Trusted Platform Module – A Survey

Kenneth Ezirim, Wai Khoo, George Koumantaris, Raymond Law, and Irippuge Milinda Perera

The Graduate Center of CUNY
{kezirim,wkhoo,gkoumantaris,rlaw,iperera}@gc.cuny.edu

October 22, 2012

Abstract. Recently, computer security has been trending toward hardware security, notably Trusted Platform Module (TPM). This paper describes some of the recent research of TPM, giving a high level idea of how it works and its inception into mainstream security. This paper also describes how some of its unique features make it a secure trusted module. The TPM architecture and functionality will also be presented. Lastly, we describe some security issues of TPM. Issues such as how to bootstrap trust in a computer with TPM and the security of authorization data in TPM. The guideline of TPM specifies its essential security and functionality, but leaves the implementations up to the programmers. Therefore, this paper does not compare various implementations of TPMs.

1 Introduction

The Trusted Platform module (TPM) is a low cost security module that delivers the basis of a safe computing environment. It is typically implemented as a tamper resistant integrated circuit (IC). It is designed to be a building block for trusted computing. Unlike smart cards, the TPM is bound to a specific platform. It is essential to understand the fundamental design objectives of trusted computing In order to completely understand the development of the TPM.

Trusted computing allows users to assess the trustworthiness of computers with which they interact and create a foundation of trust for software processes [1]. The Trusted Computing Group (TCG) has attempted to describe what trusted computing mean and in the process it produced a series of standards to be used in order to design trusted platforms. The TSG defines trusted computing as the expectation that "a device will behave in a particular manner for a specific purpose" [2]. Their definition really claims the fact that whoever interacts with a trusted device can be given assurance that the device will behave exactly as they expect it to. The purpose of the Trusted Platform module (TPM) is to provide this assurance to the client and the users interacting with the client [1].

The TPM provides a fundamental set of security features which have been defined by the Trusted Computing Group TCG. A trusted platform should provide at least three basic features: protected capabilities, integrity measurement, storage and reporting [2, 3].

Protected Capabilities are a set of commands with exclusive authorization to access protected locations such as memory, and registers [3]. Accessing these protected locations makes it safe to operate on sensitive data. Other protected capabilities can include additional security functionality such as random number generation, sealing data to system state or cryptographic key [3].

Integrity Measurement is the process of obtaining metrics of platform characteristics that affect the trustworthiness/integrity of a platform. Integrity storage and reporting is the process of storing those metrics and summarizing/presenting their results [3]. The preliminary point of measurement is called the root of trust. A static root of trust begins measuring from a well-known starting state

such as a power on self-test POST [3]. A dynamic root of trust for measurement changes from an un-trusted state to one that is trusted. A platform may be allowed to enter any state possible as well as undesirable or insecure states, but it cannot lie about states it was in or not in [3].

The TCG specifications enable more secure computing environment and the Trusted Platform Module (TPM) is the building block to achieve its goals. The adoption of Trusted Computing using the TPM is a significant step toward refining confidence in business over the Internet and allows existing applications to benefit from enhanced security.

2 Recent History of TPM

The TPM (Trusted Platform Module) is a required step to guarantee safer computing in multiple environments. Through the years TPM had evolved from creating a platform for trusted PC computing, to integrating the same method across a wide variety of uses. According to market research reports in [2,4], more than 100 million branded PCs and laptops with TPM's were sold just in 2007 and nearly 250 million PCs were shipped with TPM hardware by 2009. Currently TPM is used by nearly all PC and notebook manufacturers, primarily offered on professional product lines [4]. The TCG claims the technology has reached its tipping point with TPMs now in more than 600 million PCs [4].

Hardware-based security may finally become a practical option for organizations around the globe, in large part due to TCG founder member, Microsoft. Currently a number of applications are based on TPM, such as secure email file and software encryption [5]. TPM's are also included in hard drives and mobile devices. Furthermore TPM's reliability is tested while its technology is emerging in to different areas such as PC computing and mobile computing.

In March 2009, hard drive suppliers started shipping SED's (self-encrypting hard drive) based on the TCG's specifications [1]. SED is a self-encrypting hard drive based on the TCG's specifications with a circuit built into the disk drive controller chip. SED provides hardware-based data security by encrypting data as it is written to the drive, then decrypting the data with the key as the data is retrieved [1]. The encryption key used in SEDs is called the Media Encryption Key (MEK). Locking and unlocking a drive requires another key, called the Key Encryption Key (KEK) supplied by the user [6].

Windows 8 will have a big role for TPM, in the operating system to date. A feature of Windows 8 will be the secure boot [7]. This feature will measure the BIOS of a machine and report remotely to the enterprise network. Then the enterprise network will certify that no malware changed the bios since the last time the computer was on the network [3, 5].

