Trust of Hardware
Zhang Fan1, Wu Guoqing1, Tao Jun2, Yuan Mengting1, Liu Xiaoli1
1. Computer School of Wuhan University, Wuhan, China
2. Network Center, China Telecom, Hefei, China

Zhangfan-cixjy@163.com

Abstract
Initially, we point out that hardware trust is pretty important to a trusted computing platform based on prior work. Then we propose the notion of measuring hardware trust. We also analyze key factors needed to be considered when designing a hardware-measurement architecture, and give an outline of our approach. Next, we show the measurement result and further discuss some other applications of hardware measurement. Finally, we conclude that sufficient attentions should be paid to hardware trust. Measuring and managing hardware trust are strongly recommended.

1. Introduction
Nowadays, various hardware devices, such as USB devices, wireless devices, and Bluetooth devices and so on, have penetrated into our daily life, making our life more and more flexible and comfortable.

However, with the wide use of these devices, more and more hardware security problems emerge at the same time. In [1], the author shows examples from IEEE 1394 devices, USB devices, and wireless devices respectively to show possible attacks through peripheral devices. In [2], the authors present that we should be on the alert for the firmware attacks on hardware. Firmware attack, as a dangerous attack which can lead to tremendous damages, is no longer an assumption. In [3], the authors present smart devices. These devices are embedded with a security chip to enhance their security functions and make them smarter. These smart devices can not only protect them from being attacked, but also notify nearby devices to increase their security levels to defend possible subsequent attacks once an attack is detected. Actually, embedding a chip into a harddisk or other devices is possible, [4] embeds a search engineering into a harddisk to improve the search efficiency greatly. Reference [5] discusses the trusted hardware. The author argues that hardware will suffer from more and more attacks in the future, and formal development approaches should be used to construct a trustworthy hardware. In [6], the authors read data from a hardware device without being noticed in an unusual way that can break the common software-based access control. Their experiments reveal that vendors of products, such as smartcards and secure microcontrollers, should take storage security into consideration.

All of the work above discloses the fact that one the one hand, hardware trust is pretty important to a trusted computing environment; on the other hand, traditional hardware devices need necessary security improvements so that they can be used in a highly trusted computing environment.

In 2003, the Trusted Computing Group (TCG) [7] was established. Robert Thibadeau, as a member of Technical Committee and Board of Directors of the Trusted Computing Group, proposes a framework to apply “trusted computing for disk drives and other peripherals [8]”. In that paper, Robert gives the general risk model of peripheral devices, and analyzes how we can apply access control to secure them, so that these trust enhanced devices can be the independent roots of trust.

Following prior work of hardware trust, we propose the trust measurement of hardware in this paper. This paper is organized as follows. Section 2 describes the trust measurement of hardware. Section 3 outlines our approach, gives the measurement result and also discusses some other applications of hardware measurement. Section 4 draws the conclusion.

2. Trust measurement of hardware
2.1 Factors to be considered
To implement a hardware measurement architecture, some factors should be considered carefully.

The key factors are “who, what, when, how, and other”, which are shown in figure 2.1. Once these problems are solved, engineers can design a more suitable framework according to their requirements.
In the following, we will illustrate how we design our framework by considering these five factors respectively.

(1) Who. First of all, we should choose the root entity that measures other hardware devices. The entity itself must be the root of platform trust and is absolutely trusted. As we know, in the TCG specification, the Trust Platform Module (TPM) is the root of trust, so TPM is naturally this kind of entity. In the future, if hardware devices are secured as [8] by manufacturers, they can also be the independent root of trust, thus they can also be that kind of entity. Currently, we choose TPM as the root to measure other devices. More details about TPM can be found in the TPM specification [9].

(2) What. Then, we need to decide what kind of information to collect. Information is collected to identify whether the device is tampered with or not, so the best position to collect this information is the possible vulnerable points of hardware. In [8], the author gives the general risk model of peripheral devices.

(3) When. Next, we ought to determine when to perform the measure. Typically, we can measure hardware trust in two phases. One is when a platform powers up. The other is after OS is running. In the former phase, some “fixed” devices, such as Motherboard, CPU, and Memory and so on can be measured. In the latter phase, some “non-fixed” devices, such as hot-swappable devices, wireless devices and Bluetooth devices should be measured. Implementing measurements in both phases is required, because we can see in section 3 that measurements in either phase have their specific functions and characteristics, and they cannot take the place of each other.

