Trusted Computing — Special Aspects and Challenges

Ahmad-Reza Sadeghi

Horst Görtz Institute for IT Security, Ruhr-University Bochum, Germany
sadeghi@crypto.rub.de

Abstract. The advent of e-commerce, e-government, and the rapid expansion of world-wide connectivity demands end-user systems that adhere to well-defined security policies. In this context Trusted Computing (TC) aims at providing a framework and effective mechanisms that allow computing platforms and processes in a distributed IT system to gain assurance about each other's integrity/trustworthiness. An industrial attempt towards realization of TC is the initiative of the Trusted Computing Group (TCG), an alliance of a large number of IT enterprises. The TCG has published a set of specifications for extending conventional computer architectures with a variety of security-related features and cryptographic mechanisms. The TCG approach has not only been subject of research but also public debates and concerns. Currently, several prominent academic and industrial research projects are investigating trustworthy IT systems based on TC, virtualization technology, and secure operating system design.

We highlight special aspects of Trusted Computing and present some current research and challenges. We believe that TC technology is indeed capable of enhancing the security of computer systems, and is another helpful means towards establishing trusted infrastructures. However, we also believe that it is not a universal remedy for all of the security problems we are currently facing in information societies.

1 Introduction

The increasing global connectivity, distributed applications and digital services over the Internet, both for business and private use, require IT systems that guarantee confidentiality, authenticity, integrity, privacy, as well as availability. Examples include online banking, e-commerce and e-government services, content delivery, etc. Modern cryptography and information security provide a variety of very useful technical security measures such as encryption and strong authentication mechanisms to achieve these security targets. However, these measures provide only partial solutions as long as the underlying computing platforms still suffer from various security problems in hardware and software: Beside architectural security problems and the inherent vulnerabilities resulting from high complexity, common computing platforms require careful and attentive system administration skills, and are still unable to effectively protect against execution of malicious code and tampering.

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 98–117, 2008.

In this context some fundamental and challenging issues are how to define and to determine/verify the integrity/trustworthiness[1] of a computing platform or in general of an IT system, and how could common computing platforms support such functionalities. Note that even a perfectly secure operating system cannot verify its own integrity. Today, the authorized access to online services or data over the Internet is controlled by means of secure channel protocols where the standard approaches are security protocols like Transport Layer Security (TLS) [14] or Internet Protocol Security (IPSec) [25]. However, these protocols do not provide any guarantees regarding the integrity/trustworthiness of the communication endpoints. Security breaches on the Internet today rarely involve compromising the secure channel because endpoints are much easier to compromise since they are frequently subject to attacks by malware (e.g., viruses and Trojan horses). Using a secure channel to an endpoint of unknown integrity is ultimately futile.[2]

Hence, reliable mechanisms are desired that allow to verifiably reason about the "trustworthiness" of a peer endpoint (local or remote), i.e., whether its state (the hardware and software configuration) conforms to a defined security policy.

A recent industrial initiative towards this goal is put forward by the *Trusted Computing Group* (TCG), a consortium of a large number of IT enterprises, which proposes a new generation of computing platforms that employs both, supplemental hardware and software. The claimed goal of this architecture is to improve the security and the trustworthiness of computing platforms (see, e.g., [35,42]). The TCG has published several specifications on various concepts of trusted infrastructures [54].[3] The core component the TCG specifies is the Trusted Platform Module (TPM). The current widespread implementation of the TPM is a small tamper-evident cryptographic chip [57,53]. Many vendors already ship their platforms with TPMs (mainly laptop PCs and servers). To this end, the conventional PC architecture is extended by new mechanisms to (i) protect cryptographic keys, (ii) authenticate the configuration of a platform (attestation), and (iii) cryptographically bind (sensitive) data to a certain system configuration (sealing), i.e., the data can only be accessed (unsealed) if the corresponding system can provide the specific configuration for which the data has been sealed.

Trusted Computing technology was subject of public debates due to its claimed capabilities, in particular, in conjunction with Digital Rights Management (DRM) (see, e.g., [3,16,36]). Concerns were aroused that TC technology may give

[1] "Trust" is a complicated notion that has been studied and debated in different areas (social science and humanities as well as computer science). A possible definition of the notion "trustworthy" is the degree to which the (security) behavior of the component is demonstrably compliant with its stated functionality (e.g., see [7]).

[2] In the words of Gene Spafford, *"Using encryption on the Internet is the equivalent of arranging an armored car to deliver credit card information from someone living in a cardboard box to someone living on a park bench."*[50].

[3] Claimed role is to develop, define and promote open, vendor-neutral industry specifications for trusted computing including hardware building blocks and software interface specifications across multiple platforms and operating environments.

vendors and content providers new abilities to get control over personal systems and users' private information. Hence they may allow commercial censorship, political censorship, and product discrimination or undermine the General Public License (GPL [18]).

Meanwhile Trusted Computing has attracted many researchers and developers, and the capabilities as well as the shortcomings of the TCG proposals are better understood. Currently, several prominent research and industrial projects are investigating open trustworthy computing platforms (PC, servers, embedded) and applications based on Trusted Computing, virtualization technology and secure operating system design[4]. Since the body of related literature in this area has grown rapidly, and we have limited space, we only highlight some special aspects of the TC technology, and present some current research challenges and proposals made so far to tackle some shortcomings of the TCG specifications. Hence, we do not claim completeness and apologize for not referring to some of the works done in this area.

2 Main Aspects of the TCG Specification

2.1 Core Components and Functionalities

The main components of the TCG approach are a hardware component called *Trusted Platform Module* (TPM), a kind of (protected) pre-BIOS (Basic I/O System) called the *Core Root of Trust for Measurement* (CRTM), and a support software called *Trusted Software Stack* (TSS), which has various functions like providing a standard interface for TPMs of different manufacturers communicating with the rest of the platform or with remote platforms.

