Sixth Framework Programme
Information Society Technology

ReTrust

RE-TRUST

Remote EnTrusting by RU-n-time Software authentication

Project Number: FP6 - 021186

Deliverable: D4.5

Comparative analysis of RE-TRUST with TC
### Summary

<table>
<thead>
<tr>
<th>Project Number:</th>
<th>FP6-021186</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Title:</td>
<td>RE-TRUST: Remote En Trusting by RUN-time Software auThentication</td>
</tr>
<tr>
<td>Deliverable Type:</td>
<td>RP</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Deliverable Number:</th>
<th>D4.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contractual Date of Delivery:</td>
<td>September 2009</td>
</tr>
<tr>
<td>Actual Date of Delivery:</td>
<td>September 2009</td>
</tr>
<tr>
<td>Title of Deliverable:</td>
<td>Comparative Analysis of RE-TRUST with Trusted Computing</td>
</tr>
<tr>
<td>Workpackage Contributing to the Deliverable:</td>
<td>WP4</td>
</tr>
<tr>
<td>Nature of the Deliverable:</td>
<td>Report</td>
</tr>
<tr>
<td>Author(s):</td>
<td>Dries Schellekens and Brecht Wyseur (K.U.Leuven)</td>
</tr>
<tr>
<td>Reviewer(s):</td>
<td></td>
</tr>
</tbody>
</table>

**Abstract:**

This deliverable comprises of a comparative analysis between RE-TRUST and Trusted Computing. We introduce the Trusted Computing approaches, and how they enable load-time attestation of applications, followed by a discussion on the comparison and synergies with the RE-TRUST approaches.

**Keywords:** Trusted computing, trusted platform module, application attestation

**Classification:**

**Name of Client:** European Commission

**Distribution List:** European Commission Project Partners

**Authorised by:** 1.0

**Issue:** RE-TRUST Deliverable

**Reference:** --

**Total Number of Pages:** Prof. Yoram OFEK

**Contact Details:** University of Trento, Dept. of Information and Communication Technology (DIT), Trento, Italy

Phone: +39 0461 88 3954 (office)

Fax +39 0461 88 2093 (DIT)

E-mail: ofek -at- dit.unitn.it
## Contents

1 Introduction  

2 Background  
   2.1 Closed Platforms  
   2.2 Open Platforms  
   2.3 Secure Coprocessor  
   2.4 Trusted Computing Platforms  
   2.5 Compatibility with Legacy Operating System  

3 Trusted Computing  
   3.1 TCG Overview  
      3.1.1 Trusted Platform Module  
      3.1.2 Integrity measurement  
      3.1.3 Integrity reporting  
   3.2 Application Level Attestation  
      3.2.1 Trustworthiness of Operating System  
      3.2.2 Load-time Binary Attestation  
      3.2.3 Hybrid Attestation Schemes  

4 Comparison and possible synergies  

Bibliography
1 Introduction

This deliverable describes the work performed within the context of Subtask T4.4.1 of the RE-TRUST project. The objective of this task is to provide a comparative analysis of RE-TRUST solutions and alternative solutions using trusted hardware.

We specifically focus on the Trusted Computing approach. This refers to the architecture that was introduced by the Trusted Computing Platform Alliance (TCPA) and its successor the Trusted Computing Group (TCG) and that was originally designed for the open general purpose Personal Computer (PC) platform. A TCG enabled PC contains an additional hardware component, the so-called Trusted Platform Module (TPM), and a supporting software stack that reliably measures and records software components that get loaded during startup. The basic idea is that this enables a verifier to get a guarantee of the load-time integrity of a platform.

The RE-TRUST approach on the other hand intends to provide a solution for run-time integrity (as opposed to load-time integrity), and relies on the continuous availability of a network connection to a trusted entity instead of the availability of a trusted hardware core (the TPM). This deliverable intends to compare these two approaches.

The deliverable is organized as follows: first we present the background of the Trusted Computing initiatives and the state of the art in software security with respect to the architecture that we discuss. In Sect. 3 we present the technical details of the TCG approach, where Sect. 3.2 discusses how TC enables the verification of the integrity of applications. Sect. 4 then presents the comparison between the TC approach and the RE-TRUST approaches.

