Software Engineering Institute

Trust and Trusted Computing Platforms

David A. Fisher
Jonathan M. McCune
Archie D. Andrews

January 2011

TECHNICAL NOTE
CMU/SEI-2011-TN-005

CERT Program
Unlimited distribution subject to the copyright.

http://www.sei.cmu.edu

Carnegie Mellon

This report was prepared for the

SEI Administrative Agent
ESC/XPK
5 Eglin Street
Hanscom AFB, MA 01731-2100

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

This work is sponsored by the U.S. Department of Defense. The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

Copyright 2011 Carnegie Mellon University.

NO WARRANTY

THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN “AS-IS” BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

Use of any trademarks in this report is not intended in any way to infringe on the rights of the trademark holder.

Internal use. Permission to reproduce this document and to prepare derivative works from this document for internal use is granted, provided the copyright and “No Warranty” statements are included with all reproductions and derivative works.

External use. This document may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other external and/or commercial use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

This work was created in the performance of Federal Government Contract Number FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.

For information about SEI publications, please visit the library on the SEI website (www.sei.cmu.edu/library).

Table of Contents

Abstract **iii**

1 Background **1**

2 Introduction **2**

3 Trust Platform Module **3**
3.1 Primitive Operations 3
3.2 Extending Trust 3
3.2.1 Higher Level Security Functions 3
3.2.2 Integrating Trust in Software Design and Implementation 4
3.2.3 Correct Implementation of Chains of Trust 5

4 Data Protection, Device Identity, and Chains of Trust **6**
4.1 Data Protection 6
4.1.1 Microsoft BitLocker 6
4.1.2 Self-Encrypting Disks 6
4.2 Device Identity 7
4.2.1 Only Known Machines on Sensitive Networks 7
4.3 Chains of Trust 7
4.3.1 Chain of Trust 7
4.3.2 Static Root of Trust 7
4.3.3 Dynamic Root of Trust 8
4.3.4 Remote Attestation 8
4.3.5 Flicker 8

5 Implications for Use of TPM **10**

6 Implications for Trust **11**

7 Future Directions **13**

8 Summary and Conclusions **15**

References/Bibliography **17**

Abstract

Hardware-based trusted computing platforms are intended to overcome many of the problems of trust that are prominent in computing systems. In this paper, a result of the Software Engineering Institute's Independent Research and Development Project "Trusted Computing in Extreme Adversarial Environments: Using Trusted Hardware as a Foundation for Cyber Security," we discuss the capabilities and limitations of the Trusted Platform Module (TPM). We describe credential storage, device identity, chains of trust, and other techniques for extending hardware-based trust to higher levels of software-based infrastructure. We then examine the character of trust and identify strategies for increasing trust. We show why acceptance of TPM-based trust has been limited to date and suggest that broader acceptance will require more focus on traditional trust issues and on end-to-end services.

1 Background

This paper is the result of a 2010 Software Engineering Institute's Independent Research and Development (IRAD) project "Trusted Computing in Extreme Adversarial Environments: Using Trusted Hardware as a Foundation for Cyber Security." The purpose of the IRAD study is to evaluate the promise and limitations of using trusted hardware as a foundation for achieving demonstrably high assurance of end-to-end security properties of applications executing in extreme adversarial environments.

In this study, we examine the capabilities and limitations of hardware-based trusted platforms in general, and the Trusted Platform Module (TPM) from the perspective of trusted applications in particular. Through this examination, we obtain an understanding of the methods recommended and used to extend trust to higher levels of infrastructure and application software. We also examine the nature of trust and trustworthiness and draw some tentative conclusions on why the TPM has not yet had the market success one might expect. This report focuses on the TPM and trust issues.

We also examine traditional approaches to trust, as used and known to be effective in standalone applications to identify strategies for increasing trust. Comparison of the approaches of hardware-based trusted computing platforms with the proposed strategies provides insight into the gap between the promise of trusted computing platforms and the reality of trusted applications. The latter observations may help explain the limited market acceptance of trusted computing platforms and point the way toward more viable approaches from both technical and business perspectives.

2 Introduction

Trustworthiness is a measure of the integrity, ability, competence, and surety of an entity to provide a service. As the number of security risks in automated and networked systems continues to rise, assurance of the trustworthiness of systems becomes increasingly important to safely and cost effectively use services involving automation or networks.

Trust is the degree of confidence (i.e., certainty, reliance, or belief) that one has in the trustworthiness of a person, organization, application, or system to satisfy an individual's expectations in providing a particular service. The degree of confidence depends not only on the trustworthiness of the service and its provider, but also on the user's knowledge of that trustworthiness. One should trust only the trustworthy, but that is impossible without evidence of their trustworthiness.

Issues of trust and of adequate evidence of trustworthiness are complex, difficult, and until recently, seldom addressed by the developers of automated and network systems [Schneider 1999]. They are, however, a critical aspect of everyday human life. Trust involves many risks that must be managed. Automated systems and networks introduce additional trust issues, primarily in the area of security, that are uncommon in social systems alone. The use of authenticators (passwords, tokens, and biometrics), digital signatures, crypto checksums, reputation systems, and digital identity management methods, for example, are security mechanisms that are sometimes necessary for trust in automated and networked systems, but do not address the trustworthiness of the service providers nor the functionality or quality of the services themselves.

