part of the resources allocated while protecting the rest of the system.

On another hand, data centre administrators need to have an overview of how many Trusted Virtual Domains (TVD) [11] are hosted at the infrastructure, which nodes they are using, and which free resources could be allocated to new and existing customers. Moreover each customer must (be obliged to) adhere to the policy defined by contract without interfering with other customers' policies.

Therefore the goal of the architecture for the Virtual Data Centre is to provide an infrastructure capable of running different virtual machines provided by customers (possibly spreading them over many different physical hardware) while offering separation mechanisms for shared network and storage. The scenario also requires a management infrastructure capable of automated deployment of virtual resources like TVD networks [12] and automated enforcement of data centre's and customers' security policies [13]. Finally management consoles with improved security properties are required for administering multiple disjoint domains in parallel.

5 Using TC to protect CIS

CIS security threats can be grouped into the following three main categories.

T1 (malware). CIS are composed by different software modules running over a distributed computing system. This software can contain viruses, trojans or spyware as well as security holes due to bad programming. Furthermore, as a CIS is typically an heterogeneous system developed by different parties and continuously evolving through years, these types of problem are likely to happen often enough to consider this menace as quite a serious one.

T2 (insiders). CIS are large systems administered by many different subjects over a wide area. In this context it is not unlikely to find some unfaithful operator to damage the CIS or even the underlying physical system. These subjects know enough about the system to be considered a real and powerful threat.

T3 (external attackers). As nowadays common in every networked system, there are attackers that try to gain access to the computing systems from outside. Since CIS often control vital physical systems (e.g. water distribution) in addition to classical motivations (such as fame and profit) terrorism must also be taken into account.

In addition to these threat categories, it is also possible to group the possible attacks into two main categories according to their target: the communications or the platforms.

Network attacks target the communication among the nodes of the CIS. For instance, such attacks can lead to disclosure of data read by the sensors or modification of the control parameters sent to the physical operating equipment, easily provoking the misbehaviour of the CIS system. Moreover, due to poor network configuration, the attackers could sniff passwords if they are transmitted in clear on the network, or monitor the traffic to gain information about the system activity.

Platform attacks are carried out directly against the nodes of the CIS. Local vulnerabilities – such as poor password choice, missing security patches or bad configuration of the

nodes – can be used to gain access to the system or to perform a privilege escalation.

TC can help to provide protection against these threats and attacks and also benefit the general design and operations of a CIS in several ways.

5.1 Node state verification

In a highly distributed and physically unprotected system it is of utmost importance to know if the software image of each node has been tampered with. The remote attestation capability of TC can be used to request and reliably report the software state of a remote node.

This is especially important when several players concur to the design, management, operation, and maintenance of a CIS. Software modules can be unintentionally modified due to human errors, or deliberately modified due to attacks, but both cases will be quickly detected when performing a remote attestation.

Furthermore, TC may force operators to keep up to date the software running on their nodes: network access could be restricted only to those nodes that have installed relevant security patches, while the not updated ones may be either isolated or allowed only to access a limited network area just for urgent administrative operations (such as security updates).

Verification could also take place in a different way: a central management facility could poll the various nodes by opening network connections towards them just to check via remote attestation the software status of the nodes. In this way it would be very easy to detect mis-configured elements and promptly reconfigure or isolate them, leading to a powerful distributed Intrusion Detection System (IDS).

5.2 Trusted authentication

Proper authentication is the first step towards achieving effective access control [16]. Authentication is usually enforced by assigning login accounts to authorized users who can then access the network using their passwords.

However, this is not a strong security control: users easily fall victim of social engineering attacks or their passwords are stolen by malware such as key loggers. Furthermore, it is also possible that the remote system used for authentication is a rogue one.

TC can help by protecting credentials used for authentication and preventing the possibility that a rogue system is used for authenticating. This is done by using sealing to store the access credentials (e.g. username and password) and by verifying the platform integrity prior of providing access to the credentials. In this way an individual can unseal the credentials and gain access to the CIS only if he both provides the correct passphrase and uses a correctly configured and un-tampered system. In this scenario, it is impossible for an attacker to use a CIS user itself as attack vector.

If a stronger authentication mechanism is needed – such as smart card based login [15] – the TPM itself can be used as a smart card to store sensitive data (e.g. the private key of the user), with a higher security level because the key can be used only by a specific program while a stolen smart card can be used on any system and with any application.

5.3 Trusted log

TC opens also the door to reliable logging, where log files contain not only a list of events but also a trusted trace of the component that generated the event and the system state when the event was generated.