In software security, the TPM chip stores the signatures of every piece of software on the machine and identifies abnormal activity [3]. For example,if an organization has the signatures of their legitimate software in the TPM they can determine if the actual software trying to run matches that signature. The organization can discover any changes and if the signatures don't match, then the IT department will be warned, permitting the organization to eliminate the thread.

Users can easily encrypt files, folders and emails as well as more securely manage passwords storing the keys associated with them securely in the TPM chip [6]. For multi-factor authentication requirements, the TPM complements fingerprint readers and stores the keys associated with them. Computer manufacturers provide detailed how-to instructions to enable and use the TPM with their devices [2, 4].

According to Mr. Steven Sprague from the Armed Forces Communications and Electronics Association (AFCEA), TPM will redefine a transition of the network architecture from a network

based on connections, to a network based on identity. The TPM chip helps to power self-encrypting, solid-state electrical disk drives. The self-encryption makes it possible to remotely disable a device and, if needed, erase critical data [8]. The integrity of the device can also be evaluated by a remote service before granting it access to any resources. The new Windows 8 phones will utilize TPM to protect the device's and user's identity [1].

Recently hackers were able to read the data stored on the TPM chip, for instance cryptographic keys (RSA, DES) such as those also used by Microsoft's BitLocker [5, 8]. One incident back in 2010 was conducted by Christopher Tarnovsky of Flylogic Engineering. He managed to figure out a processor in the "SLE 66CLX360PE" family used in the TPM and extract the actual chip from the housing in a special lab using various procedures that involved liquids and gases. The manufacturer of the TPM, Infineon claims that that the necessary steps to hack the device aren't easy to reproduce and require a considerable amount of special equipment [8]. In Mr. Tarnosky's incident there was a 200,000 investment put in to it. But in the case of corporate and government espionage were information is well worth the risk of such an investment, TPM security chips will be compromised.

3 Functionality of TPM

The TPM provides an entire suite of tools used for secure authentication. All of the ideas behind the TPM are based on moving security onto hardware. Existing security systems have an inherent flaw, in that they run on top of unknown hardware. An unsecure system on an otherwise secure network, compromises everything.

Key encryption is a prime example. Anyone with the private key has full authenticated access to a network. If even one system gets compromised, even for just a moment, existing keys can easily be copied over and used for an attack later on. Even worse there is no guaranteed way to tell that your keys have been copied, as they are just another file. Having a file based approached for security won't work.

Systems with TPM instead store sensitive data in a secure location on a separate piece of hardware. The TPM chip uniquely identifies the hardware and does not allow sensitive data like keys to leave the TPM. Sensitive data can be used from within the TPM, but cannot be directly accessed outside of it. Even if someone managed to get unrestricted access to the computer, they would not have access to the sensitive data stored on the TPM.

The basic premise for the TPM is the same idea behind mobile phones. The TPM is a chip that authenticates the hardware itself. The chip is designed such that only a small subset of safe actions are allowed. (ex. Encrypting a message with a key is allowed, however access to the key directly is restricted) Since most attacks originate from unknown hardware, being able to identify the device eliminates the possibility of someone stealing a key and reusing it later on. But this is not the only benefit of TPM's. TPM's provides an entire cryptographic suite of tools including secure storage, random number generators, several key engines, and even registers for processing from within the TPM.

Even though the details of TPM implementations can be done in hardware or software, the main building blocks of TPM however, are specified by TCG as shown in Fig. 1.

3.1 I/O

The I/O block is the bridge between internal and external components. Not only does the I/O block manages information flow between components via the bus, it also enforces access policies to

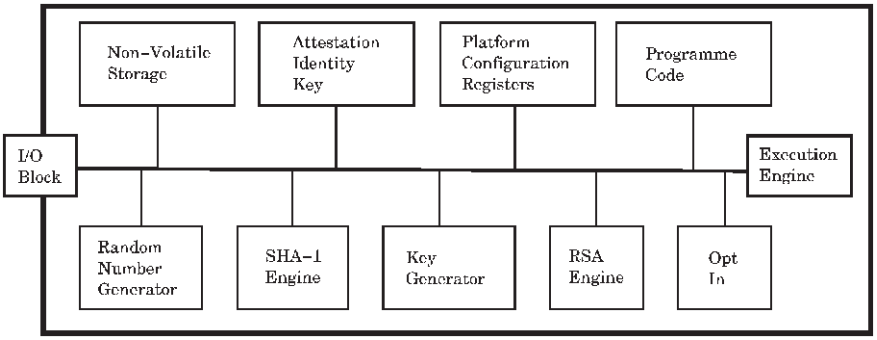

Fig. 1. TPM Building Blocks [9]

various components [9, 10]. The policies are maintained by the opt-in block, which is discussed in Section 3.11.