Hardware-based trusted computer platforms are intended to overcome many of the problems of trust that are prominent in computing systems [Smith 2005], [England 2004]. The TPM is a simple, passive, integrated circuit chip intended to provide several trustworthy security capabilities, to be extensible to higher levels of computer software and network infrastructure, and to create a foundation for trusted applications [Pearson 2003], [Challener 2008], [Grawrock 2009], [TCG 2007]. In practice, however, hardware-based trusted computer platforms are rarely used to develop computational or network infrastructure and have had limited market success in end applications to date, even when the TPM chip is present [Anderson 2003].

3 Trust Platform Module

The TPM is a hardware chip that is intended to enhance the level of trust that one has in certain aspects of computing systems and networks. It is intended to be a trustworthy device for certain low-level security services, including hardware-based credential storage, device identity, and software integrity. In conjunction with trustworthy software, it is intended to serve as a trustworthy base for trusted infrastructure leading to software applications that are trusted. In general, people desire that trust not exceed trustworthiness, recognize that trust can never be complete, and need levels of trust adequate for their current purpose (whatever that might be).

The TPM chip's simplicity, passiveness, and limited capacity account for its low cost and provide confidence that its design and implementation can be trusted. The higher levels of trust expected from the TPM and other hardware-based trust platforms are not realizable in software solutions alone. For example, a software implementation of key generation and confidential storage of keys will not have comparable trust levels that implementing the same capability within a passive chip will have. As a hardware device, the TPM is able to offer a more constrained application program interface (API) than a software module. Hardware that is more complex or active would have lower levels of trust because its trustworthiness would be more difficult to assess and verify.

3.1 Primitive Operations

As a simple low-capacity device, the TPM has a small number of primitive operations and limited storage. It can perform hashing; generate random numbers and asymmetric cryptographic keys; securely store keys (i.e., with confidentiality and integrity); perform asymmetric signing, encryption, and decryption operations on small blocks of data; and confirm its own identity [TCG 2007]. In theory these primitive operations can be extended to almost any well-understood security function by using a *chain of trust*, a term for bootstrapping trust in a software architecture that extends the trusted security platform to higher and higher levels, one level at a time [Arbaugh 1997].

3.2 Extending Trust

To extend trust in security mechanisms beyond the TPM requires functional and quality adequacy of the higher level security function being implemented, integrating trust in the software design and implementation, and correct implementation of the chains of trust. Each of these three is addressed below. Note, however, that all three are necessary for any use of the TPM. Any security function not built in must ultimately be correctly composed from built-in capabilities. As a passive device, the TPM can do nothing on its own; every use of the device requires software (whether in RAM or ROM), though there are viable architectures where not all of the software must be trusted. Each link in the chain of trust must involve a theoretically sound process and correct implementation of that process.

3.2.1 Higher Level Security Functions

The first steps in building higher level security functions are to implement software functions that use the TPM primitives to provide public key authentication, integrity measurement, and attestation. These functions can then be used to ensure that private keys cannot be given away or stolen, that altered code is detected, that only authorized services can use private keys, and that even constrained physical attacks are unlikely to obtain encryption keys. The TPM has some inherent capacity and performance limitations that can be overcome by the systems designer in most cases. These cases include implementing symmetric encryption on the computer system's primary CPU, where it has higher performance and can handle large blocks of data; implementing asymmetric encryption in software; and using encrypted storage of keys outside the TPM. In each of these cases the functionality enabled is based on the security provided by the TPM. These cases are examples of how developers and advocates of the TPM have done extensive work in demonstrating how it is possible to build a multitude of essential security functions from those primitive to the TPM.

3.2.2 Integrating Trust in Software Design and Implementation

Trust in the functionality and quality attributes of a software application is important independent of the security of the infrastructure required to give the user assured access to that functionality. Trust issues involve a variety of considerations beyond those of security and are addressed in Section 4. Trust issues are important not only in end-user applications, but also in each function of the infrastructure and in the software-implemented security functions. The TPM does not provide any trust functions beyond those for security (i.e., primarily confidentiality and integrity), nor does the community provide advice on how to design or implement trust at higher levels. Trust issues beyond security are seen as the responsibility of software authors at each level and for each application.

Leveraging the security functions that the TPM enables requires the software developer to plan for appropriate use. As an example, a simple capability enabled by the presence of TPMs in computing devices is their ability to use the TPM as a secure key-store. Asymmetric credentials, such as public/private keypairs for certificate-based authentication in a virtual private network (VPN), secure sockets layer (SSL), secure shell (SSH), or other protocols can be generated and maintained in the TPM. These keys have significantly better protection against attack, as the private component of the key never leaves the TPM unencrypted. Without a sophisticated hardware attack, there is simply no way to compromise the actual value of the private key. The TPM further provides access control mechanisms for such keys. While sophisticated policies based on the hash of code that has been loaded for execution[1] are possible, very simple policies based on passwords or personal identification numbers (PINs) are also possible. The TPM-based flavors of simple mechanisms that rely on the TPM for secure storage have increased security because the TPM itself implements dictionary-attack defenses[2] by dramatically reducing its response time in the presence of repeated incorrect guesses. Offline dictionary attacks[3] are prevented by the TPM because protected keys are stored in such a way that requires cooperation of the TPM to reveal them to the host platform. Thus, an attacker's best course of action becomes to attempt a hardware attack. This is a very strong security property given the prevalence of network-based attacks.