2 Background

As today’s software is becoming more and more mobile and inherently networked, and their tasks get increasingly critical, mechanisms should be in place to establish trust relationships between computing platforms. For instance, in online banking the bank wants be assured that a financial transaction is generated by a legitimate client of the bank and not by malware that has infected the client’s computer. Similarly, providers of digital content like music, movies and e-books want to check whether a so called Digital Rights Management (DRM) system is properly installed on the consumer’s platform. The DRM software typically restricts the usage of the digital content; e.g., the content can only be played on a certain amount of computers or media players, for a limited number of times or during a specific time period. In online games “misbehaving” users must be identified. The usage of bots that automate certain actions in the game, or the installation of cheat software that give the user advantages over the other players (e.g., viewing through walls) must be detected. As a final example, it would be desirable in VPN solutions to grant remote access to a corporate network for the public Internet not only based on user credentials (e.g., password, biometrics), but also on the verification
of the platform’s integrity.

For all these applications, it is clear that only legitimate, untampered client applications should be granted access to a service. Hence, an authorized entity wants to be able to both identify a remote platform and verify if its software is running untampered. The Trusted Computing Group (TCG) defines this process as *remote attestation*. If tampering is detected, this verifier will want to disconnect the client from the network, stop the service to this particular client, or even force that client application to halt its execution.

### 2.1 Closed Platforms

In closed systems, communicating platforms have an a priori trust relationship. The client platform is assumed to only run the legitimate software of the service provider and cryptographic keys to access the service can be embedded inside the device. Typical examples are Consumer Electronics devices like DVD players and recorders, portable media players, satellite TV receivers, digital TV set-top boxes, and game consoles. Often the user of such device has an incentive to modify the original software or extract the embedded keys; for instance to play a DVD with a foreign region code, to watch pay TV for free, or to play a pirate copy of a computer game.

Often the integrity of code executing on a closed platform does not have to be verified remotely as no software interface is provided to install malicious modified code. The fact that a device has access to the certain cryptographic keys is sufficient evidence that the service provider is communicating with an authentic platform. Therefore there is an implicit trust relationship. The closeness of the platform’s software forces attackers to resort to hardware attacks on the platform. Consequently numerous security mechanisms are commonly implemented in hardware: e.g., the initial boot loader of the platform is stored in Read-only Memory (ROM) and only starts authorized code (i.e., signed by the device manufacturer), cryptographic keys are stored in a tamper resistant module like a smart card, and the communication and memory buses of the platform are physically and/or cryptographically protected against eavesdropping and tampering.

### 2.2 Open Platforms

On open platforms such as the Personal Computer (PC) an adversary potentially has total control over all the software on the platform including the Operating System (OS). The adversary can remotely compromise the platform through a security vulnerability, but he can also have local control of the platform if he is trying to attack an application on his own machine. The latter is for instance the case when someone attempts to circumvent a DRM system. Moreover, adversaries with local access can perform hardware attacks, such as using Direct Memory Access (DMA) to read and/or alter the main computer memory.
Establishing a secure execution environment in such conditions is a big challenge. A lot enabling technologies has been researched in this area. Tamper resistant software\cite{[4, 12, 27]} has built-in integrity checks to detect tampering of its code. Typically this approach is complemented with obfuscation techniques \cite{[15, 16, 33, 59]} that harden the reverse engineering of the binary executable and as such make it more difficult to understand how to circumvent a tamper detection mechanism. Finally white-box cryptography\cite{[61]} aims to hide cryptographic keys into applications, either in large collection of lookup tables \cite{[13, 14]} or in executable code \cite{[39]}. The latter case is a form of tamper resistant software, as code modifications will alter the key and consequently cripple the functionality of the application.

However, when these software techniques are used to protect standalone, non-networked applications, their security is limited. Self-checking software can be attacked with hardware support \cite{[58, 60]} and the tamper response mechanism is often a weak point \cite{[57]}, obfuscation makes the reverse engineering process more time consuming but not impossible, and most proposals for a white-box block cipher have been broken \cite{[28, 6, 22, 62]}. Tamper resistant software typically calculates a checksum on its code and checks whether the checksum corresponds with an expected value. In offline applications this expected value has to be stored inside the software and the decision whether tampering has occurred, has to be taken locally by the client software itself.