3.2 Non-Volatile Storage

This non-volatile memory stores several long-term (embedded) keys and authentication credentials such as Endorsement Key (EK), Storage Root Key (SRK), owner's password and persistent flags [10]

The SRK naturally is the root of this secure storage and thus manages it. The EK, however, is a unique feature in TPM and should deserve more details. In order for TPM to operate, the EK pair must be embedded in it; the private key is permenantly embedded in it (i.e. unique to each TPM and thus the platform). The public key, however, is stored in a certificate and it is only used in a limited number of operations. EK is used to generate an alias, Attestation Identity Keys or AIKs (Section 3.3), which are used for routine operation. EK is also used for encrypting data sent to the TPM during the ownership process. The EK pair is generally provided by the manufacturers before shipping [9].

A certificate can then be created from the pair which contain the public key and security properties of the TPM. This certificate is generally signed by Trusted Platform Module Entity (TPME), a certification authority, who can verify the certificate contains a public key whose corresponding private key is stored in the TPM with the specified security properties [9]. Essentially to verify that the EK pair are stored in a genuine TPM. Users who wish to customize their own security can delete and re-install the EK. However, they need to make sure that their certificate is signed by the proper TPME.

3.3 Attestation Identity Keys (AIK)

As described in Section 3.2, AIK is generated from EK which is stored in non-volatile memory. However, AIKs may be stored outside the TPM securely. In fact, TCG recommends it and states that the TPM must provide enough room in volatile memory where one or more AIKs can be loaded to speed up execution [9, 10]. Furthermore, each TPM can generate multiple AIKs. This feature allows a user to be anonymous [9].

3.4 Platform Configuration Registers (PCR)

PCRs are a set of registers that store *integrity metrics*, which is an unique feature for TPM. The metrics measure the integrity of any code before launching, ranging from BIOS to application codes [9].

Like AIKs, PCRs can be implemented in either volatile or non-volatile storage. One strict requirement is that these registers must be reset at system restart or whenever there is a power lost. The reason behind this is to ensure PCRs contain the most recent integrity metric should a system reconfigured and rebooted [9].

The standards require TPMs to have at least 16 PCRs of size 20 bytes. Registers 0 - 7 are reserved for TPM use. Registers 8 - 15 are free for any applications use, including operating system [9, 10].

3.5 Program Code

Program code contains the firmware that is used to initialize the device. Because of its purpose, it is permanently stored on the tamper proof TPM and assume to be trustworthy, in the sense that if it fails, all security policy will be broken. Then, the program code is the obvious "root of trust" for integrity measurements, which is also known as Core Root of Trust for Measurement (CRTM) [9].

3.6 Execution Engine

The execution engine simply executes the program code described in Section 3.5, which perform initialization and measurement taking [9, 10].

3.7 Random Number Generator (RNG)

The standard dictates that the random number generator is to be seeded by a true random bit stream. The random numbers produced are used to generate cryptographic keys, nonce creation, and to strenghten passwords [9, 10].

3.8 SHA-1 Engine

As the name implies, this engine implements secure hash algorithm, SHA-1, which hashes the input data and produces a 20-byte digest. It is used for digital signatures and generation of hash of key for number of cryptographic procedures [9, 10].

3.9 RSA Key Generation

As we all know, key generation can be computationally intensive, especially with the RSA algorithm. Thus, the standard requires the TPM to support keys up to 2048 bit modulus [10].

3.10 RSA Engine

Because of its complexity, RSA is housed in its dedicated engine. Like RSA, this engine is used for signing, encryption, and decryption. Note that encryption and decryption are done by separate storage key pairs [9].

3.11 Opt-In

As the name implies, the user has to opt-in to use the TPM, which means that the user has to take ownership and configure the TPM. In the process of taking ownership, the TPM will transitions through several states as described below. Changing the state of these flags requires authorization. The opt-in block ultimately provides mechanisms and protection to maintain the TPM state via the state of these flags.

TPM Operational States In short, the TPM can exists in a range of states from disabled (and deactivated) to fully enabled and ready for ownership.

disabled/enabled All operations are restricted except reporting TPM capabilities and accepting updates to the PCRs.

activated/deactivated The subtle difference between this operational variable and the previous one is that when TPM is deactivated, it will respond to a change of state or owner.

owned/unowned In an owned state, it means that a key pair has been established and the owner can perform any operations on TPM, including state change.

4 Security Issues of TPM

In this section, we survey several known security issues related to TPMs. The first issue we present is the most fundamental problem faced by the users of TPMs: how to bootstrap trust in the computer containing the TPM. Followed by that, we present two issues related to the TPM "authorization data" or authdata as mentioned in the TCG documentations. We also discuss security issues associated with the use of the TPM in authenticating processes running on a platform and the security loophole that allows attackers to reset the secured measurements stored in the PCRs.