This is particularly useful to enforce a non-repudiation policy, to support evidence creation for computer forensics, or to detect false logging events created by an attacker to mess-up the log.

Additionally the log file could be manipulated only by trusted applications if it was sealed against them, so that no direct editing (for insertion or cancellation) would be possible, for example by a mis-behaving administrator.

5.4 Data protection

In general TC helps to protect sensitive data by hardware means. In particular, sealing protects critical data from access by unauthorized software because access can be bound to a well-defined system configuration.

This feature allows the implementation of fine-grained access control schemes that can prevent agents from accessing data they are not authorized for.

Moreover, it blocks the access to those data in the case of an attack to the node. Again, in a large distributed scenario, where it is impossible to guarantee the physical security of each node, this is an essential feature against unauthorized software manipulation of the nodes.

This also becomes very important when several applications with different trust level are running on the same node and maybe these applications have been developed by providers with different duties. In this way we can easily implement the well-known concept of “separation of duties”: even if a module can by-pass the access control mechanisms of the operating system and directly access the data source, it will be unable to operate on it because the TPM and/or security services running on the system will not release to the module the required cryptographic credentials.

5.5 Trusted sensor networks

TC can protect also the sensors used to collect data can in a CIS. While the concept of trusted WSN is quite novel and there is not a specification for TC-compliant hardware for sensors, it is already possible to identify fields where TC can be applied thus enhancing the overall protection. It is important to protect both the sensors’ integrity and the communication between the sensors and the other nodes.

The communication can be protected using standard cryptography. Due to the limited computing resources and the low cost of most sensors, it is likely that a simple protocol based on symmetric cryptography is used. In this context TC will enhance the security by providing means for protecting the secret keys shared between the sensors and the controlling nodes. Where it is possible, the sensors should be equipped with a hardware component that will ensure the protection of the keys. If this is not possible due to limits to the cost of the sensor, the key could be stored on the controlling node and TC will guarantee that communication not protected with such key will be rejected, leading to dropping packets that were possibly sniffed or

crafted by an attacker. On the other side, if the design allows it, the sensor can be provided with TC-oriented hardware components (such as Roots of Trust) that will enable secure storage and integrity measurement and reporting.

Additionally, thanks to the special architecture of the sensors, it is possible to implement promising security features. For instance, recently direct support for the Java programming language has been introduced on many sensors and some of them are capable of running a tiny Java Virtual Machine (JVM) as operating system. The JVM allows a fine granularity control (i.e. it can work at class granularity) which can be used as foundation to develop a fine-grained system. Such system must be capable of enforcing policy on the communication between the classes. This result can be achieved by instrumenting the JVM to selectively block or allow access among the different Java classes instantiated according to the policy. For instance it is possible to separate the security-relevant part of an application (e.g. the code that authenticates the data to send) from the rest (e.g. the user interface). The access control system will be capable of granting access to the secure code if a particular policy applies (e.g. if all the communications happen in a secure form) and to revoke such access otherwise.

Another important benefit that TC would bring to the CIS involves the usage of standard systems [14]. CIS are now in their third evolution phase. After the first monolithic systems composed by mainframes, they evolved to distributed architectures where multiple stations were connected through LAN and shared information in real time. Nowadays their architecture is based on the usage of open system architectures rather than a vendor-controlled proprietary environment and utilizes open standards and protocols thus distributing functionality across a WAN rather than a LAN. In this scenario the usage of standard specifications as the ones proposed by the TCG would help to interconnect in a trusted and standard way the different nodes of the system. Moreover, the usage of virtualization separates the critical parts of the system from the others, allowing use of third-party components and Commercial-Off-The-Shelf (COTS) software.

5.6 Cost and performance issues

Designers and managers of CIS should not be scared by the cost and overhead introduced by TC.

Cost is usually not an issue because – by design – the TPM has been conceived as a low-cost item that nowadays is found both as a separate component and as an embedded element of chipsets (for desktops, servers, and microcontrollers).

When an extremely low cost for a node is necessary (such as in very small wireless sensor nodes) then a different architecture should be used: the TC features should be available at a higher level hierarchical node that controls the sensors and reports only its own integrity. The dialog between the controller and the sensors can be protected with ad-hoc techniques [17, 18, 19]. With this architecture, trust based on TC is limited to the core of the CIS while the leafs should use other non-TC strategies for building trust, such as duplicate measures to thwart attacks against individual sensors.