Networked applications suffer less from these issues. The integrity checksums do not have to be present in the client software and the comparison between the run-time and the pre-computed checksum is performed remotely by the service provider, which is not under control of an attacker. Additionally the service provider can periodically replace the client application with a new version, containing a different cryptographic key and/or obfuscated in another way \cite{[11]}. This code replacement can be used to limit the time an adversary has to reverse engineer a version of the application.

An adversary having complete control over an untrusted platform also has control over its input and output network traffic. This makes it difficult for a remote verifier to be assured of communicating with a particular environment on a given platform. The attacker can forward the remote attestation protocol from a tampered platform to an honest platform. Similarly, he can compromise the platform immediately after the attestation protocol has verified the integrity of the platform.

Even more, the verifier has to determine whether the software is running directly on the OS of the platform or in a simulator, emulator or virtual machine. So called genuinity tests \cite{[30]} have been developed to verify if software is running on certain hardware. These tests leverage detailed knowledge about the processor of the untrusted platform and are slow to execute on other processors or to simulate. In practice however, the proposed solution turns out to be flawed \cite{[51]}. The Pioneer system proposed in \cite{[48, 49]} establishes whether software on an untrusted host is untampered by calculating a checksum over its run-time memory image. If the resulting checksum is not reported within a defined time frame, the verifier assumes that the checksum function itself has been altered; the timing information helps to detect
the overhead caused by modifications to the checksum functionality and redirection of
the network flow. The proposed solution was first proposed for embedded systems with
a low end microcontroller [50] and later for legacy PC systems, but it relies on some
strong assumptions.

Other techniques try to verify computations performed on the untrusted host, e.g.,
by embedding trace gathering code in the original program and locally cross checking the
trace [40] or by verifying certain assertions [9]. Alternatively, one can limit the impact
of tampering by moving critical code away from untrusted platforms or by computing
on encrypted data and/or with encrypted functions [45] (see Deliverable D3.3 for more
details). Techniques such as program slicing split software into non-critical and critical
code slices. Only the non-critical code is run on the untrusted platform, guaranteeing
that the critical slices can not be tampered [66, 17, 9, 10]. This is a form of server side
execution. The bulk of the work performed in Work Package 2 focussed on these novel
software based approaches for remote attestation.

2.3 Secure Coprocessor

The software based attestation schemes proposed for open platform will never give the
same confidence level as the hardware mechanisms of a closed platform. Therefore, in
the nighties the idea arose to add a secure coprocessor to the open PC platform [63,
64, 55, 56]. Basically this coprocessor offers a closed execution environment next to the
untrustworthy legacy OS. The security mechanisms of closed platforms are applied: the
coprocessor only executes authenticated code and physical shielding provides hardware
tamper resistance.

Although a lot of research has been done on secure coprocessors, its commercial
success is limited. The IBM 4758 [18] is the only available off-the-shelf product. It is
a PCI card with a 486 processor, a cryptographic engine, and battery backed RAM for
non-volatile storage and it runs a proprietary OS called CP/Q++ that supports custom
applications. Its successor, the IBM 4764, uses a PPC 405 processor and embedded
Linux as OS. The IBM secure coprocessor family support outbound authentication [53,
54]: the ability of coprocessor applications to authenticate themselves to remote parties.

In some sense the latest generation of smart card meets the definition of secure
coprocessor. Traditionally smart cards have been constrained in processing power and
storage capacity and a dedicated smart card reader was needed. Hence the application
of classical cards has been limited to some specific tasks, like authentication (e.g., SIM
card), identification (e.g., eID card) and financial payments (e.g., EMV credit card).
However, the latest generation of smart cards is getting a high speed interface (i.e.,
USB), a high density non-volatile memory (i.e., Flash memory) and a more powerful
microprocessor (e.g., 32 bit ARM).

With the appropriate cryptographic techniques the coprocessor can establish a se-
cure communication channel to a remote entity through the network connection of the
untrustworthy host computer. This channel can be used to report the identity and
integrity of code executing in the secure environment, and to update and configure its software components. However, it is less straightforward to establish a secure path to a human operator of the open platform: the input of the user (e.g., key strokes entered on the keyboard or mouse movement) and the output to the user (e.g., information displayed on the computer screen) can be intercepted and manipulated by malicious software on the PC.