The TPM is the main component of the specification and provides a secure random number generator, non-volatile tamper-resistant storage, key generation algorithms, cryptographic functions like RSA encryption/decryption, and the hash function SHA-1 (see Figure 1). The TPM protects a variety of keys. Two of its main keys are the endorsement key (EK), an encryption key that uniquely identifies each TPM, and the Storage Root Key (SRK) or Root of Trust for Storage (RTS), uniquely created inside the TPM. The private SRK never leaves the TPM, and it is used to encrypt all other keys created by the TPM. The TPM state contains further security-critical data shielded by the TPM. Amongst them is a set of registers called *Platform Configuration Registers* (PCR) that can be used to store hash values. The hardware ensures that the value of a PCR register can only be modified as follows: $PCR_{i+1} := \text{SHA1}(PCR_i|I)$, with the old register value PCR_i, the new register value PCR_{i+1}, and the input I (e.g. a SHA-1 hash value). This process is called *extending* a PCR. Hash values computed during this process are called *measurements* in TCG terminology.

The TCG has published two specifications versions 1.1b [57] and 1.2 [53] for the TPM where the latter has more and also improved functionalities. In particular TPM 1.2 provides at least 4 concurrent monotonic counters and privacy

[4] See, e.g., EMSCB project (`www.emscb.org`) and OpenTC (`www.opentc.net`)

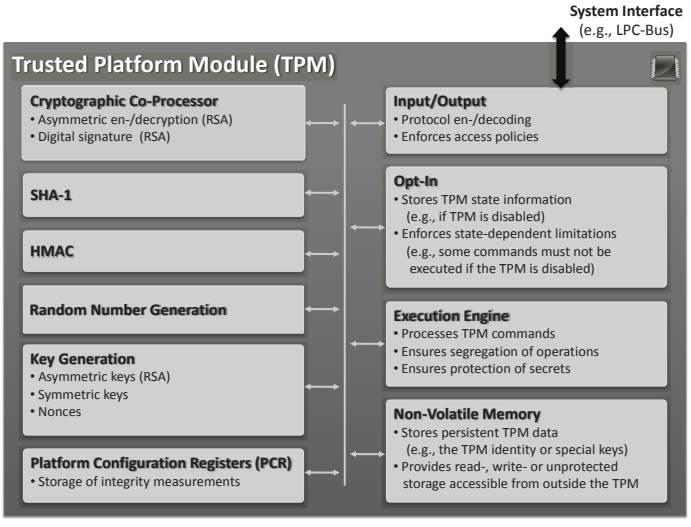

Fig. 1. TPM Architecture

enhanced protocols such as *Direct Anonymous Attestation* (DAA) [9]. Based on TPM functionalities, the TCG specification defines four mechanisms called *integrity measurement*, *attestation*, *sealing* and *maintenance* which are explained briefly in the following:

Integrity Measurement & Platform Configuration. Integrity measurement is done during the boot process by computing the hash value of the initial platform state. For this purpose the CRTM computes a hash of (*measures*) the code and parameters of the BIOS and extends the first PCR register by this result. A chain of trust is established if an enhanced BIOS and bootloader also measure the code they are transferring control to, e.g., the operating system. The security of the chain relies strongly on explicit security assumptions about the CRTM. The state of the PCRs is also called the platform's *configuration*.

Attestation. The TCG attestation protocol is used to give assurance about the platform configuration to a remote party. To guarantee integrity and freshness, PCR values and a fresh nonce provided by the remote party are digitally signed with an asymmetric key called *Attestation Identity Key* (AIK), which is under the sole control of the TPM. A trusted third party called *Privacy Certification Authority* (Privacy-CA) is used to guarantee the pseudonymity of the AIKs. In order to overcome the problem that the Privacy-CA can link transactions to a certain platform (e.g., by means of the same AIK), version 1.2 of the TCG specification defines a cryptographic protocol called *Direct Anonymous Attestation* (DAA) [9], a cryptographic credential scheme that aims at providing anonymity of the platform and unlinkability of remote authentication transactions of the TPM. DAA can be used to sign an AIK, or PCR values.

Sealing/Binding. Data can be cryptographically bound/encrypted to a certain platform configuration by the sealing operation of the TPM. The unseal operation releases the decrypted data only if the current configuration (software and/or hardware) equals the configuration, which has been defined when the data was sealed. Binding is like conventional asymmetric encryption where encrypted data can only be recovered by the TPM that knows the corresponding secret key (no platform configuration check is required).

Secure and Authenticated Boot. The former means that a system terminates the boot process in case the integrity check (e.g., comparing the system measurements with a securely stored reference value) fails, whereas the latter aims at proving the platform integrity to a (remote) verifier.

Maintenance. Maintenance functions can be used to migrate the SRK to another TPM: The TPM owner can create a maintenance archive by encrypting the SRK under a public key of the TPM vendor and an own secret key. In case of a hardware error the TPM vendor can load the SRK from the maintenance archive into another TPM. Currently the maintenance function is still optional and, to our knowledge, not implemented in the currently available TPMs. Moreover, the maintenance function works only for TPMs of the same vendor.

2.2 Trust Model and Assumptions

The TCG defines a component or a system as "trusted" *if it always behaves in the expected manner for the intended purpose*. Note that this definition differs subtly from the mostly used definition that a system or component is "trusted" *if its failure breaks the security policy of the system* (see, e.g., [2]). This definition requires the number of trusted components in a system, also called Trusted Computing Base (TCB), to be as small as possible.

The correctness and soundness of the TCG proposed functionalities are based on some assumptions, which we briefly discuss below. Building systems that can reasonably satisfy these assumptions in practice is a technical challenge and subject of ongoing research, which we will shortly consider in the subsequent sections.

- The platform configuration cannot be forged after the corresponding hash values are computed and stored in the TPM's PCR registers. Otherwise no reliable integrity reporting is possible. Note that currently available operating systems such as Windows or Linux can easily be modified, e.g., by exploiting security bugs or by changing memory which has been swapped to a hard disk. Hence, one requires an appropriate secure operating system design methodology.
- A verifier can determine the trustworthiness of the code from digests (hash) values. Note that today's operating systems and application software are extremely complex, making it very difficult, if not impossible, to determine their trustworthiness. Hence, in practice one requires more effective, reliable, and realistic methodology for this purpose than relying on hash values of binary codes.