---

[1] The TPM chip's integral storage enables integrity checking of code loaded for execution by comparing the hash value of the code to the hash code of an expected version of the code. Any differences, such as if the code has been altered or malicious code inserted, will be identified and execution halted. See *Dynamics of a Trusted Platform: A Building Block Approach* [Grawrock 2008] for a complete explanation of integrity checking capabilities.

[2] *Dictionary attack* refers to guessing key information such as passwords by using programs that test every word in the dictionary. A typical defense for this type of attack is to lock down access after a certain number of failed attempts.

[3] An offline dictionary attack is one in which the attacker steals the password file and then tries to guess the passwords with no interaction with the system under attack.

Even if TPM-based keys are not integrated into a multitude of existing protocols, there is still significant value to be attained by using the TPM to represent the identity of a particular device for a particular service. Precisely because the TPM-protected keys never leave the chip unencrypted, a keypair can be used to represent the identity of a physical device. Protocols can be enhanced with such support and achieve very strong audit mechanisms. For example, if a stolen laptop is used in an attempt to access a sensitive network, the identity of that laptop may be ascertained, which can enable at least (1) definitively revoking its privileges with respect to the sensitive network and (2) helping to identify the time or event during which the laptop was stolen. A potentially effective policy for networks that handle sensitive information (whether it be inside the U.S. Department of Defense (DoD) or for compliance reasons in a commercial setting) is to allow *only known machines on sensitive networks*. TPM-based identities for machines cannot be trivially copied or spoofed like software credentials or unprotected hardware identities such as Ethernet MAC addresses.

3.2.3 Correct Implementation of Chains of Trust

A chain of trust, discussed more fully in Section 4, is a term used to describe the different trust properties in multiple layers of software in a computer system. There exist reasonably complete and rigorous processes to ensure that chains of trust are correct. Although the process of establishing and maintaining a chain of trust are theoretically sound, they are difficult and error prone in practice. Trust in the resulting software declines with each new level of layered trust, because each depends on the trustworthiness of all lower levels and such chains are brittle, i.e., any changes such as software updates will result in a broken chain. The prescribed doctrine of trusted computing platforms is that platforms must be built bottom up, beginning with an initial root of trust for measurement established in the hardware and initiated by a fixed piece of trusted code in the BIOS. This code then starts a series of measurements that measures the next code to be executed and extends a platform configuration register (PCR) in the TPM to ensure that an accurate and immutable record of the next software layer (link in the chain of trust) is recorded before transferring control to that next layer [Sailer 2004]. Control is then transferred to the measured code for execution, which, in addition to whatever functionality it otherwise provides, performs similar measurement and extension for any code it calls for execution. By this process and subsequent verification measurements in the chain of validation-execution-measurement, violations to code integrity can be detected. In theory, this process can continue through the many levels of system boot and operating system initialization, and perhaps eventually to applications. Even if done with extreme care, however, this process alone is unlikely to inspire confidence in the trustworthiness of anything as complex as an operating system. Put another way, we may learn what code was loaded, but we do not learn anything about the security properties of the loaded code.

In addition to the static root of trust mechanisms described above, where all software on the platform is measured, other approaches, such as Flicker [McCune 2008], provide protection and attestation capabilities for only a small code module. These approaches make more direct use of the TPM to provide application-level security functionality and are also discussed in Section 4.

4 Data Protection, Device Identity, and Chains of Trust

In this section we discuss three types of system security features that can be enabled with TPMs. The first two, data protection and device identity, are readily available and can be deployed today. The third, chains of trust, enjoys some limited deployment as part of Microsoft BitLocker and is readily available as an open source prototype, but currently lacks full ecosystem support from the private sector. Increased deployment of TPMs for data protection and device identity today will reduce vendors' barriers to entry and adoption if and when chain-of-trust solutions become available.

4.1 Data Protection

4.1.1 Microsoft BitLocker

Microsoft's BitLocker is a tool for full-disk encryption, but it optionally supports authorization that involves the TPM. With TPM-based authorization, portions of the Windows loader and kernel that execute during system initialization are measured and extended in the TPM's platform configuration registers (PCRs). These PCR values can be used to gate access to the TPM-based master key that must be available to fully decrypt the system's hard drive. This architecture derives security advantages from these two characteristics:

1. The TPM-based private key will never be available in the clear off-chip and is thus less susceptible to certain forms of software-based theft than a solely password-based key.
2. The software measurement values in the PCRs enable straightforward detection of certain forms of malicious modification to the early Windows software stack.

BitLocker is noteworthy in that it makes use of very specific TPM interfaces and provides a concise, useful property.

4.1.2 Self-Encrypting Disks

Many drive manufacturers now offer self-encrypting disk drives (SEDs). These drives implement encryption algorithms in their firmware and are able to operate at line speed, thereby eliminating any performance reasons to avoid full-disk encryption. Several companies offer management solutions for these drives, enabling enterprise IT departments to remain ultimately in charge of access to the data on these drives. Thus, interesting new architectures become possible, such as corporate laptops that must phone-home to decrypt sensitive files. Lost or stolen laptops can be de-privileged, so that even an attacker who guesses the user's password will be unable to decrypt the drive. Ultimately, access to encrypted partitions is based on some form of secret or credential. TPM-based credentials can help to ensure that critical data can be made accessible to only approved software configurations.