Cryptographic performance of TPM is low because it is not a general-purpose cryptographic accelerator but just a trust anchor. It implements the SHA and RSA algorithms, as well as some limited secure storage. For desktops and servers this is not a problem as they have plenty of computing power that can be used to perform the massive cryptographic operations needed to protect data and communications. However for small sensor nodes TPM can become a bottleneck if frequently used. In general, CIS designers should be aware of the limited cryptographic power of TPM and therefore require remote attestation only when really needed.

When it comes to trusted software environments built on top of TPM, in our experience the XEN-based environment is powerful but demanding, while the L4/Fiasco set-up is very lightweight and therefore suitable also for nodes with limited computational resources.

If a virtualized trusted environment is used, inside a partition we can execute a full operating system with all its capabilities (e.g. Suse Linux), a stripped-down OS (such as DSL, Damn Small Linux) with only the required drivers and capabilities, or just a mini-execution environment providing just the required libraries for small embedded single task applications.

Finally, in case multiple compartments are not needed, the TC paradigm can be directly built on top of the hardware, without any virtualization layer, hence further reducing its footprint.

6 Conclusions

While some technical problems are still to be solved before large-scale adoption of TC is a reality, we nonetheless think that it is ready to become a major technology of the current IT scenario. This is true not only for general purpose ICT systems but especially for critical information systems where the TC advantages in terms of protection and assurance far outweigh its increased design and management complexity.

Acknowledgment

This work has been partially funded by the EC as part of the OpenTC project (ref. 027635). It is the work of the authors alone and may not reflect the opinion of the whole project.

References

[1] D. Kuhlmann, R. Landfermann, H.V. Ramasamy, M. Schunter, G. Ramunno; D. Vernizzi, “An Open Trusted Computing Architecture - Secure virtual machines enabling user-defined policy enforcement”, *IBM Research Report*, Computer Science, RZ 3655 (# 99675), August 2006

[2] Trusted Computing Group, TPM Specification Part 1-3, Version 1.2 Level 2 Revision 103, July 2007

[3] G. Strongin, “Trusted computing using AMD ‘Pacifica’ and ‘Presidio’ secure virtual machine

technology", *Information Security Technical Report*, Elsevier, Volume 10, Issue 2, 2005, pp. 120-132,

[4] D. Grawrock, "Dynamics of A Trusted Platform - A building block approach", Intel Press, 2009

[5] P.T. Barham, B. Dragovic, K. Fraser, S. Hand, T.L. Harris, A.Ho, R.Neugebauer, I.Pratt, A.Warfield, "Xen and the Art of Virtualization", *SOSP-2003: ACM Symposium on Operating Systems Principles*, October 19-22, 2003, Bolton Landing (NY USA), pp. 164-177

[6] M. Hohmuth, "The Fiasco Kernel: Requirements Definition", *TU Dresden Technical Report*, TUD-FI98-12, December 1998

[7] S. Lo Presti, G. Ramunno, D. Kuhlmann, "Private Electronic Transactions - The OpenTC proof-of-concept prototype", *The OpenTC newsletter*, no. 3, January 2008

[8] D. Weber, A. Weber, "'Corporate Computing at Home' – the scenario of the second OpenTC proof-of-concept prototype", *The OpenTC newsletter*, no. 5, March 2008

[9] D. Kuhlmann, "A quick walkthrough of the second OpenTC proof-of-concept prototype", *The OpenTC newsletter*, no. 5, March 2008

[10] F. Armknecht, Y. Gasmi, A.R. Sadeghi, P. Stewin, M. Unger, G. Ramunno, D. Vernizzi, "An efficient implementation of trusted channels based on Openssl", *STC'08: ACM workshop on Scalable Trusted Computing*, Nov 1, 2008, Alexandria, (VA USA), pp. 41-50

[11] J. L. Griffin, T. Jaeger, R. Perez, R. Sailer, L. van Doorn, R. Caceres, "Trusted Virtual Domains: Toward secure distributed services" HotDep 2005: *IEEE Workshop on Hot Topics in System Dependability*, June 30, 2005, Yokohama (Japan)

[12] S. Cabuk, C.I. Dalton, H.V. Ramasamy, M. Schunter, "Towards automated provisioning of secure virtualized networks", *CCS '07: ACM conference on Computer and communications security*, Oct 29 – Nov 2, 2007, Alexandria, (VA USA), pp. 235–245

[13] S. Cabuk, C.I. Dalton, K. Eriksson, D. Kuhlmann, H.V. Ramasamy, G. Ramunno, A.R. Sadeghi, M. Schunter, C. Stueble, "Towards automated security policy enforcement in multi-tenant virtual data centers", *Journal of Computer Security*, IOS Press, Special Issue on EU FP6 Projects, 2009, pp. 1-33