– Secure channels can be established between hardware components (TPM and CPU), e.g., when both components are on the same chipset. Note that currently TPM chips are connected to the I/O system with an unprotected interface that can be eavesdropped and manipulated easily [28]. Secure channels to remote parties can be established based on cryptographic mechanisms and a Public Key Infrastructure (PKI).

The main hardware-based components, CRTM and TPM, are assumed to be trusted by all involved parties. Currently the CRTM is not contained in a tamper-resistant module. It should be noted that the TCG specification requires protection of these components only *against software attacks*. Nevertheless, certain hardware attacks may defeat the security of the TCG approach. Some TPM manufacturers have already started a third party certification of their implementation with respect to security standards (Common Criteria [13]) to assure a certain level of tamper-resistance. Although an integration of the TPM functionality into chipsets makes the manipulation of the communication link between TPM and the CPU significantly more difficult and costly, it also introduces new challenges since an external validation and certification of the TPM functionalities (e.g., required by some governments) will be much more difficult.

Even though the TCG approach explicitly allows an application to distinguish between different TPM implementations, the trade-off between costs and tamper-resistance, which is certainly application dependent, will finally determine the level of tamper-resistance.

2.3 Trusted Network Connect (TNC)

The specification for Trusted Network Connect (TNC) [52] has been published by the TNC working group of the TCG. It should be a vendor independent network standard that enhances network security by combining network access control with Trusted Computing. The goal is to integrate the concepts of Trusted Computing with existing network access control mechanisms.

The overall goal of TNC is to prevent compromise of the hosts that connect to a network or other network resources and thus the network itself. The specification suggests platform authentication through a proof of identity in combination with the integrity status of the platform that wants to connect to a network. Therefore network access control is based on extended attributes like platform authentication (e.g., by using an AIK), endpoint compliance, or software state information, which are collected and attested to a verifier. Based on this information the verifying instance is able to decide whether it is secure to extend the network to that platform.

2.4 Mobile Trusted Module (MTM)

The Mobile Trusted Module is a technology that can serve as an integrity control mechanism for protecting non-discretionary services in embedded devices. It adds features to the baseline TPMs in the domain of secure booting. In case

of mobile or embedded platforms, one uses the notion *Mobile Trusted Module* (MTM) [55,56]. Although an MTM has similar features as a common TPM, there are some differences. The Mobile Trusted Module specification defines two types of MTMs: the Mobile Remote-Owner Trusted Module (MRTM) and the Mobile Local-Owner Trusted Module (MLTM). The difference between them is that the MRTM must support mobile-specific commands defined in the MTM specification as well as a subset of the TPM 1.2 commands. Typically, phone manufacturers and network service providers use an MRTM. These parties only have remote access to the MTM whereas the MLTM is used by the user who has physical access to the device and his applications. The different parties, called stakeholders, have different requirements on the integrity, device authentication, SIM Lock/device personalization, secure software download, mobile ticketing and payment, user data protection, privacy issues and more. How these different types of MTMs are implemented is not defined since the specification published by the TCG is quite vague.

3 Property-Based Attestation and Sealing

Integrity verification of applications and their underlying Trusted Computing Base (TCB) is especially important in the context of security policies enforced in a distributed system. Here, remote integrity verification mechanisms should enable a remote party to verify whether an application *behaves* according to certain security policies.

The TCG solution for remote integrity verification are mechanisms called remote *binary attestation*, remote *binary binding*, and *binary sealing*. Although binding and sealing are two slightly different concepts, in the following only the term binding is used for simplicity. Nevertheless, the concepts of property-based binding can also be applied to the sealing functionality.

Loosely speaking, binary attestation and binary binding are based on (i) a measurement of the chain of executed code using a cryptographic digest and (ii) some trust assumption (see Section 2.2). However, TCG proposals have some shortcomings: (i) they reveal the information about the platform's hardware and software configuration and thus make fingerprinting attacks on the platform much easier (*security and privacy*), (ii) they allow remote parties to exclude certain system configurations (*discrimination*), e.g., a content provider can collaborate with an operating system vendor to allow only the software of that vendor to download certain content, (iii) data bound to a certain configuration is inaccessible after any update in firmware or software, or after hardware migrations (*data availability*), and (iv) the verifier is required to know all possible trusted configurations of all platforms (*scalability*).

A more general and flexible extension to the binary attestation is *property-based attestation* [41,38,27]: on higher system levels, attestation should only determine whether a platform configuration or an application has a desired property. Property-based attestation/binding should determine whether the target

machine to be attested fulfills certain requirements (e.g., provides certain access control methods). This avoids revealing the concrete configuration of software and hardware components. For example, it would not matter whether Web browser A or B is used, as long as both have the same properties. For most practical applications, the verifier is not really interested in the specific system or application configuration. This is even disadvantageous for the verifier since he has to manage a multitude of possible configurations. Basically, properties change rarely compared to binaries on program updates.

Some proposals in the literature consider the protection and prove the integrity of computing platforms in the context of secure and authenticated (or trusted) boot (see, e.g., [4], [15], [45], [49], [58]). A high-level protocol for property-based attestation is presented in [38]. The solution is based on property certificates that are used by a verification proxy to translate binary attestations into property attestations. In [41] the authors propose and discusses several protocols and mechanisms that differ in their trust models, efficiency, and the functionalities offered by the trusted components. In particular, [41] discusses how the TSS, the TPM library proposed by the TCG, can provide a property-based attestation protocol based on the existing TC hardware without a need to change the underlying trust model. Another refinement of this idea is proposed in [27]. Moreover, based on ideas of [41], [12] proposes a cryptographic zero-knowledge protocol for anonymous property-based attestation.

In [24] the authors propose *semantic remote attestation* using language-based trusted virtual machines (VM) to remotely attest high-level program properties. The general idea is to use a trusted virtual machine (TrustedVM) that verifies the security policy of the machine that runs within the VM.

In [31], [34] and [33] the authors propose a software architecture based on Linux providing attestation and binding. The architecture binds short-lifetime data (e.g., application data) to long-lifetime data (e.g., the Linux kernel) and allows access to that data only if the system is compatible to a security policy certified by a security administrator.