4.2 Device Identity

4.2.1 Only Known Machines on Sensitive Networks

Credentials stored in the TPM have hardware-based protection for their private keys. This is an excellent form of strong device identity and can be used to greatly strengthen protocols used for network access control. For example, existing network access control systems may be based on only a computer system's Ethernet MAC address, which can be trivially spoofed by malicious software. Other existing solutions may use cryptographic credentials, but those credentials are stored as files on the system's hard drive and handled by software where they may be exposed to malware or malicious users. With the private keys stored safely in the TPM, the chances of a software-based attack leaking a private key are effectively eliminated. Sensitive networks should implement an admission control policy based on strong, hardware-backed device identity. Machines that cannot pass this form of authentication should not be allowed on sensitive networks. A seemingly simple policy of allowing only known machines on a network can prevent the introduction of unexpected or invalid system configurations into sensitive networks and effectively close off what are today relatively easy avenues of attack.

4.3 Chains of Trust

As previously stated in section 3, trust can be extended into software by building on the primitives associated with the TPM. In this section, we discuss several strategies available today that leverage trusted computing technologies to increase the robustness and trustworthiness of commodity computing systems. We first introduce the notion of a chain of trust and remote attestation based on such a chain. We also discuss some more general design concepts that may be of interest to the developers of next-generation systems who wish to take advantage of trusted computing technologies to strengthen their applications against many common types of attack.

4.3.1 Chain of Trust

A *chain of trust* is a term used to describe the trust dependencies in a computer system consisting of multiple layers of software that have different temporal and spatial (e.g., central processing unit [CPU] privilege rings) trust properties. In TPM-style designs, the chain of trust is generally manifested as a sequence of cryptographic hash operations performed over executable code, configuration information, and data of interest.

4.3.2 Static Root of Trust

A *static root of trust* is instantiated when the TPM is activated by the platform owner and is initiated when a system is first powered on. It is intended to persist for the entire boot cycle. For the integrity of the chain to remain intact, all software must be *measured* prior to its being executed. Measurement involves performing a cryptographic hash over the item of interest and then *extending* the chain of trust with this new measurement. An aggregate value summarizing the chain resides in the platform's TPM chip.

There are several drawbacks to integrity measurement architectures based on a static root of trust. The first is code size. Even if the measurement chain is unbroken, the sheer volume of code that executes on a typical system today renders its security properties unverifiable. The second is the need to maintain an unbroken measurement chain. If any code executes that was not first measured, the chain breaks. Likewise, if a program interprets a configuration file that is not measured, then the chain is broken. Even data that is processed by the program may need to be measured to be fully able to conclude whether the system is in a trustworthy state. Thus, static root of trust is most valuable for systems that perform very well-defined, specific tasks. A VPN gateway is an example of such a system, i.e., the static chain of trust begins in the firmware and extends through the BIOS to the boot loader, then through the operating system (OS) kernel and the minimum set of services required to provide a VPN service. A virtualization-based platform that has the ability to execute multiple, distinct virtual machine images may use static root of trust to verify the integrity of the hypervisor and associated virtualization infrastructure.

4.3.3 Dynamic Root of Trust

A *dynamic root of trust* (DRT) differs from a static root of trust in that it can be created on-demand, at any time. Platforms supporting dynamic root of trust have more sophisticated CPU and chipset extensions to support the creation of an isolated execution environment. A dynamic root of trust atomically reinitializes the CPU to a known-good state, reconfigures the system's memory management unit to isolate certain regions of memory from potentially malicious direct memory access (DMA) capable peripherals, sends a special signal to the platform's TPM chip to indicate a DRT operation, and sends the contents of the code to be executed inside the isolated environment to the TPM to be measured (for use in subsequent integrity checks or data sealing operations). In this case, the chain of trust for the new code to be measured is quite short. AMD and Intel (the leading x86-class CPU manufacturers) implement this feature differently, but in both cases there are fewer than ten measurements to process, as opposed to hundreds or thousands that are required to implement static root of trust architectures.

4.3.4 Remote Attestation

TPM-equipped computer systems with software that is architected to accumulate measurements into their TPM's PCRs can invoke network protocols to perform *remote attestation.* An attestation is a digitally signed set of cryptographic hash values that describe the software configuration of the system of interest. The signing key used to produce an attestation resides exclusively in the TPM on the system of interest, so that the recipient of an attestation can be certain as to the originator of the attestation message. This enables the remote *verifier* to cross-reference the received attestation with a database of known-good or expected values, and to potentially draw conclusions as to the trustworthiness of the system of interest. A simple use case may be the conclusion that the system is equipped with current (patched), best-known versions of the intended applications. A more sophisticated use case may conclude that the system is executing precisely the intended software configuration, where the intended configuration was selected only after rigorous formal analysis of the constituent applications and the trust implications of their composition on a single system.

4.3.5 Flicker

Flicker is a research system developed at Carnegie Mellon University that is applicable on systems that offer support for dynamic root of trust. Flicker enables developers to write applications that can manage security-sensitive operations in hardware-enforced isolation from all other code and devices on the system, and to generate attestations that can potentially convince an external party or remote system that this was the case. While Flicker is a platform for writing applications and not an end-user product, developers who are writing applications that have important data to protect may be able to significantly harden their applications by leveraging the Flicker architecture or similar designs.