The secure coprocessor approach differs from the techniques investigated in Work Package 3. A secure coprocessor runs the entire application that needs to be protected in an execution environment isolated from the rest of the untrusted platform. However, the hardware assisted protection techniques of the RE-TRUST project use the secure hardware to strengthen the software based remote attestation and mainly run a protection mechanism in the secure coprocessor.

2.4 Trusted Computing Platforms

In the nighties academic researchers proposed architectures to improve the trustworthiness of the PC bootstrap process. All assume the BIOS, which acts as initial boot loader of the PC platform, to be immutable and use it as trust anchor for a secure bootstrap. Arbaugh introduced the concept of chaining layered integrity checks [2, 1]. Each software component loaded during the boot process (starting from the BIOS) checks the integrity of next component (by verifying a digital signature) before passing control to it. He also defined a mechanism for automatic recovery of corrupt or invalid bootstrap components [3]. This proposal effectively turn the PC into a closed platform as it restricts the software that can be booted.

Groß [24, 26] defined a secure bootstrap architecture that supports remote attestation. The platform contains a unique asymmetric key pair signed by the hardware manufacturer and the operating system is signed by the OS producer. During startup the integrity of the OS is checked by verifying the digital signature, and the platform signs the identity of the OS with its private key yielding a boot certificate. During operation the integrity of the platform can be remotely verified with a cryptographic challenge-response protocol that transfers the boot and hardware certificate.

The Trusted Computing Platform Alliance (TCPA) and its successor the Trusted Computing Group (TCG) opted for a different approach that respects the openness of the PC platform. A TCG enabled platform reliably measures the software components that get loaded during startup by calculating their cryptographic hash and records these measurements in a hardware security module, the Trusted Platform Module (TPM). This approach does not impose restrictions on the OS that the platform can boot, as the TPM merely operates as a logging device that does not actively intervene in the bootstrap process. Later the booted configuration can be reported to a remote entity with an attestation protocol and used to bind secret keys to the platform. The former enables service providers to restrict access to a network service based on the measured platform configuration and identity.
The remote attestation provided by TCG platforms has a number of issues which limit practical deployment. Firstly, in its original form the TCG attestation process posed some privacy concerns, which are partially addressed by the Direct Anonymous Attestation (DAA) protocol \[7, 8\] of the latest TPM specification. Secondly, binary measurement of the platform configuration has scalability issues because managing the multitude of possible configurations can be troublesome, and allows for discrimination of certain configurations. Lastly, attestation of individual applications \[44\] necessitates a secure operating system. Property-based attestation \[42, 43\] and the latest virtualization technology promise to overcome some of these shortcomings.

### 2.5 Compatibility with Legacy Operating System

Pure software approaches for remote attestation, relying on timed execution of a checksum function, impose a number of limitations. It is impossible to uniquely identify the platform, creating an opportunity for proxy attacks. To determine the expected execution time of the checksum computation, detailed knowledge about the processor of the untrusted platform is needed. The adversary will be tempted to replace the processor with a faster one such that the extra computing cycles can be used to tamper with the checksum function. The expected execution time can be unreliable because the verifier has to make a worst case assumption on the network latency, which can be rather unpredictable on the Internet. As identified in Deliverable D4.6 other techniques developed in the RE-TRUST project can often be circumvented by environmental attacks which alter the underlying operating system and/or the platform’s hardware.

On the other hand, a lot of computers equipped with TPMs are sold today, but their functionality is hardly used. The main reason for this is the lack of the required software support. If legacy operating systems such as Windows and Linux are used on a TCG platform, the chain of trust can be easily subverted; e.g., by loading a malicious device driver or by exploiting a kernel level security vulnerability. A solution that is often proposed to increase the trustworthiness of the PC platform while maintaining backward compatibility, is the usage of a virtual machine monitor or hypervisor \[20, 21, 31, 34, 65\]. In this way a security critical application can run in a dedicated compartment isolated from the virtual machine hosting the legacy OS. The integrity of the application compartment and the hypervisor can be verified with a remote attestation protocol. Trusted virtualization layers have been researched and developed, for instance in Microsoft’s Next-Generation Secure Computing Base (NGSCB) project \[19, 41\] and the European OpenTC project \[31\], but are not yet commercially available.