However, many challenges remain to be solved and are subject of ongoing research: how to define useful properties applicable for practice, how to build efficient mechanisms to determine properties of complex and composed systems, and how to formally capture this notion.

4 Secure Data Management

The integrity measurement mechanism securely stores the platform's initial configuration into the registers (PCRs) of the TPM. Any change to the measured software components results in changed PCR values, making sealed data inaccessible under the changed platform configuration. While this is desired in the case of an untrustworthy software suite or malicious changes to the system's software, it may become a major obstacle for applying patches or software updates. Such updates do generally not change the mandatory security policy enforced by an

operating system (in fact, patches *should* close an existing security weakness not included in the system specification). Nevertheless, the altered PCR values of the operating system make the sealed information unavailable under the new configuration. As mentioned in Section 3 the semantic of the sealing operation is too restrictive to efficiently support sealed information through the software life-cycle including updates/patches. The main problem is the lack of a mapping between the security properties provided by a platform configuration and its measurements. This difficulty is also pointed out in [42]. A further problem with the TCG's proposal is how to handle hardware replacements in a computing platform. Such replacements are necessary due to outdated or faulty hardware. In corporate contexts, hardware is typically replaced every few years. Any sealed data bound to a given TPM cannot directly be transferred to another TPM, because it is encrypted with a key protected by the SRK, which in turn is stored within the TPM. Further, the maintenance function does not allow platform owners to migrate to a TPM of a different vendor.

In [26] the authors address these problems and propose possible solutions for the secure migration, maintenance, and a more flexible sealing procedure. They also use the ideas on property-based attestation [41,38] (see also Section 3) to construct property-based sealing. However, some of these solutions require changes to the TPM specification and consequently to the TPM firmware. The future versions of the TPM specification may integrate some of these ideas. The challenge is, however, to reduce the TPM complexity but still have appropriate solutions for the mentioned problems above.

In [45] the authors present an integrity measurement architecture (IMA) for Linux (see also Section 8). However, platform updates and migration are not addressed, and, as the PCR values are employed to protect the current list of measurements, working with sealed data seems to be difficult.

5 Trusted Channels — Beyond Secure Channels

The standard approach for creating secure channels over the Internet is to use security protocols such as Transport Layer Security (TLS) [14] or Internet Protocol Security (IPSec) [25], which aim at assuring confidentiality, integrity, and freshness of the transmitted data as well as authenticity of the involved endpoints. However, as mentioned before, secure channels do not provide any guarantees about the integrity of the communication endpoints, which can be compromised by viruses and Trojans. Based on security architectures that deploy Trusted Computing functionalities (see also Section 10), one can extend these protocols with integrity reporting mechanisms (either binary or property-based attestation), as proposed in [21] for the case of TLS (see also related work in [21]).

In this context, an interesting issue would be to analyze how to extend other cryptographic protocols and mechanisms (e.g., group based cryptography like group key exchange) with integrity reporting and binding/sealing mechanisms

and all this under the weakest possible assumptions. Besides, the underlying security architecture should be capable of handling configuration changes and its validation in run time environment and be able to manage the corresponding access control, e.g., to generate new session keys. However, efficient and effective run time attestation remains an open problem.

6 Compliance and Conformance

The Trusted Platform Module (TPM) acts as the root of trust and is the basis for all other specifications of the TCG. Vendors already deploy computer systems, e.g., laptop computers, that are equipped with a TPM chip. The TPM is a basic but nevertheless very complex security component. Its specifications are continuously growing in size and complexity (120 commands and up to 19 command parameters in TPM v1.2), and there is still no published analysis on the minimal TPM functionalities that are practically needed. In addition to this, TPM users have to completely trust implementations of TPM manufacturers regarding the compliance but also conformance (security) to the TCG specification. This also requires the user to trust the TPM implementation that no malicious functionalities have been integrated (trapdoors or Trojan horses). Finally, the TCG adversary model considers software attacks only (see also 2.2).

Due to the complexity of the TPM specifications, one expects that not all TPM chips operate exactly as specified. In practice, different vendors may implement the TPM differently. They may exploit the flexibility of the specification or they may deviate from it by inappropriate design and implementation[5].

In [40], the authors introduce a prototype test suite developed for TPM compliance tests. Based on the test results, they point out the non-compliance and bugs of many TPM implementations currently available at the market. They present a testing strategy, some sample test results, and an analysis for different TPM implementations of different manufacturers, and discuss how one can construct attacks by exploiting non-compliance and deficiencies of protection mechanisms. However, their test suite does not cover the entire TPM specifications.

In the recent years many efforts have been invested into thorough analysis of cryptographic algorithms and security protocols resulting in a variety of methods and automatic tools. Security models and manual security proofs up to formal and automatic verification have been developed for this purpose, leading to a deeper understanding of proposed algorithms and protocols. However, not much is publicly known whether the same has been done for the TPM specification. While [40] focuses mainly on TPM compliance issues, [23] presents the results of an automated verification of some generic scenarios based on TPM functionalities

[5] One may argue that vendors can deviate from the TPM specification because the TCG does not enforce its brand name strongly enough. However, the TCG specification itself has a certain degree of flexibility which may be exploited. This is also the case with many other standards. Pushing a brand name or logo “TCG inside” may not be sufficient in general given the complexity of the TPM.

and identifies some security problems and inconsistencies which have been adopted into the recent version of the specification.

7 Virtual TPM

Virtualization [17,22] is a very useful technology that allows the cost-effective use of hardware and resource sharing. In the recent years virtualization technology is enjoying its rediscovery. Virtualization allows to run several virtual machines (VM) (e.g., several operating systems) on top of the same hardware platform, move the VMs among different platforms etc. *Virtual Machine Monitors* (VMM), or *hypervisors*, acting as a control instance between virtual machines and physical resources, provide isolation of processes. However, for security-critical applications where confidentiality and integrity of data are required, one needs mechanisms that assure the integrity of the underlying software (VMs and the virtualization layer), or that these components conform to the defined security policy. Combining VMM with Trusted Computing functionalities based on a hardware-based root of trust (e.g., TPM) can provide such mechanisms under specific assumptions (see also Sections 2.2 and 10).