5 Implications for Use of TPM

Trusted security platforms and the TPM in particular offer a level of security capability that cannot be achieved in software alone. It is inexpensive and readily available. It is installed in many computer systems and sometimes enabled by the manufacturer. The hardware itself is generally trusted by the developers who use it.

Despite these desirable attributes, in practice the TPM is rarely used even when the chip is present. It is rarely used in the development of computational or network infrastructure and has had limited market penetration in end-user applications. Enabling installed chips may be inconvenient or difficult, but a more likely explanation is a lack of vendor-provided software to leverage the TPM. Software development at very low levels of the operating system is required for vendor-provided software to leverage the capabilities of the TPM. Such development requires operating system, systems programming, and security expertise possessed by few developers. These requirements provide significant technical and cost barriers not only for end users but also for application developers.

Furthermore, end users do not seek low-level security functions. They desire useful end-to-end security capabilities that can be used without additional software development. Device identity, self-encrypting disks, and BitLocker are examples of successful user-level security capabilities that can be used and adopted today.

At the same time, end users always need trustworthy applications. When applications involve software infrastructure (including operating systems or libraries) in their implementations or intermediaries (including software or communications) in delivery of their services, issues of identity of the services and integrity of communications arise. Users care about the trustworthiness of the applications they use. They care about the integrity, ability, competence, availability, and surety of a system to provide expected services. They do not care nor need to know how security is achieved within their applications. Exploiting the TPM to achieve trustworthy products should be the responsibility of infrastructure and application providers. As long as that responsibility is left to users, the TPM is not likely to receive its widespread use.

Issues of infrastructure security and use of the TPM are only a small part of users' concerns for trust. Creditable evidence of trustworthiness of services, automated support for assessing and validating trust, and an effective set of strategies for increasing and maintaining trust and trustworthiness are needed.

Because IT infrastructure often evolves organically without predetermined knowledge of its end use, the infrastructure is often at odds with the effective use of the TPM. Effective use of hardware-based roots of trust requires a clear vision of full-system architecture as it relates to the use of security to enable end-to-end trust. Broader acceptance and more effective business models for exploiting trusted computing platforms will also require more focus on traditional trust issues and on end-to-end services. From a trust perspective, the TPM could help end users by allowing applications to be treated more as black boxes.

6 Implications for Trust

TPM-enabled hardware platforms are readily available but seldom used, with an estimated 250,000,000 installed but less than one percent activated [Sprague 2010]. As the community realizes the TPM is capable of enhancing some, but not all, measures of a systems' trustworthiness, there is a need to investigate the nature of the trust that is either not supported or not easily exploited by the capability provided by the TPM, and to identify those areas that should be further addressed.

Trust requires assessment of evidence of trustworthiness combined with levels of confidence in those assessments. Without confident assessment of trustworthiness, there is nothing meaningful to compare with our suppositions about needs. *An appropriate combination of well-reasoned needs and justified trust can make the difference in survival, safety, and cost effectiveness in any activity.*

The service provider can provide guidance in the form of claims of functionality and quality, and may even provide evidence of trustworthiness, but the services actually used may or may not be among those claimed. The user of a service cares about the trustworthiness of those aspects of a service that he or she depends on, uses, or intends to use. Without a better option, the user determines what functionality and quality can be trusted using an intuitive assessment and validation process that often yields results different from those claimed. This is, for example, why programmers often exploit software bugs or undocumented APIs to obtain functionality or quality that is otherwise difficult to obtain. The user cares about the integrity, ability, competence, and surety of an entity to provide a service and can be quite upset when a service is changed (even if it is a bug fix). This further illustrates users' focus on their primary task. When users are comfortable getting their work done with a buggy program, any change may be disruptive or unwelcomed. From the point of view of trust, it is the user, and not the service provider, who determines what constitutes an adequately trustworthy service.

Until recently, the products and services that users trust have often been ones for which they are in direct contact with their providers. Sometimes there are infrastructure and other intermediating services that must be trusted. The use of infrastructure, communications, or other intermediating services offers the possibility of altered functionality or degraded quality of a service, typically in the form of corrupted information or interrupted service. Service providers often view such issues as outside their control or concern. Infrastructure providers (such as OS and communication providers) see them as security issues (i.e., information integrity and availability of service, respectively). *Users prefer to view a remote service together with any intervening infrastructure as a single combined service.* Their trust is in the resulting end-to-end service and not in its individual components. For example, a user might prefer a less capable end service with reliable communications over a more capable end service with frequent data corruption.

Because trust often involves services used or provided by humans, even in automated and networked systems, any effective solution to issues of trust must encompass the human side of the trust equation, i.e. the socio-technical systems, properties, and issues. At the same time, any solution that includes multiple layers of software or hardware or is implemented from several automated components must address the interdependent issues of trust among the automated components.

Problems with infrastructure (including network communications, operating systems, and other software layers), where the infrastructure is not the end service, are most commonly viewed as security issues. Integrity of information and availability of services (together with confidentiality) are the defining focus of security. In automated and networked systems, there are always infrastructure or other intermediating services, making security a critical factor in such systems. That the TPM is intended to provide security services at the platform (i.e., infrastructure) level indicates this recognition. Like other services, security services can never be fully assured and require trust beyond that associated with the end service.