4.1 Challenges in Bootstrapping Trust in a TPM

As we have discussed earlier, one of the main functions of the TPMs is to establish trust in the software executing on a computer. But, a crucial aspect the TPM specifications have overlooked is how to bootstrap trust in computer containing the TPM to begin with. For example, if a user does not trust his computer, he cannot perform most of the activities such as doing online banking, reading/writing confidential documents, visiting confidential websites, etc., regardless of the existence of a TPM

In [11], Parno attempts to shed some light on the problems with bootstrapping trust in a TPM.[1] The contributions of [11], which we will briefly explain later on, can be summarized as follows.

1. Formally define the problem of bootstrapping trust in a platform containing a TPM.
2. Show how the formal model captures the cuckoo attack as well as hints on possible solutions.
3. Propose solutions to circumvent the cuckoo attack and also discuss the pros and cons of each proposed solution.
4. Since none of the proposed solutions are currently completely satisfactory, provide recommendations for future platforms intending to use TPMs.

Predicates

Predicate	Meaning
$\texttt{TrustedPerson}(p)$	User trusts person p.
$\texttt{PhysSecure}(c)$	Computer c is physically secure.
$\texttt{SaysSecure}(p, c)$	Person p says computer c is physically secure.
$\texttt{Trusted}_\texttt{C}(c)$	Computer c is trusted.
$\texttt{Trusted}_\texttt{T}(t)$	TPM t is trusted.
$\texttt{On}(t, c)$	TPM t resides on computer c.
$\texttt{CompSaysOn}(c, t)$	Computer c says TPM t is installed on computer c.

Axioms

1. $\forall p, c \;\; \texttt{TrustedPerson}(p) \wedge \texttt{SaysSecure}(p, c) \rightarrow \texttt{PhysSecure}(c)$
2. $\forall t, c \;\; \texttt{On}(t, c) \wedge \neg\, \texttt{PhysSecure}(c) \rightarrow \neg\, \texttt{Trusted}_\texttt{T}(t)$
3. $\forall t, c \;\; \texttt{On}(t, c) \wedge \texttt{PhysSecure}(c) \rightarrow \texttt{Trusted}_\texttt{T}(t)$
4. $\forall t, c \;\; \texttt{On}(t, c) \wedge \texttt{Trusted}_\texttt{T}(t) \rightarrow \texttt{Trusted}_\texttt{C}(c)$
5. $\forall t, c \;\; \texttt{On}(t, c) \wedge \neg\, \texttt{Trusted}_\texttt{T}(t) \rightarrow \neg\, \texttt{Trusted}_\texttt{C}(c)$
6. $\forall c, t \;\; \texttt{CompSaysOn}(c, t) \rightarrow \texttt{On}(t, c)$

Fig. 2. The predicates and the axioms of the formal model of [11].

Formal Model As for the first contribution, the author has formally defined the problem of bootstrapping trust using predicate logic. The predicates and the axioms of his formal model are shown in Fig. 2. Basically, the predicates describe the relevant properties of the system while the axioms encode facts about the platform.

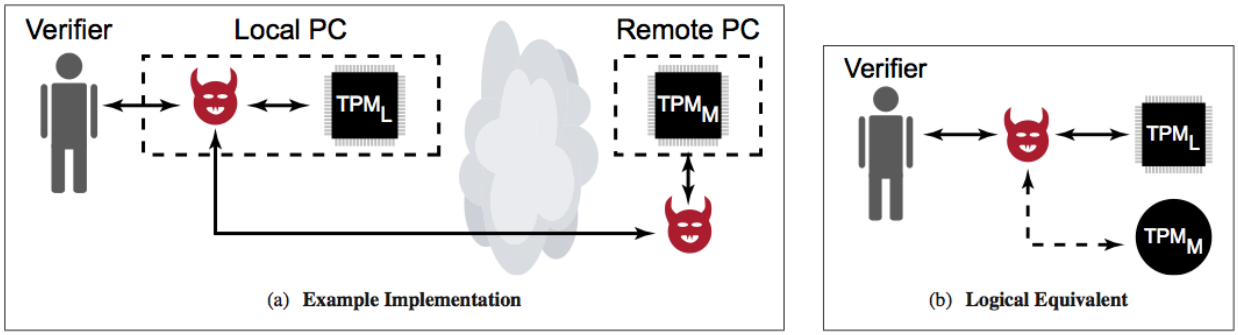

Fig. 3. The cuckoo attack of [11].

Cuckoo Attack A graphical representation of the cuckoo attack is shown in Fig. 3. Essentially, this is an impersonation attack where the adversary pretends to be the local TPM (TPM_L) installed at the user's computer.