In this context TPM virtualization makes TPM's capabilities available to multiple virtual machines running on the platform. However, a virtual (software) TPM (vTPM) underlies a different trust model than a hardware TPM and hence, there should be a secure link between each virtual TPM and the unique hardware TPM. This is a challenging task in particular with regard to migration of vTPM instances among different platforms that may have a different level of trust and security requirements. Moreover, the state of a virtual TPM shall not be subject to manipulation or cloning.

In [8] the authors propose an architecture where all vTPM instances are executed in one special VM. This VM also provides a management service to create vTPM instances and to multiplex the requests. Optionally, the vTPM instances may be realized in a secure co-processor card. In their approach, migration of vTPM instances is realized through an extension of the TPM command set. When migration of a vTPM is requested, the state of the vTPM is encrypted with a symmetric key that itself is encrypted by a migratable key of the real TPM. On the destination site, the symmetric key and then the state of the vTPM are decrypted and the vTPM instance resumes.

GVTPM [39] is a virtual TPM framework that supports various TPM models and even different security profiles for each VM under the Xen hypervisor. In contrast to providing fully virtualized TPM functionality, [46] virtualizes only one functionality of a TPM 1.2, namely monotonic counters.

However, one still needs more flexible architectures for secure management of vTPMs that among others (i) do not require extensions, and consequently more complexity for TPMs, (ii) are less dependent on binary hash-based measurements to properly manage updates and resealing, and (iii) can migrate VMs and their associated vTPMs to platforms with different security policies and different binary implementations of the underlying virtual machine monitor (VMM).

8 Integrity Measurement

AEGIS [4] performs an integrity check during the boot process of the whole operating system. It protects the integrity reference values by building a chain of trust and protecting the root reference value by special hardware. Enforcer [31] is a security module for the Linux kernel, which works as an integrity checker for file systems. It uses a TPM to verify the integrity of the boot process and to protect the secret key of an encrypted file system. A certain configuration of a system can be proved by comparing the values inside the TPM register to previously computed reference values. If data or applications are sealed with these values, the integrity of the underlying platform is indirectly assured. [45] introduces IMA, an integrity measurement architecture for Linux. It extends the Linux kernel and inserts measurement hooks in functions relevant for loading executable code. In this way, they extended the measurement chain from the BIOS and bootloader to the application level. However, frequent changes in application files on a running system, e.g., due to updates and patches, steadily increase the measurement list and may become impractical. Remote parties can first verify the integrity of the table of measurements using remote attestation, and can then decide if the current platform configuration is trustworthy.

9 New Processor Generation

Hardware vendors have improved both, the security features provided by the CPU and the appropriate chipsets by the following features.[6] First, a new CPU mechanism to protect security-critical code and data from untrusted code, e.g., an existing operating system. This allows a security kernel to be executed in parallel to an existing legacy operating system. Second, a CPU secure startup-technique based on TPM functionality that allows to load security critical code dynamically, i.e., after untrusted code has already been loaded. This functionality, called `SKINIT` on AMD Pacifica and `SENTRY` on Trusted Execution Technology, is an alternative realization of the CRTM that does neither require a tamper-resistant BIOS extension nor modifications of the bootloader. Third, an extension of the mainboard chipset allows to prevent DMA-enabled devices from accessing security-critical data [47,11].

10 Security Architectures — Possible Approach

The basic desired primitives required for a trustworthy IT system are (i) means for integrity verification that allow a computing platform to export verifiable information about its properties (comes from the requirement of assuring the executing image and environment of an application located on a remote computing platform), (ii) secure storage that allows applications to persist data

[6] Intel's Trusted Execution Technology formerly known as LaGrande (see `http://www.intel.com/technology/security`) and AMD's Virtualization Technology formerly code-named Pacifica (see `http://www.amd.com/virtualization`)

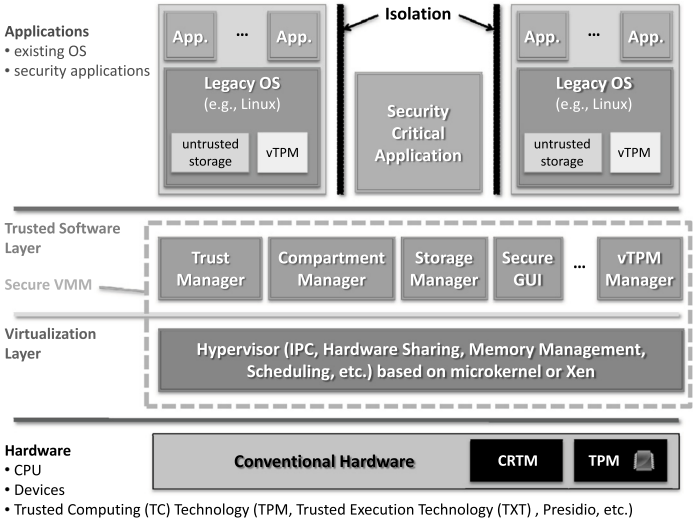

Fig. 2. Possible Security Architecture

securely between executions using traditional untrusted storage, (iii) strong process isolation, and assuring (memory space) separation between processes, (iv) secure I/O to assure that users securely interact with the intended application, and (v) interoperability and ability to use legacy software.

A possible security architecture that aims to provide the above mentioned properties is shown in Figure 2. Its realization deploys various technologies such as virtualization and Trusted Computing (TC). The Virtual Machine Monitor (VMM) consists of a security kernel that is located as a control instance between the hardware and the application layer. It implements elementary security properties like trusted channels and strong isolation between processes. Virtualization technology enables reutilization of legacy operating systems and existing applications whereas TC technology serves as the root of trust.

Hardware Layer. The hardware layer consists of commercial off-the-shelf PC hardware enhanced with trusted computing technology as defined by the Trusted Computing Group (TCG) [51] (see also Section 2).