Given the shortcomings of software based attestation schemes and the lacking software support for TCG platforms we proposed a hardware assisted software solution in \[46, 47\]. In particular we improved the Pioneer scheme by using the time stamping functionality provided by the TPM. Our solution only relies on a secure bootloader, instead of a secure operating system or a trusted virtualization layer.

Another approach is taken in the work of McCune et al. \[35, 36, 37, 38\]. They
propose to use the late launch capability offered by AMD’s Secure Virtual Machine (SVM) extensions and Intel’s Trusted Execution Technology (TXT) \cite{23} in order to create a strongly isolated execution environment that can be remote verified. The Trusted Computing Base (TCB) for this proposal is very small and hence the resulting solution potentially provide a strong level of assurance. This scheme has strong hardware requirements, i.e., the latest x86 processor and a TPM.

3 Trusted Computing

Trusted computing initiatives propose to solve some of today’s security problems of the underlying computing platforms through hardware and software changes. The two main initiatives for a new generation of computing platforms are the Trusted Computing Group (TCG) \cite{5}, a consortium of most major IT companies, and Microsoft’s Next-Generation Secure Computing Base \cite{19, 41}. We will solely focus on TCG technology, as these specifications are public and TCG enabled computers are commercially available.

3.1 TCG Overview

The TCG sees itself as a standard body only. Neither does it provide any infrastructure to fully utilize the technology, nor does it perform certification of any kind. The TCG specifications define three components that form a Trusted Platform\textsuperscript{1}.

The core is called the Trusted Platform Module (TPM) which usually is implemented by a smartcard-like chip bound to the platform.

The second component is called Core Root of Trust for Measurement (CRTM), and is the first code the TCG compliant platform executes when it is booted. In a PC, the CRTM is the first part of the BIOS, which can not be flashed or otherwise be modified.

To compensate for the lack of functionality in the TPM, the TCG specifies a TCG Software Stack (TSS), which facilitates some of the complex, but non-critical functionality and provides standard interfaces for high level applications.

3.1.1 Trusted Platform Module

The TPM is the main component of a TCG platform and offers a physical true random number generator, cryptographic functions (i.e., SHA-1, HMAC, RSA encryption/decryption, signatures and key generation), and tamper resistant non-volatile memory (mainly used for persistent key storage). Remark that no symmetric encryption algorithm is provided.

At an abstract level TPM can be described with the tuple \((EK, SRK, T)\).

\textsuperscript{1}All TCG specifications mentioned in this deliverable are available on https://www.trustedcomputinggroup.org.
The Endorsement Key EK uniquely identifies each TPM. This key is generated when the TPM is produced and comes with a number of certificates issued by the chip manufacturer, the platform manufacturer, and potentially a third party that verified the production process\(^2\).

The Storage Root Key SRK is uniquely created inside the TPM, when ownership over the TPM is taken. Other keys maintained by the TPM are encrypted under this SRK (or by any other key that is already maintained by the TPM, creating a key tree) and stored outside of the TPM, for example on hard disk. This allows the TPM to manage a virtually unlimited number of keys. The SRK can be revoked destroying all keys maintained by the TPM (e.g., by clearing ownership) and obviously the TPM can not prevent the operating system from deleting keys stored externally.

\(\mathcal{T}\) represents the TPM state embodying other security critical data protected by the TPM. This includes the owner’s authorization data (i.e., password) and the value of the monotonic counters.

The TPM offers a set of Platform Configuration Registers (PCR) that are used to store measurements (i.e., hash values) about the platform configuration. The content of these registers can only be modified using the extending operation\(^3\): \(\text{PCR}_{\text{new}} \leftarrow \mathcal{H}(\text{PCR}_{\text{old}} || M)\) with \(\text{PCR}_{\text{old}}\) the previous register value, \(\text{PCR}_{\text{new}}\) the new value, \(M\) a new measurement and \(||\) denoting the concatenation of values.