According to the TCG specifications, the TPM's manufacturer provides the TPM with an Endorsement Certificate (EC) that certifies that the TPM is a genuine, hardware TPM and authenticates the TPM's public key. But, the EC only guarantees that the public key belongs to *some* TPM, thus allowing an adversary to mount a cuckoo attack.

Followed by the formal model, the author presents how the cuckoo attack is feasible within the given model. To do so, as shown in Fig. 4, he first initializes the system by setting the assumptions and then proves that these assumptions lead to a logical contradiction by using the axioms formulated in Fig. 2. This contradiction stems from the fact that the malware in Alice's computer C can

[1] An extended version of Parno's work appears at [12]

Assumptions

Assumption	Encoding
1. Alice trusts herself.	$\texttt{TrustedPerson}(\text{Alice})$
2. Alice says her computer C is physically secure.	$\texttt{SaysSecure}(\text{Alice}, C)$
3. The adversary controls machine M containing TPM_M.	$\texttt{On}(\text{TPM}_M, M)$
4. M is not physically secure.	$\neg\ \texttt{PhysSecure}(M)$
5. Malware on Alice's machine C causes it to say that TPM_M is installed on C.	$\texttt{CompSaysOn}(C, \text{TPM}_M)$

Proof

(1)	$\texttt{TrustedPerson}(\text{Alice})$	Assumption 1
(2)	$\texttt{SaysSecure}(\text{Alice}, C)$	Assumption 2
(3)	$\texttt{PhysSecure}(C)$	Axiom 1: (1), (2)
(4)	$\texttt{CompSaysOn}(C, \text{TPM}_M)$	Assumption 5
(5)	$\texttt{On}(\text{TPM}_M, C)$	Axiom 6: (4)
(6)	$\texttt{Trusted}_\texttt{T}(\text{TPM}_M)$	Axiom 3: (5), (3)
(7)	$\texttt{Trusted}_\texttt{C}(C)$	Axiom 4: (5), (6)
(8)	$\texttt{On}(\text{TPM}_M, M)$	Assumption 3
(9)	$\neg\ \texttt{PhysSecure}(M)$	Assumption 4
(10)	$\neg\ \texttt{Trusted}_\texttt{T}(\text{TPM}_M)$	Axiom 2: (8), (9)
(11)	$\neg\ \texttt{Trusted}_\texttt{C}(C)$	Axiom 5: (5), (10)
(12)	$\bot$	7, 11

Fig. 4. The assumptions and the proof related to the cuckoo attack of [11].

cause it to say that the adversarial TPM (TPM_M) is installed in C while it is actually installed in a remote computer M controlled by the adversary.

Proposed Solutions After the demonstration of the feasibility of the cuckoo attack, the author proposes several solutions to circumvent this attack. These solutions fall into three broad categories. We will now briefly explain these categories and the most attractive solution (if one exists) in each category.

Remove Network Access. The most obvious solution to avoid the cuckoo attack is to sever the local malware's access to the remote TPM by blocking network access. Unfortunately, removing network access is fundamentally not the right way because the formal model does not require network access in any of the assumptions and it also prevents a legitimate user from accessing the network. Also, if the attacker physically possesses the TPM, he can extract the TPM's private key and simulate the TPM locally within the malware installed in the user's computer.

Eliminate Malware. As the author shows, eliminating the malware equates to removing the 5^{th} assumption of the formal model. Unfortunately, this method is hard to achieve and possibly leads to an endless loop. The intuition behind this endless loop is that now we are trying to remove the malware in order to securely communicate with the TPM which is there to fight malware.

Establish a Secure Channel. Given the frustrations of the previous two categories of solutions, the most attractive solutions seem to be the ones that create a secure channel between the TPM and the user processes running on the computer, thus eliminating the malware's capability to mount the impersonation attack. The author presents four ways to create a hardware secure channel and another four ways to realize a cryptographic secure channel.

Of the hardware channels, the most attractive solution is designing a special-purpose hardware interface for the TPM because it removes almost every opportunity for user error, does not introduce extra secrets to the system, and does not require software updates. The only drawback of this approach is the cost and the industry collaboration required to introduce a new interface just for the TPM.

Of the cryptographic secure channels, the author argues for using a simplified version of the Seeing-is-Believing (SiB) approach originally suggested by McCune *et al.* in [13] and later extended to the kiosk computing in [14]. The idea of the original SiB approach is to have a 2-D barcode

containing the hash of the computer's identity (via a public key certificate) physically attached to the casing of the computer. A user can verify the identity of the computer by taking a picture of the 2-D barcode and consulting an online verification server. The only two drawbacks with this approach are 1) since the camera-equipped smartphones are usually complex devices, the users must be certain that their smartphones are not compromised, 2) an adversary can maliciously change the 2-D barcode to his choosing, especially in a kiosk setting. In order to remove the first drawback, the author suggests using a simple alphanumeric string to encode the hash of the computer's identity. In this way, the users can use a very simple device (*e.g.*, a pager) to run the verification.