Virtualization Layer. The main task of the virtualization layer, also called *hypervisor*, is to provide an abstraction of the underlying hardware, e.g., CPU, interrupts and devices, and to offer an appropriate management interface as well as inter-process communication (IPC). Device drivers and other essential operating system services, such as process and memory management, run in isolated user-mode processes. Moreover, this layer enforces an access control policy based on these resources. Providing a virtualized environment is one approach

to secure computing systems that process potentially malicious code. This technique is widely used for providing V-Servers, i.e., servers that feature several virtual machines. While users have full control over the virtual environment, they should not be able to cause damage outside that environment. Although virtualization offers abstraction from physical hardware and some control over process interaction, there still are problems to be solved. For example, in the x86 architecture, direct memory access (DMA) devices can access arbitrary physical memory locations. However, new hardware generations (see Section 9) aim to address these problems and could eventually lead to secure isolation among virtual machines. Virtualization technology can be leveraged for building an appropriate environment for a variety of applications, especially because several works, such as [43,44], have already begun to consider architectures that feature policy enforcement in the virtualization framework.[7] Possible implementations can be based on microkernels [29] or Xen hypervisor [6].

Trusted Software Layer. The trusted software layer provides various security related services and uses the functionality offered by the virtualization layer to provide security functionalities on a more abstract level. A possible design of these services is illustrated in Figure 2, which is being currently implemented within different projects.[8] The main services and their main tasks are as follows: A *Compartment Manager* for managing compartments, i.e., logically isolated components, e.g., virtual machines (VMs); a *Storage Manager* for compartments' persistent storage providing integrity, confidentiality, authenticity and freshness; a *Trust Manager* for providing trusted computing services, e.g., attestation, sealing, trusted channels to other (remote) compartments; a *User Manager* for managing user accounts and credentials; and a *Secure UI* for establishing a trusted path between the user and compartments, e.g., to assure to which application a user is communicating and to prevent keyloggers and spyware attacks.

Application Layer. On top of the security kernel, several instances of legacy operating systems (e.g., Linux) as well as security-critical applications (e.g., grid jobs, online banking, hard disk encryption, e-voting, VPN, DRM, etc.) can be executed in strongly isolated compartments. The proposed architecture offers migration of existing legacy operating systems. The legacy operating system provides all operating system services that are not security-critical and offers users a common environment and a large set of existing applications. If a mandatory security policy requires isolation between applications of the legacy OS, they can be executed by parallel instances of the legacy operating system.

Verifiable Initialization. For verifiable bootstrapping of the TCB, the CRTM measures the Master Boot Record (MBR) before passing control to it. A secure

[7] sHype is a security extension for Xen developed by IBM. sHype provides a MAC-based security architecture by adding static type enforcement (static coloring) on communication channels and allows to enforce a *Chinese Wall* restriction on concurrently running domains.

[8] EMSCB (`www.emscb.org`) and OpenTC (`www.opentc.net`)

chain of measurements is then established: before a program code is executed, it is measured by a previously (measured and executed) component.[9] The measurement results are securely stored in PCRs of the TPM. All further compartments, applications, and legacy OS instances are then subsequently loaded, measured, and executed by the Compartment Manager.

Examples for a security architecture as above is PERSEUS [37]. Other related work include Microsoft's *Next-Generation Secure Computing Base* (NGSCB) and *Terra* [20]. NGSCB was originally planed as a security architecture based on the TCG specifications [35]. However, the status of this project is currently unclear.[10] Terra is a trusted virtual machine monitor (VMM) that partitions a hardware platform into multiple, strictly isolated virtual machines (VM).

11 Applications

Trusted Computing framework and the related security architectures (see, e.g., Section 10) enable policy enforcement, and in particular multilateral security, which is crucial for applications with sophisticated security needs such as commercial grid, e-government, e-health, online banking to name some. In the following we consider some of the recent work on application of TC functionalities.

In [32,30] the authors propose to utilize TC technology for grid applications, with [48] more closely examining which scenarios require TC techniques. The common protection mechanisms in grid usually do not concern themselves with protecting the grid user (the person or entity wishing to utilize resources). The user is forced to trust the provider, often without the possibility to verify whether that trust is justified. However, in most of the current literature on grid security the user is usually not regarded as trustworthy. This trust asymmetry could potentially lead to a situation in which the grid provider causes large damage to the user with little risk of detection or penalty.

Trusted monotonic counters enable different types of distributed services that are vulnerable to replay attacks, including offline payment, e-wallets, virtual trusted storage, and management of stateful licenses in various digital rights management (DRM) applications. Stateful licenses are required in scenarios for trading and using digital goods where the enforcement of the underlying policies requires to securely maintain the state information about the past usage or environmental factors. In [46] the authors propose a trusted monotonic counter solely based on multiplexing the TPM counters without requiring a trusted operating system whereas [5] proposes a security architecture for the secure management (e.g., usage, and transfer) of stateful licenses and content on a virtualized open platform. The proposed architecture makes use of the monotonic counters of the

[9] For this purpose, one can use a modified GRUB bootloader c(`www.prosec.rub.de/tgrub.html`).

[10] Microsoft has often changed the name of its security platforms: Palladium (see, e.g., [47]), NGSCB, Longhorn and recently Windows Vista which uses TPM v1.2 for hard disk encryption (system partition) to protect data against theft.

TPM combined with a software security service to provide trusted storage with freshness.

In [19], the authors propose the design and implementation of a security architecture that aims at preventing both classical and malware phishing attacks. The security architecture deploys ideas of compartmentalization for isolating applications of different trust level, and a trusted wallet for storing credentials and authenticating sensitive services. Other work on preventing phishing attacks based on TC can be found in [1].

In [10] the authors describe a secure network virtualization framework for establishing *Trusted Virtual Domains* (TVDs) and its implementation based on Xen hypervisor [6]. The proposed framework aims at connecting groups of related virtual machines running on separate physical machines as if there were on their own separate network while enforcing the domain's security policy (in their case isolation, confidentiality, and information flow control). Trusted computing functionalities can assure that core components for establishing TVDs are trustworthy, i.e., conform to a the well-defined security policy.