[14] C.L. Bowen, T.K. Buennemeyer, R.W. Thomas, "Next generation SCADA security: best practices and client puzzles", *IAW'05: IEEE SMC Information Assurance Workshop 2005*, June 15-17, 2005, West Point (NY, USA), pp. 426-427

[15] T. Sauter, C. Schwaiger, "Achievement of secure Internet access to fieldbus systems", *Microprocessors and Microsystems*, Elsevier, vol. 26, no. 7, September 2002, pp. 331-339

[16] V.M. Igure, S.A. Laughter, R.D. Williams, "Security issues in SCADA networks", *Computers & Security*, Elsevier, October 2006, vol. 25, no. 7, pp. 498-506

[17] K. Kifayat, M. Merabti, Q. Shi, D. Llewellyn-Jones, "Group Based Secure Communication for Large-Scale Wireless Sensor Networks", *Journal of Information Assurance and Security*, Dynamic Publishers, September 2007, vol. 2, no. 2, pp. 139-149

[18] A. Sorniotti, L. Gomez, K. Wrona, L. Odorico, "Secure and Trusted in-network Data Processing in Wireless Sensor Networks: a Survey", *Journal of Information Assurance and Security*, Dynamic Publishers, September 2007, vol. 2, no. 3, pp. 189-199

[19] M. AL-Rousan, A. Rjoub, A. Baset, "A Low-Energy Security Algorithm for Exchanging Information in Wireless Sensor Networks", *Journal of Information Assurance and Security*, Dynamic Publishers, March 2009, vol. 4, no. 1, pp. 48-59

Trusted Computing Glossary

AIK (Attestation Identity Key)
Special RSA key used by the TPM for an operation of remote attestation.

DAA (Direct Anonymous Attestation)
Cryptographic privacy-friendly protocol used in TC to provide integrity evidence without disclosing the user identity.

EK (Endorsement Key)
Special RSA key used to asses the genuineness of a TPM.

MTM (Mobile Trusted Module)
Security element based on TPM specifications and used as basis for TC on embedded and mobile platforms.

PCA (Privacy Certificate Authority)
Special certification authority that verifies the genuineness of a TPM and issues a certificate for an AIK belonging to that TPM.

PCRs (Platform Configuration Registers)
Special memory regions managed by the TPM and used to securely store the integrity measurements.

TCB (Trusted Computing Base)
Coupling of TPM, critical software and security-oriented services.

TCG (Trusted Computing Group)
Not-for-profit group that released the specifications of TC and manages their maintenance.

TP (Trusted Platform)
Computing system composed by a TPM and software modules implementing TC-oriented features.

TPM (Trusted Platform Module)
Low-cost cryptographic chip used as basis for TC.

RA (Remote Attestation)
Act of reporting the integrity measurements to an external entity.

RoT (Root of Trust)
Hardware components that must be trusted since their mis-behaviour cannot be detected.

RTM (Root of Trust for Measurement)
Takes trustworthy integrity measurements that describe the system.

RTR (Root of Trust for Reporting)
Reports the integrity measurements to an external entity.

RTS (Root of Trust for Storage)
Securely stores critical data.

SRK (Storage Root Key)
Special RSA key that protects the other keys used for cryptographic functions.

Author Biographies

Antonio Lioy received a MSc (1982) in electronic engineering "summa cum laude" and a PhD (1987) in computer engineering, both from the Politecnico di Torino. He is Full Professor at the Politecnico di Torino, where he leads the TORSEC research group active in the area of information systems security. His research interests are in the fields of network security, PKI, and policy-based system protection.

Gianluca Ramunno is a researcher in the security group of Politecnico di Torino, where he received his MSc (2000) in electronic engineering and PhD (2004) in computer engineering. His initial research interests were in the fields of digital signature, e-documents, and time-stamping, where he performed joint activity within ETSI. Since 2006 he has been investigating the field of Trusted Computing, leading the Politecnico di Torino activities in this area within the EU FP6 project OpenTC.

Davide Vernizzi received a MSc (2005) in computer engineering jointly from the École Nationale Supérieure d'Informatique et de Mathématiques Appliquées of Grenoble and Politecnico di Torino. He is currently involved in his PhD in computer engineering at Politecnico di Torino. His research interests are mainly focused on Trusted Computing, trusted networks and privacy-related issues. Since 2006 he has been involved in the EU FP6 project OpenTC.