Future Recommendations As you can see, none of the proposed solutions are completely suitable for the current situation with the TPMs. Therefore, the author suggests there is considerable room for additional solutions with stronger security guarantees by providing several avenues for further research.

4.2 Issues with authdata in TPM

The second and third security related issues we are surveying are connected to the "Authorization Data" (authdata) in TPMs. The first is basically an offline dictionary attack on weak authdata. The second is another form of impersonation attack caused by the shared authdata. As you will soon see, the TPM specifications surprisingly encourage sharing of authdata.

As we have explained earlier, the TPM stores cryptographic keys and other sensitive data in protected locations within the TPM itself. In order for a user process to access these protected items, the process has to prove to the TPM the knowledge of the corresponding authdata. Currently this is done by using authdata as an HMAC key in any communication between the user process and the TPM.

Offline Dictionary Attack on Weak authdata The authdata is chosen by the user process at time of storing the protected items within the TPM. The TPM specification allows upto 160 bits for authdata allowing it to be a high-entropy value. However, since the TPM specification does not impose any restriction on the entropy of the authdata, the user process can naively derive the actual authdata from a human-memorable low-entropy password. This is when it becomes feasible for an attacker to mount an offline dictionary attack on the authdata and extract the sensitive items stored by a user process in the TPM.

The offline dictionary attack goes as follows.

1. The attacker observes several valid authorization HMAC values exchanged between the user process and the TPM. Notice that almost all of the pre-images of the observed HMAC values are sent in the clear.
2. Later, once the attacker has enough HMAC values, he tries to guess the HMAC key, which is currently authdata, by using a traditional dictionary attack.

In [15], Chen and Ryan identify this attack and proposes three attractive approaches to change the existing TPM command set, thus preventing an adversary from mounting the attack. All of their solutions are based on the Simple Password Exponential Key Exchange by Jablon [16, 17]. The intuition behind the solutions given in [15], is twofold: 1) make the HMAC key high-entropy regardless of the entropy of authdata, 2) use public key instead of private key cryptography for the exchange of the HMAC key between the user process and the TPM.

Impersonation Attack on Shared authdata The cryptographic keys stored in a TPM are organized in a tree hierarchy with the Storage Root Key (SRK) of the TPM at the root. If a group of user processes are authorized to use a key, the authdata of that key may be shared among them. More importantly, the TPM specification requires the sharing the authdata for SRK in order for multiple user processes to generate and store their own keys in the TPM.

In [18], the authors show that the sharing of authdata as currently allowed in the TPM specification has a major undesirable consequence. Specifically, an adversarial process sharing the authdata with honest processes can mount a two-way impersonation attack. In other words, the adversarial process can impersonate an honest process to the TPM and obtain access to the protected items stored by that process, and also the adversarial process can impersonate the TPM to the honest processes and fake all the storage capabilities of the TPM.

The reason why an adversarial process can mount this impersonation attack is because of the way the authorization protocols in TPM generate the session keys. Currently, when a user process and the TPM establish a session, they publicly exchange two nonces n_e, n_o and generate the session key by computing the HMAC of the nonces under authdata as the HMAC key as shown below.

$$S = \mathsf{hmac}_{\mathsf{authdata}}(n_e, n_o)$$

As you can see, if authdata is shared with an adversarial process, that process can trivially compute the session key S since it can easily eavesdrop on the n_e, n_o values. After computing S, the adversarial process can decode all the encrypted communication carried out between the honest process and the TPM.

The way this attack is prevented in [18] is by using public key cryptography. In the proposed solution, the TPM now possess a pair of keys $\mathsf{PK}_{\mathsf{TPM}}, \mathsf{SK}_{\mathsf{TPM}}$. Of these two keys, it keeps the secret key $\mathsf{SK}_{\mathsf{TPM}}$ to itself and shares the public key $\mathsf{PK}_{\mathsf{TPM}}$ with the user processes. When a user process establishes a session, it generates a high-entropy secret key K and sends it to the TPM encrypted under the public key $\mathsf{PK}_{\mathsf{TPM}}$. Next, the TPM and the user process publicly exchange two nonces n_e, n_o and generate the session key as follows.

$$S = \mathsf{hmac}_K(n_e, n_o)$$

Notice now an adversarial process sharing the authdata and listening on the communication channel has no way of retrieving the secret key K. Thus, the security of the new authentication protocol follows.