Other applications include deployment of TC for biometric systems security, video broadcasting, medical privacy, Web security, protection of massively multiplayer online games (MMOGs), RFID applications with privacy need, single sign-on, secure information sharing, protecting digital signatures, TC-based security architecture for mobile platforms, VPN and hard disk encryption applications.[11]

12 Conclusion and Future Work

Trusted Computing (TC) is an emerging technology that can enhance the security of computing platforms in many ways. Nevertheless, TC technology is not a universal solution to all of the IT security problems, but rather an additional tool to pave the way towards trusted infrastructures. The technological realization of TC brings new technical but also legal and economical challenges. It is difficult to predict whether or not the security benefits of TC outweighs the efforts needed to face these challenges. Though we strongly believe that research on this technology results in a deeper understanding of complex IT systems and how to maintain their integrity, since trusted computing concerns security aspects at many system abstraction layers (hardware, operating systems and application software). Many challenges still remain as we discussed previously: designing effective and efficient remote proof of trustworthiness of codes, or of a platform (e.g., property-based attestation), incorporating the underlying hardware in the chain of trust, designing a minimal and less complex but effective TPM, simplifying complex cryptographic protocols by using TC functionalities, developing secure and efficient VMMs, establishing appropriate formal security models and analysis for the corresponding mechanisms and architectures.

[11] See `www.isg.rhul.ac.uk/~pnai178/tcgresources.htm`, and `www.emscb.org` for a collection of papers on these topics.

In particular, trusted mobile platforms seem to be an important future application and market segment since embedded devices are increasingly used for security critical applications and have challenging requirements due to their constrains but also the diversity of the parties involved as well as their requirements and interests.

Even though the security level strongly depends on the details of design and implementation, Trusted Computing as an abstract functionality is inevitable for realizing applications and IT infrastructures with sophisticated security requirements.

Acknowledgment. We are very grateful to Frederik Armknecht, Christian Stüble and in particular to Christian Wachsmann for the fruitful discussions and their valuable comments on the early draft of this survey.

References

1. Alsaid, A., Mitchell, C.J.: Preventing Phishing Attacks using Trusted Computing Technology. In: INC (July 2006)
2. Anderson, R.J.: Security Engineering: A Guide to Building Dependable Distributed Systems, 1st edn. John Wiley & Sons, New York, USA (2001)
3. Anderson, R.J.: The TCPA/Palladium FAQ (2002), `http://www.cl.cam.ac.uk/~rja14/tcpa-faq.html`
4. Arbaugh, W.A., Farber, D.J., Smith, J.M.: A secure and reliable bootstrap architecture. In: Proceedings of the IEEE Symposium on Research in Security and Privacy, IEEE Computer Society, Technical Committee on Security and Privacy, pp. 65–71. IEEE Computer Society Press, Los Alamitos (1997)
5. Asokan, N., Ekberg, J.-E., Sadeghi, A.-R., Stüble, C., Wolf, M.: Enabling fairer digital rights management with trusted computing. In: ISC (2007)
6. Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., Warfield, A.: Xen and the art of virtualization. In: SOSP (2007)
7. Benzel, T.V., Irvine, C.E., Levin, T.E., Bhaskara, G., Nguyen, T.D., Clark, P.C.: Design principles for security. Technical Report NPS-CS-05-010, Naval Postgraduate School (September 2005)
8. Berger, S., Caceres, R., Goldman, K.A., Perez, R., Sailer, R., van Doorn, L.: vTPM: Virtualizing the Trusted Platform Module. In: Proceedings of the 15th USENIX Security Symposium, pp. 305–320. USENIX (August 2006)
9. Brickell, E., Camenisch, J., Chen, L.: Direct anonymous attestation. In: ACM-CCS (October 2004)
10. Cabuk, S., Chris, H.R., Dalton, I., Schunter, M.: Towards automated provisioning of secure virtualized networks. In: ACM-CCS (2007)
11. Carroll, A., Juarez, M., Polk, J., Leininger, T.: Microsoft "Palladium": A business overview. Technical report, Microsoft Content Security Business Unit (August 2002)
12. Chen, L., Landfermann, R., Löhr, H., Rohe, M., Sadeghi, A.-R., Stüble, C.: A protocol for property-based attestation. In: ACM-STC, ACM Press, New York (2006)

13. Common Criteria Project Sponsoring Organisations. Common criteria for information technology security evaluation. Norm Version 2.1, CCIMB-99-031 – 33, Common Criteria Project Sponsoring Organisations (August 1999), http://csrc.nist.gov/cc/CC-v2.1.html
14. Dierks, T., Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.1, RFC4346 (April 2006), http://www.ietf.org/rfc/rfc4346.txt
15. Dyer, J., Lindemann, M., Perez, R., Sailer, R., van Doorn, L., Smith, S.W., Weingart, S.: Building the IBM 4758 Secure Coprocessor. IEEEC 34(10), 57–66 (2001)
16. Felten, E.W.: Understanding Trusted Computing — Will Its Benefits Outweigh Its Drawbacks? IEEE Security and Privacy, 60–62 (May/June 2003)
17. Figueiredo, R., Dinda, P.A., Fortes, J.: Resource virtualization renaissance. IEEE Computer 38, 28–31 (2005)
18. Foundation, F.S.: GNU General Public License, Version 3, http://gplv3.fsf.org/
19. Gajek, S., Sadeghi, A.-R., Stüble, C., Winandy, M.: Compartmented security for browsers — or how to thwart a phisher with trusted computing. In: ARES (2007)
20. Garfinkel, T., Pfaff, B., Chow, J., Rosenblum, M., Boneh, D.: Terra: a virtual machine-based platform for trusted computing. In: SOSP, pp. 193–206. ACM, New York (2003)
21. Gasmi, Y., Sadeghi, A.-R., Stewin, P., Unger, M., Asokan, N.: Beyond secure channels. In: ACM-STC (2007)
22. Goldberg, R.P.: Architectural Principles for Virtual Computer Systems. PhD thesis, Harvard University (1972)
23. Gürgens, S., Rudolph, C., Scheuermann, D., Atts, M., Plaga, R.: Security evaluation of scenarios based on the TCG TPM specification. In: Biskup, J., López, J. (eds.) ESORICS 2007. LNCS, vol. 4734, pp. 438–453. Springer, Heidelberg (2007)
24. Haldar, V., Chandra, D., Franz, M.: Semantic remote attestation: A virtual machine directed approach to trusted computing. In: USENIX Virtual Machine Research and Technology Symposium May 2004, Also Technical Report No. 03-20, School of Information and Computer Science, University of California, Irvine (October 2003)
25. Kent, S., Atkinson, R.: Security Architecture for the Internet Protocol, RFC2401 (November 1998), www.ietf.org/rfc/rfc2401.txt
26. Kühn, U., Kursawe, K., Lucks, S., Sadeghi, A.-R., Stüble, C.: Secure data management in trusted computing. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, Springer, Heidelberg (2005)
27. Kühn, U., Selhorst, M., Stüble, C.: Property-Based Attestation and Sealing with Commonly Available Hard- and Software. In: ACM-STC (2007)
28. Kursawe, K., Schellekens, D., Preneel, B.: Analyzing Trusted Platform Communication. In: ECRYPT-CRASH (2005)
29. Liedtke, J.: Towards real micro-kernels. Commun. ACM 39(9) (1996)
30. Löhr, H., Ramasamy, H.G.V., Schulz, S., Schunter, M., Stüble, C.: Enhancing Grid Security Using Trusted Virtualization. In: ATC (2007)
31. MacDonald, R., Smith, S., Marchesini, J., Wild, O.: Bear: An open-source virtual secure coprocessor based on TCPA. Technical Report TR2003-471, Department of Computer Science, Dartmouth College (2003)
32. Mao, W., Jin, H., Martin, A.: Innovations for Grid Security from Trusted Computing, http://www.hpl.hp.com/personal/Wenbo_Mao/research/tcgridsec.pdf