Ultimately, trust assessment is the responsibility of the end user, whether the end user of an application, an application using the OS and application components, or higher-level OS functions using lower-level OS functions. The cost of assessing the trustworthiness of a service is often proportional to the complexity of the service as seen by the end user, rather than the complexity of its implementation. Consequently, requiring the end user to evaluate the trustworthiness of a service's components to determine the trustworthiness of the service generally imposes additional cost and effort on the end user. In particular, the user cares about whether a service uses a hardware-based root of trust (only because its absence ensures certain vulnerability), but from a trust perspective does not care whether a vulnerability results from poor use of a hardware root of trust or from its absence.

7 Future Directions

The business model that has been successful for engaging the TPM is to provide end-to-end security products that happen to leverage TPMs. Other models have not been widely adopted and if fully realized would result only in security products, not necessarily more secure or trustworthy applications. A business model of potentially greater effectiveness is the use of the TPM internal to applications, products, or systems to maintain and preserve trustworthiness that could otherwise be undermined by security flaws in lower-level infrastructure. This approach could also be used to provide more secure and trustworthy infrastructure. Research is needed to define and assess business strategies for broader and more effective exploitation of hardware-based trusted platforms. As the installed base of TPMs continues to grow, there are more and more opportunities for private industry to offer applications that are inherently more trustworthy by leveraging the TPM.

Trust is of critical importance in all human activity. Without trust, we can accomplish nothing. With unjustified trust, nothing can be adequately assured, safe, or efficient. Automated systems and networks impose additional problems of trust, but do not traditionally provide adequate support for trust. Automated support for trust is essential for effective use of automated systems and networks. The TPM and other hardware-based trust mechanisms are a step in the right direction but inadequate in current practice. While they provide automated support for certain security aspects of trust, issues of trust go far beyond security. There is a need for investigation and understanding of the potential role of technology in supporting other aspects of trust.

Trust could be an important new domain of computational and network technology. Obtaining cost-effective solutions involving automated and networked systems is a longstanding problem in information assurance, infrastructure protection, software engineering, and everyday life. Solutions require justified confidence in the integrity, ability, competence, and surety of an entity to provide needed services of adequate functionality and quality where and when needed.

Support for trust will require more rigorous understanding of trust and trustworthiness, effective strategies for achieving trustworthiness and assessing trust, and development of automated tools for trust. Research in these areas will likely borrow from other domains with overlapping concerns. These domains include, most conspicuously, security, survivability, dependability, emergent behavior, and modeling and simulation.

The security, survivability, and dependability communities each provide partial solutions. Security, with its focus on availability, integrity, and confidentiality, is recognized as an often critical aspect of solutions, but security methods are seldom tied to the specific needs of the application. Security cannot provide complete solutions; it is at best inefficient when applied without adaptation to changing circumstances and specific needs, and when in a one-size-fits-all form can interfere with work and undermine effective solutions.

Survivability [Lipson 2000] was an attempt to overcome the shortcomings of fortress model security by focusing on the end goals of mission satisfaction, survivability, and system evolution [Lipson 2006], and the security technologies that contribute to those end goals. Survivability never gained traction because security responsibilities were entrenched in organizations devoid of responsibility for mission success, because there was a dearth of both survivability methods and promising research, and because, like security, survivability does not offer the potential for complete solutions.

Dependability is an all-encompassing conglomeration of technologies promising complete solutions to any problem in automated systems, but only in an idealized world that lacks competition, intelligent adversaries, and need for adaptation and evolution [Avizienis 2001]. It seeks correct solutions and disdains informal and empirical approaches.

Emergent behavior is inherent in all complex and networked systems [Fisher 1999]. Trusted solutions cannot be effective without recognizing and addressing emergent effects. In network security applications, emergent behavior may include cascade effects, epidemics, inevitable accidents, phase shifts, and sometimes even new emergent domains with their own defining properties. It may also be possible to exploit emergent effects [Fisher 2006] to enable or encourage trustworthiness.

Modeling and simulation are essential research tools in any domain that is complex, safety critical, or prohibitively expensive to experiment with real systems [Anderson 2006]. Modeling and simulation are needed for the design of new systems and analysis of existing systems. Depending on the nature of the questions being asked, discrete event simulations, dynamic simulations, or systems dynamics (probabilistic) simulations may be most appropriate. The more that models can be abstractly specified and reused [Fisher 2010], the more simulations can be separated from models, and the more models and simulations are one-to-one with the systems being modeled, the less error prone and more accurate will be the simulation results.

Finally, EAEs involving antagonists with nation-state levels of resources and malicious intent provide an ideal context for testing and validating research results in the above identified domains. They illustrate the importance not only of resisting attacks and mitigating their effects, but also of discovering compromises when they occur and continuously evolving systems to more effective and trustworthy solutions. They also point out that, in the long run, victory is impossible without a sustainable asymmetric advantage.

8 Summary and Conclusions

The TPM is a simple, passive, integrated circuit chip intended to serve as a hardware root of trust for trusted infrastructure leading to software applications that are trusted. It is an inexpensive enabler for credential storage, device identity, and trusted applications and provides security capabilities that cannot be provided by software alone with the same degree of confidence. The TPM derives both trust and trustworthiness from its simple passive character. A more complex device, an active one with general purpose computing capabilities, or one that shares registers with a CPU could not engender similar levels of trust.