The authors in [18] actually present their solution as a new authorization protocol named Session Key Authorization Protocol (SKAP). Next, using ProVerif [19], which is a widely used tool for assessing the security properties of protocols, the authors have shown that their new protocol prevents the impersonation attack.

4.3 Time-of-Check Time-of-Use Vulnerabilities (Software Attack)

The main focus of the trusted platform development has been on mechanisms that can be used to establish chain of trust, measure platform state, protect the secrets and data and attest remotely the state of the platform [20]. Developer have been able to exploit the security measures that TCG architecture offers to write software that undergo state transitions while being loaded into the memory.

One of the security measures of the TCG architecture is to ensure that the measured values contained in the Platform configuration register(PCR) match with the actual measured configuration of the software. TPM is used to attest the various states using a chain of trust between the states such that state S_i is attested prior to the attestation of state S_{i+1}. It is found that the attestation carried by TPM does not guarantee a trustworthy behavior of the software after it has been loaded into the memory. Since integrity measurements are taken prior to the loading of the program into memory, the assumption remains that the state of the program in memory remains unchanged after loading. The time gap between the time a program is loaded into memory and the time when it is actually used can be exploited to induce some runtime vulnerabilities. This constitutes the time-of-check time-of-use class of attacks.

Currently TPM implementation considers only static measurements of the state of data and software. This means that measurements are taken once during load time and changes in the measured values afterwards are not monitored. A potential attacker can modify the loaded binary in the RAM. This is possible given the kernel vulnerabilities, for example in Linux, that allowed attackers to bypass memory mapping and IPC restrictions [21]. Such modifications of a TPM trusted process are not reflected in the TPM software state measurements and thus constitutes a security breach. The modification of the program in the RAM leaves a security hole that can be exploited by an adversary to supply crafted input that likely to have influence on the behavior of the program.

Proudler in [22] suggested that next-generation trusted platform should be able to enforce policies. With the introduction of policies, the concern would extend from the implementing the trusted platform mechanisms to events that are classified and controlled mainly by policy. These policies should enforce the immutability of specific sections of the program's executable file as well as private data that the executable relies on. A feasible approach to enforce the policies would be to have TPM monitor those critical segments of the codes and write events that might end up changing these memory segments. A memory event trapping framework is proposed in [20] that implements trap handlers that actually work in conjunction with TPM to execute operations that help enforce security policies. The framework supports selective immutability and generates higher granularity memory access traps.

4.4 Reset Attack

During the boot process on a platform, a trusted BIOS takes measurement of the hardware and software installed on that machine and stores them in the PCR. Initially the PCR are filled with 20 NULL bytes and only after the BIOS sees them in this state that it can take the measurements and called the Extend() function to extend the measurements to the PCRs. The reset attack is one form of passive attack of the TPM. The main idea behind this form of attack, addressed in [23], is that the PCRs can be reset without restarting the whole platform on which TPM is installed, thereby compromising the chain of trust.

What makes this attack possible is the critical assumption that the PCR cannot be reset to the initial state without restarting the whole system. An attacker, monitoring the measurements sent by the BIOS to TPM and capable of zeroing out the PCRs, would be capable of changing the configuration of the platform and place it in a "trusted state". The speed of the buses connecting the memory and hard drives are pretty fast making the attack pretty difficult to execute. It is easier to carry out on a slower bus.

TPM is normally housed on the Low Pin Count (LPC) bus. The LPC bus supports the ground driven line reset. It has been shown that when the line is driven to ground, any device connected to this bus must reset, including the BIOS, key board and mouse controllers. This implies that it is possible to drive this line while the whole platform is still actively running. By so doing, an attacker can possibly fool TPM that the platform is restarting where as it is still running, since there are no form of authentication on the LPC bus. The TPM thinking that the platform is restarting would resume the authentication process and at this period the trusted boot process could be monitored and recorded by the adversary. The security of the trusted platform is violated as a result these actions. An attacker can thus replicate the trusted boot process and make changes to the PCRs with the help of a malicious device driver.

The reset attack can be mitigated by implementing some form of authentication on the LPC bus to prevent the possibility of an attacker to easily circumvent the chain of trust of the TPM authentication process. This might come in form a mechanism that disallows the reset of platform devices except with reserved privileges.

5 Summary & Conclusion

TPM's are a versatile tool that can be used in many industries. It can be used in the health profession guaranteeing only the doctors and nurses who are working with a patient will be able to pull up their records. It can be used to secure online banking and ecommerce. It can be used as a tool for Digital Rights Management. It can be used as a tool to prevent cheating in online games. In government agencies these tools can be used to change the existing clearance system and harden security down to a person to person or computer level.