### 3.1.2 Integrity measurement

The initial platform state is measured by computing cryptographic hashes of all software components loaded during the boot process. The task of the CRTM is to measure (i.e., compute a hash of) the code and parameters of the BIOS and extend the first PCR register with this measurement. Next, the BIOS will measure the binary image of the bootloader before transferring control to the bootloader, which in its turn measures the operating system\(^4\). In this way a *chain of trust* is established from the CRTM to the operating system and potentially even to individual applications.

### 3.1.3 Integrity reporting

The TCG *attestation* allows to report the current platform configuration \((\text{PCR}_0, \ldots, \text{PCR}_n)\) to a remote party. It is a challenge-response protocol, where the platform con-

\(^2\)In practice Infineon currently is the only manufacturer providing an endorsement certificate. This means that it is not straightforward to distinguish a genuine hardware TPM from a software clone.

\(^3\)The version 1.2 specification introduces a number of PCR can be reset by higher privileged (determined by locality) code [23].

\(^4\)TrustedGRUB (https://prosec.trust.rub.de/trusted_grub.html) is an example of an open source bootloader that is enhanced to measure the operating kernel system. The OSLO bootloader [29], on the other hand, uses the AMD SKINIT instruction to create a dynamic root of trust for measurement; this has the advantage that the – potentially untrusted – BIOS is not included in the chain of trust.
figuration and an anti-replay challenge provided by the remote party are digitally signed with an Attestation Identity Key (AIK). If needed, a Stored Measurement Log (SML), describing the measurements that lead to a particular PCR value, can be reported as well. The AIK are pseudonyms of the Endorsement Key which uniquely identifies a TPM. A trusted third party called Privacy CA is used to certify the AIKs. Version 1.2 of the TCG specification defines a cryptographic protocol called Direct Anonymous Attestation [7] to eliminate the need for a Privacy CA, as it can potentially link different AIKs of the same TPM.

TCG technology also has the concept of sealing, enabling certain data or keys to be cryptographically bound to a certain platform configuration. The TPM will only release this data if a given configuration is booted. This can be considered as an implicit form of attestation: the application that needs to be externally verified, can seal a secret in the TPM and consequently, if the application is able to unseal the secret, the platform is known to be in a given state.

3.2 Application Level Attestation

TCG attestation is designed to provide remote verification of the complete platform configuration, i.e., all software loaded since startup of the platform. Hence, in order to remotely verify the integrity of a program on a trusted computing platform, first the integrity of the boot sequence up to the operating system must be established and next the operating system should report the integrity of the application(s) that a remote verifier is interested in.

3.2.1 Trustworthiness of Operating System

Establishing this chain of trust to individual programs is not straightforward in practice, as it necessitates a trustworthy operating system. The operating system needs to measure the integrity of all privileged code it loads including third party device drivers, because these can be used to subvert the integrity of the kernel. Commonly loadable kernel modules are used to inject kernel backdoors. However, legacy operating system are monolithic, too big and too complex to provide a sufficiently small Trusted Computing Base [36] and hence they are prone to security vulnerabilities.

As legacy operating system can not guarantee a chain of trust beyond the bootloader, trusted computing initiatives typically opt for a microkernel or hypervisor in combination with hardware virtualization support to achieve both security and backward compatibility [20]. Security sensitive applications will execute in a separate compartment, strongly isolated from the virtual machine that hosts the legacy operating system. In this way, the (potentially) insecure OS is excluded from the chain of trust. Obviously in this solution the underlying virtual machine monitor needs to be trusted.
3.2.2 Load-time Binary Attestation

A first approach to attest individual program is to directly apply the TCG (i.e., load-time binary) attestation on all userland components [44]. On the creation of user level processes, the kernel measures the executable code loaded into the process (i.e., the original executable and shared libraries) and this code can subsequently measure security sensitive input its loads (e.g., arguments, configuration files, shell scripts). All these measurements are stored in some PCR register and the SML.

In its basic form TCG attestation has some shortcomings. First, a huge number of possible configurations exist, because every new version of a component will have a different binary and hence produces a different hash value.