33. Marchesini, J., Smith, S., Wild, O., Barsamian, A., Stabiner, J.: Open-source applications of TCPA hardware. In: ACSAC, ACM, New York (2004)
34. Marchesini, J., Smith, S.W., Wild, O., MacDonald, R.: Experimenting with TCPA/TCG hardware, or: How I learned to stop worrying and love the bear. Technical Report TR2003-476, Department of Computer Science, Dartmouth College (2003)
35. Mundie, C., de Vries, P., Haynes, P., Corwine, M.: Microsoft whitepaper on trustworthy computing. Technical report, Microsoft Corporation (October 2002)
36. Oppliger, R., Rytz, R.: Does trusted computing remedy computer security problems? IEEE Security & Privacy 3(2), 16–19 (2005)
37. Pfitzmann, B., Riordan, J., Stüble, C., Waidner, M., Weber, A.: The PERSEUS system architecture. Technical Report RZ 3335 (#93381), IBM Research Division, Zurich Laboratory (April 2001)
38. Poritz, J., Schunter, M., Van Herreweghen, E., Waidner, M.: Property attestation—scalable and privacy-friendly security assessment of peer computers. Technical Report RZ 3548, IBM Research (May 2004)
39. Rozas, C.: Intel's Security Vision for Xen (April 2005), `http://www.xensource.com/files/XenSecurity_Intel_CRozas.pdf`
40. Sadeghi, A.-R., Selhorst, M., Christian Stüble, C., Wachsmann, Winandy, M.: TCG Inside? — A Note on TPM Specification Compliance. In: ACM-STC (2006)
41. Sadeghi, A.-R., Stüble, C.: Property-based attestation for computing platforms: Caring about properties, not mechanisms. In: ACM SIGSAC, The 2004 New Security Paradigms Workshop, ACM Press, New York (2004)
42. Safford, D.: The need for TCPA. IBM Research (October 2002)
43. Sailer, R., Jaeger, T., Valdez, E., Caceres, R., Perez, R., Berger, S., Griffin, J.L., van Doorn, L.: Building a MAC-Based Security Architecture for the Xen Open-Source Hypervisor (2005)
44. Sailer, R., Valdez, E., Jaeger, T., Perez, R., van Doorn, L., Griffin, J.L., Berger, S.: sHype: Secure hypervisor approach to trusted virtualized systems. Technical Report RC23511, IBM Research Division (February 2005)
45. Sailer, R., Zhang, X., Jaeger, T., van Doorn, L.: Design and implementation of a TCG-based integrity measurement architecture. Research Report RC23064, IBM Research (January 2004)
46. Sarmenta, L.F.G., van Dijk, M., O'Donnell, C.W., Rhodes, J., Devadas, S.: Virtual monotonic counters and count-limited objects using a tpm without a trusted os. In: ACM-STC, pp. 27–42 (2006)
47. Schoen, S.: Palladium details (2002), `http://www.activewin.com/articles/2002/pd.shtml`
48. Smith, M., Friese, T., Engel, M., Freisleben, B.: Countering Security Threats in Service-Oriented On-Demand Grid Computing Using Sandboxing and Trusted Computing Techniques. Journal of Parallel and Distributed Computing (2006)
49. Smith, S.W.: Outbound authentication for programmable secure coprocessors. In: Gollmann, D., Karjoth, G., Waidner, M. (eds.) ESORICS 2002. LNCS, vol. 2502, pp. 72–89. Springer, Heidelberg (2002)
50. Spafford, G.: Risks Digest 19.37 (September 1997), `http://catless.ncl.ac.uk/Risks/19.37.html`
51. Trusted Computing Group, `www.trustedcomputinggroup.org`

52. Trusted Computing Group. TCG Architecture Overview (April 2004)
53. Trusted Computing Group. TPM main specification. Main Specification Version 1.2 rev. 85, Trusted Computing Group (February 2005)
54. Trusted Computing Group (TCG). About the TCG, http://www.trustedcomputinggroup.org/about/
55. Trusted Computing Group (TCG). TCG Mobile Reference Architecture, Specification version 1.0, Revision 1 (June 12, 2007)
56. Trusted Computing Group (TCG). TCG Mobile Trusted Module Specification, version 1.0, Revision 1 (June 12, 2007)
57. Trusted Computing Platform Alliance (TCPA). Main specification, Version 1.1b (February 2002)
58. Yee, B.S.: Using Secure Coprocessors. PhD thesis, School of Computer Science, Carnegie Mellon University, CMU-CS-94-149 (May 1994)