Although the TPM is generally trusted by developers who use it, to date it is rarely integrated within the computational or network infrastructure, and has had limited market success in end applications even when the chip is present. To understand why, we examined the methods recommended and used for extending trust from the TPM to higher levels of application software. Building provably correct chains of trust from BIOS through several levels of operating systems and into applications is theoretically sound, but is a tedious and brittle process that must be redone whenever any software along the way changes. More practical methods are used but with increased vulnerabilities and often lower levels of trust. The measurement technique used in conjunction with the TPM is vulnerable because it cannot detect changes made and restored between measurements. Neither does measurement provide protection for mutable storage.

A viable business model for exploiting trusted platform technology in a general purpose platform has not yet emerged. However, as the level of deployed TPMs increases, so do the opportunities for commercial products dependent on the existence of deployed TPMs. A key appears to be integration and support by operating systems and application use of the resulting APIs. As a practical matter, applications developers and end users will not leverage the TPM unless its functionality is easily accessible. They cannot be expected to develop the chains of trusted software required at the operating system level. More encouraging are increased use of TPM chips in certain dedicated security applications such as Microsoft BitLocker, storage of PKI private keys and other credentials (e.g., biometric identifiers), and potentially secure machine identities on sensitive networks. Part of the problem may be that without a large installed base of TPMs, there is little incentive for application developers, and without developers' demand, there is little incentive for the OS to support them or for more systems to install them. The implication to DoD and others with a currently large installed base of TPMs is that the majority of those already deployed will unlikely be used without a critical mass of installations that triggers a broader market of applications and OS support. Price Waterhouse Coopers' recent decision to enable TPM-based credentials for 150,000 users [Messmer 2010] may be an indication of this trend.

There is also a need for hardware support for security beyond that provided by currently available trusted platform devices. The potential and realized benefits of the TPM derive from the ability to ensure integrity of information, processes, or identity either by physical isolation (e.g., key storage in the TPM) or logical isolation (e.g., encrypted communication). Hardware support for isolation of storage and communication is needed at many levels, including CPU register sharing through task switches, shared caches, uninitialized portions of memory pages, and DMA access. Hardware could assist operating systems isolating applications, eliminating security vulnerabilities inherent in current CPU architectures, and providing user-level security functions that are easily accessible through APIs. Hardware mechanisms to support dynamic roots of trust, to eliminate software intervention at BIOS and the lowest levels of the OS, to provide immutable storage as an alternative to measurement, and to facilitate logically atomic sequences of operations have the promise to make chains of trust less brittle and more trustworthy.

Our effort also looked at the character of trust and trustworthiness. Trusted platform technologies are bottom-up in character and in the limit can provide only a secure platform for needed end-to-end trusted solutions. Although infrastructure security is critical to any trust context that depends on infrastructure, information technology also is needed to enable and support trustworthiness and trust in end-user applications and services. While trust and trustworthiness have been widely exploited in other domains, they have received little attention in the world of automated and networked systems.

Research is needed to develop trust technologies applicable to automated systems. Trust technology has the potential to overcome existing limitations of survivability, security, and dependability. An effective trust technology will focus on mission fulfillment, survivability, and evolution of automated and networked systems. It will employ security methods but only when and where they are cost effectively needed. It will embrace many of the quality objectives of dependability but with greater adaptability and realism. It will seek practical cooperative solutions that are appropriate for competitive and adversarial environments. It will focus on end-to-end solutions specialized to particular needs. It will emulate and adapt proven trust methods from everyday life.

References/Bibliography

URLs are valid as of the publication date of this document.

[Anderson 2003]
Anderson, Ross. *Trusted Computing' Frequently Asked Questions*, University of Cambridge Computer Lab, Version 1.1. http://www.cl.cam.ac.uk/~rja14/tcpa-faq.html (2003).

[Anderson 2006]
Anderson, W.; Brownsword, L.; Fisher, D.; & Garcia S. *Transitionability of Complex Modeling and Simulation Approaches to Software-Intensive Systems Development* (CMU/SEI-2006-SR-018). Software Engineering Institute, Carnegie Mellon University, December 2006. http://www.sei.cmu.edu/library/abstracts/reports/06sr017.cfm

[Arbaugh 1997]
Arbaugh, W. A.; Farber, D. J.; and Smith, J. M. "A reliable bootstrap architecture." *Proceedings of the IEEE Symposium on Security and Privacy*, 1997.

[Avizienis 2001]
Avizienis, A.; Laprie, J. C.; & Randell B. *Fundamental Concepts of Dependability* (Research Report No 1145). LAAS-CNRS, April 2001.

[Chanllener 2008]
Challener, David; Yoder, Kent; Catherman, Ryan; Safford, David; van Doorn, Leendert; *A Practical Guide to Trusted Computing,* Pearson 2008

[England 2003]
England, Paul; Lampson, Butler; Manferdelli, John; Peinado, Marcu; Willman, Bryan; "A Trusted Open Platform", *Computer*, Vol 36, No.7, (2003) pp. 55-62.