TPM's however are not the solution for everything. There are inherent flaws and privacy concerns. The Internet was founded on the principles of openness and anonymity, TPM's by their very nature are tools that uniquely identify a system. They are preinstalled with tools that defeat the anonymous nature of the web and allow a system to be uniquely identified. Because of this, currently users must enable TPM before they can utilize the benefits, but to eventually move to a system where TPM's are enabled by default means to lose anonymity. It puts a lot of power into the hand of designers and developers who can refuse to service users who do not use the TPM.

In addition as we've seen in this paper, TPM's by design are vulnerable to a variety of attacks like the cuckoo attack, dictionary attack, timing attack, and reset attack. These attacks prey on the core architecture of the TPM and not only bypass the security measures the TPM provides, in some cases they provide means for an attacker to pretend to be a trusted user. Many challenges still remain for the future of TPM.

Trusted Computing is a new technology that provides a unique approach to enhancing the security of a system. TPM's look to both harden security on a local and on a network level. The technology is still in its infancy and has the benefit of being cheap, reliable, and simple to deploy. When used properly it is entirely transparent to the end user. While TPM's are powerful, it cannot be seen as a solution for every security issue however and should be one of many tools used to secure your machine.

References

1. Anderson, R.: Cryptography and competition policy: issues with "trusted computing". In: Proceedings of the twenty-second annual symposium on Principles of distributed computing, ACM Press (2003)

2. TCG: Trusted platform module summary. Technical report, Trusted Computing Group (2008)
3. TCG: Tcg specification architecture overview. Technical report, Trusted Computing Group (2007)
4. TCG: How to use tpm. Technical report, Trusted Computing Group (2009)
5. Linder, R.: Adobe to revoke signing certificate after compromise of internal server. infosecurity-magazine (2012)
6. Wheeler, M.: Wave systems corp. : Wave joins industry thought leaders on mobile security panel at interop ny. Wave Systems (2012)
7. Smith, D.: Windows phone 8 has baked in cybersecurity goodness. NetworkWorld (2012)
8. Richard, D.: Hacker extracts crypto key from tpm chip. The H Security (2010)
9. Tomlinson, A.: Introduction to the tpm. In: Smart Cards, Tokens, Security and Applications. Springer (2008) 155–172
10. TCG: TCG Specification Architecture Overview Revision 1.4. Trusted Computing Group, Portland, OR. (August 2007)
11. Parno, B.: Bootstrapping trust in a "trusted" platform. In: Proceedings of the 3rd conference on Hot topics in security. HOTSEC '08 (2008) 9:1–9:6
12. Parno, B., McCune, J., Perrig, A.: Challenges in bootstrapping trust in secure hardware. In: Bootstrapping Trust in Modern Computers. Volume 10. Springer, New York (2011) 41–50
13. McCune, J.M., Perrig, A., Reiter, M.K.: Seeing is believing: Using camera phones for human-verifiable authentication. Int. J. Secur. Netw. **4**(1/2) (February 2009) 43–56
14. Garriss, S., Cáceres, R., Berger, S., Sailer, R., van Doorn, L., Zhang, X.: Trustworthy and personalized computing on public kiosks. In: Proceedings of the 6th international conference on Mobile systems, applications, and services. MobiSys '08 (2008) 199–210
15. Chen, L., Ryan, M.: Offline dictionary attack on TCG TPM weak authorisation data, and solution. In: Future of Trust in Computing. Vieweg+Teubner (2009) 193–196
16. Jablon, D.P.: Strong password-only authenticated key exchange. SIGCOMM Comput. Commun. Rev. **26**(5) (October 1996) 5–26
17. Jablon, D.P.: Extended password key exchange protocols immune to dictionary attacks. In: Proceedings of the 6th Workshop on Enabling Technologies on Infrastructure for Collaborative Enterprises. WET-ICE '97 (1997) 248–255
18. Chen, L., Ryan, M.: Attack, solution and verification for shared authorisation data in TCG TPM. In: Proceedings of the 6th international conference on Formal Aspects in Security and Trust. FAST '09 (2010) 201–216
19. Blanchet, B.: An efficient cryptographic protocol verifier based on prolog rules. In: Proceedings of the 14th IEEE workshop on Computer Security Foundations. CSFW '01 (2001)
20. Bratus, S., DíCunha, N., Sparks, E., Smith, S.: Toctou, traps, and trusted computing. In: Trusted Computing - Challenges and Applications. Springer Berlin / Heidelberg (2008) 14–32
21. Arce, I.: Kernel craze. IEEE Security and Privacy (2004) 78–81
22. Proudler, G.: Concepts of trusted computing. In Mitchell, C., ed.: Trusted Computing. (2005) 11–27
23. Kursawe, K., Schellekens, D., Preneel, B.: Analyzing trusted platform communication. In: In: ECRYPT Workshop, CRASH ñ CRyptographic Advances in Secure Hardware. (2005) 8