Furthermore load-time attestation provides no run-time assurance because there can be a big time difference between integrity measurement (i.e., startup) and integrity reporting. The platform could be compromised since it has been booted.

There can also be privacy concerns as binary measurements reveal a lot about the software that is running on the platform. The detailed knowledge about the exact program binary that is executing, could be misused to discriminate specific software configurations (e.g., access to a service by competing software can be refused) or to detect whether a platform is running software that contains security vulnerabilities.

3.2.3 Hybrid Attestation Schemes

To overcoming some of the shortcomings of binary attestation, a number of more flexible attestation mechanisms have been proposed.

BIND [52] tries to provide fine grained attestation by not verifying the complete memory content of an application, but only the piece of the code that will be executed. On top of that it allows to include the data that the code produces in the attestation data. The solution requires the attestation service to run in a more privileged execution environment and the integrity of the service is measured using the TPM.

In [25] the concept of semantic remote attestation is proposed. This is also a hybrid attestation scheme, where a virtual machine is attested by the TPM and the trusted virtual machine will certify certain semantic properties of the running program.

Property based attestation [43] takes a similar approach where properties of the platform and/or applications are reported instead of hash values of the binary images. One practical proposal is to use delegation based property attestation: a certification agency certifies a mapping between properties and configurations and publishes these property certificates [32].

All these solutions require the attestation service to run in a secure execution environment. As a consequence they mandate a trusted virtualization layer.
4 Comparison and possible synergies

Trusted computing is a holistic approach to increase the trustworthiness of today’s computer ecosystem. The needed hardware changes, in the form of a TPM and an augmented BIOS, are in place, but the extensive software and infrastructure support are lacking. Application level remote attestation requires a secure and trustworthy operating system that measures the integrity of individual programs before they are launched. The load-time binary integrity measurement as envisioned by the TCG has severe limitations. The amount of possible platform configurations (i.e., hash values of software components) quickly becomes hard to manage. The TCG approach does not give any assurance about the run-time integrity of the applications. As a result the deployment of trusted computing platform has been limited and at present the technology can be considered rather unsuccessful. A trusted virtualization layer promises to overcome some of the initial concerns. A hypervisor enforces security policies on virtual machines, such as strong isolation, and trusted computing mechanisms are used to attest the integrity of the virtualization layer. It remains to be seen whether this concept will make its way into commercial products.

On the other hand, the approach taken in the RE-TRUST project has its own problems. As identified in Deliverable D4.6 on operating system issues, the security of most RE-TRUST techniques can be subverted when the underlying operating system is under the control of the adversary. Hence, the integrity of the operating system should be remotely verified as well, preferably at run-time. Without this additional step, the RE-TRUST approach can not provide a high level of assurance. However, this need for operating system support poses a serious pitfall, which has also hindered the adaptation of trusted computing technology. Moreover, the purely software approach has scalability issues if the trusted server has to keep a lot of state information (e.g., memory layout, value of critical variables, timing information) for every client application. The trusted hardware can be used to offload this integrity verification, but this induces an additional hardware cost.

Although trusted computing and the RE-TRUST approach offer competing solutions for the remote entrusting problem, there are some prospects of synergies. A good example is the TPM timestamping technique [46, 47] developed within the RE-TRUST project. It uses trusted computing functionality to report the integrity of a trusted boot-loader in combination with the purely software based Pioneer attestation scheme [49, 48] to verify the integrity of user applications. The hybrid attestation schemes described in Section 3.2.3 explore a similar strategy. Traditional TCG attestation is used to record and report the binary identity of a hypervisor, whereas the integrity of user applications is reported by the hypervisor with an alternative attestation scheme. The hypervisor can use the RE-TRUST techniques to verify the program’s integrity at run-time.

Finally, the complementarity of both approaches was also identified in a security requirement analysis by the Open Mobile Terminal Platform (OMTP). Their TR1 doc-
RE-TRUST D4.5 – Comparative Analysis of RE-TRUST with Trusted Computing

5 lists flexible secure boot as well as run-time integrity checking as important requirements for future mobile phones. The first property is a local form of load-time attestation and hence it can be implemented with existing trusted computing mechanisms. The second security objective can be accomplished with the software techniques developed in the RE-TRUST project.

References