[Fisher 2006]
Fisher, David A. *An Emergent Perspective on Interoperation in Systems of Systems* (CMU/SEI-2006-TR-003 ESC-TR-2006-003). Software Engineering Institute, Carnegie Mellon University, March 2006. http://www.sei.cmu.edu/library/abstracts/reports/06tr003.cfm

[Fisher 2010]
Fisher, David A. "Reusable Specification of Agent-Based Models," 154-159. *19th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE)*, 2010.

[Fisher 1999]
Fisher, David & Lipson, Howard. "Emergent Algorithms - A New Method for Enhancing Survivability in Unbounded Systems." *Proceedings of the 32nd Annual Hawaii International Conference on System Sciences (HICSS-32)*. Maui, HI, January 5-8, 1999

[Grawrock 2008]
Grawrock, David. *Dynamics of a Trusted Platform: A Building Block Approach*. Intel Press, 2008.

[Lipson 2000]
Lipson, Howard & Fisher, David. "Survivability: A New Technical and Business Perspective on Security," 33-39. *Proceedings of the 1999 New Security Paradigms Workshop*. Caledon Hills, Ontario, Canada, Sept. 22-24, 1999. New York: Association for Computing Machinery, 2000.

[Lipson 2006]
Lipson, Howard. *Evolutionary Systems Design: Recognizing Changes in Security and Survivability Risks* (CMU/SEI-2006-TN-027). Software Engineering Institute, Carnegie Mellon University, 2006. http://www.sei.cmu.edu/library/abstracts/reports/06tn027.cfm

[McCune 2008]
McCune, J. M. & et.al. "Flicker: an execution infrastructure for TCB minimization," 315-328. *Proceedings of the 3rd ACM SIGOPS/EuroSys European Conference on Computer Systems 2008*. Glasgow, Scotland UK, April 1-4, 2008.

[Messmer 2010]
Messmer, E. "PwC lauds Trusted Platform Module for strong authentication: Firm migrating 150,000 users to TPM-based storage of private keys." *Network World* (September 15, 2010.)

[Pearson 2003]
Pearson, Siani, ed.; Blabcheff, B.; Chen, L.; Plaquin, D.; & Proudler, G. *Trusted Computing Platforms: TCPA Technology in Context.* Prentice Hall, 2003.

[Sailer 2004]
Sailer, Reiner; Zhang, Xiaolan; Jaeger, Trent; & van Doorn, Leendert. "Design and Implementation of a TCG-based Integrity Measurement Architecture," 223–238. *Proceedings of the 13th USENIX Security Symposium.* 2004.

[Schneider 1999]
Schneider, Fred B., Editor, *Trust in Cyberspace,* Committee on Information Systems Trustworthiness, National Research Council, 1999.

[Smith 2005]
Smith, Sean W. *Trusted Computing Platforms: Design and Applications*. Springer, 2005.

[Sprague 2010]
Sprague, Steven K. *Cyber NSF Trusted Information Workshop* (remarks). Carnegie Mellon, Pittsburgh, PA; June 7-10, 2010.

[TCG 2007]
Trusted Computing Group. *TPM Main: Part 1 Design Principles, Specification Version 1.2 Revision 103.* http://www.trustedcomputinggroup.org/files/resource_files/ACD19914-1D09-3519-ADA64741A1A15795/mainP1DPrev103.zip (2007).

REPORT DOCUMENTATION PAGE			*Form Approved* *OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.			
1. **AGENCY USE ONLY** (Leave Blank)	2. **REPORT DATE** January 2011		3. **REPORT TYPE AND DATES COVERED** Final
4. **TITLE AND SUBTITLE** Trust and Trusted Computing Platforms			5. **FUNDING NUMBERS** FA8721-05-C-0003
6. **AUTHOR(S)** David A. Fisher, Jonathan M. McCune, Archie D. Andrews			
7. **PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)** Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213			8. **PERFORMING ORGANIZATION REPORT NUMBER** CMU/SEI-2011-TN-005
9. **SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)** HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116			10. **SPONSORING/MONITORING AGENCY REPORT NUMBER** N/A
11. **SUPPLEMENTARY NOTES**			
12A **DISTRIBUTION/AVAILABILITY STATEMENT** Unclassified/Unlimited, DTIC, NTIS			12B **DISTRIBUTION CODE**
13. **ABSTRACT (MAXIMUM 200 WORDS)** Hardware-based trusted computing platforms are intended to overcome many of the problems of trust that are prominent in computing systems. In this paper, a result of the Software Engineering Institute's Independent Research and Development Project "Trusted Computing in Extreme Adversarial Environments: Using Trusted Hardware as a Foundation for Cyber Security," we discuss the capabilities and limitations of the Trusted Platform Module (TPM). We describe credential storage, device identity, chains of trust, and other techniques for extending hardware-based trust to higher levels of software-based infrastructure. We then examine the character of trust and identify strategies for increasing trust. We show why acceptance of TPM-based trust has been limited to date and suggest that broader acceptance will require more focus on traditional trust issues and on end-to-end services.			
14. **SUBJECT TERMS** trusted computing, trusted platform module, cybersecurity, secure computing, chains of trust, hardware based trust			15. **NUMBER OF PAGES** 26
16. **PRICE CODE**			
17. **SECURITY CLASSIFICATION OF REPORT** Unclassified	18. **SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	19. **SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	20. **LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. Z39-18
298-102