(4) How. How can we measure peripheral devices, after we have considered the factors discussed above? For example, the measure process is automatic, semi-automatic, or manual? The measure process is a chaining way or other forms? In our approach, we design a chaining way similar to trust chain defined by TCG. TCG defines trust chain as follows:

$$\text{CRTM} \rightarrow \text{BIOS} \rightarrow \text{OSLoader} \rightarrow \text{OS} \rightarrow \text{Applications}$$

When a platform powers up, CRTM (Core Root of Trust Measurement), which is a block of code in BIOS and is absolutely trusted, measures the trust of BIOS first. If BIOS is trusted, the platform trust is expanded from CRTM to CRTM + BIOS. BOIS then measures the trust of OSLoader. Similarly, if OSLoader is trusted, the platform trust is expanded to CRTM + BIOS + OSLoader, and OS is subsequently measured by OSLoader. The process continues on until at last OS is running and infinitely measures the trust of applications. The process seems like a chain, so it is called trust chain.

Likewise, we also use a chaining way to measure hardware trust.

$$\text{TPM} \rightarrow \text{Motherboard} \rightarrow \text{BIOS} \rightarrow \text{CPU} \rightarrow \text{Memory} \rightarrow \text{DisplayCard} \rightarrow \text{SoundCard} \rightarrow \ldots \rightarrow \text{Keyboard} \rightarrow \text{Mouse} \rightarrow \ldots$$

When a platform powers up, TPM measures the trust of Motherboard first. If Motherboard is trusted, then the embedded chips such as BIOS, south-bridge chip, north-bridge chip are measured. If BIOS is trusted, the hardware trust is expanded from TPM to TPM + Motherboard + BIOS, and CPU is subsequently measured. The process continues until all of the related peripheral devices are measured.

(5) Other factors. Beside those five factors, some others factors should also be taken into account according to actual situations. For example, in the real implementation, a balance among efficiency, security and cost should be achieved. A high level security is usually accompanied by a high cost. So, choosing
what kind of security and which level of cost should be considered integratedly.

Thus far, key factors when we design a framework for hardware measurement are discussed. We also show how we analyze these factors so as to construct our approach. In the next section, section 2.2, an outline of our approach is given.

2.2 The approach

An outline of our approach is as follows.

Initially, we assume that there is a self-measurement chip embedded in a device. The self-measurement receives the self-measurement command \texttt{Self\_MeaCmd} from TPM, and then collects information from the possible attack points shown in figure 2.2. The reason why we require a self-measurement chip is that it can be more flexible to collect information by using such a chip. Besides the flexibility, related embedded chip provided by vendors for security concerns can also enhance the security of a device from different aspects, and forms the bases of hardware measurement.

Assumption 1: Suppose each platform hardware has a self-measurement chip. When the hardware receives the self-measurement command \texttt{Self\_MeaCmd} coming from TPM, it starts the self-measurement and collects the corresponding information: \texttt{Self\_MeaInfo} = \{I\textsubscript{1}, I\textsubscript{2}, I\textsubscript{3}, ......, I\textsubscript{n}\}.

The main steps of hardware measurement are as follows.

Hardware measurement:

Step 1: TPM sends the self-measurement command \texttt{Self\_MeaCmd} to a device to be measured.

Step 2: The device receives the command and starts to collect information, and then returns the collected information \texttt{Self\_MeaInfo} = \{I\textsubscript{1}, I\textsubscript{2}, I\textsubscript{3}, ......, I\textsubscript{n}\} to TPM.

Step 3: TPM analyzes the received information to judge whether the device is trusted or not.

Step one is easy to understand. So we give an explanation of Step Two and Step Three here.

Firstly, Let us take a look at Step Two. In Step Two, the related information has been collected by the embedded chip. The next operation is to send the collected information to TPM truthfully. To ensure that TPM receives the exact information not being attacked, three attacks should be considered:

(a) Replay attack: an attacker may intercept the measurement information, and then resends the measurement result before the hardware is compromised to TPM.

(b) Tampering: an attacker may maliciously tamper the measurement information, and then resends the tampered information to TPM.

(c) Masquerading: an attacker may send the measurement information of another device which is not compromised to TPM.

Thus, the main purpose of Step Two is to guarantee that TPM receives the unattacked measurement information from a device. In [8], the author lists five access control approaches. Based on [8], we have designed a Public Key approach to provide the highest-grade secure sending, if there is a Public Key Engine embedded in the device. In the other paper, there is a detailed introduction. In a word, adopting which security level and what kind of secure sending is based on two factors: on the one hand, the security enhancement of that device provided by vendors; on the other hand, the comprehensive consideration of security, efficiency and cost.

After receiving the exact measurement information, in Step Three, TPM will analyze whether the device measured is trusted or not.

Definition 1: Suppose there is a configurable Standard Platform Device Table (SPDT). This table is configured by administrators and stored in TPM in advance. The SPDT stores the standard hardware configuration information of a platform. By comparing the measured information of a device against the SPDT, TPM can find whether a device measured is trusted or not. Table 2.1 gives an example of SPDT.

<table>
<thead>
<tr>
<th>device info</th>
<th>expected info</th>
<th>extended</th>
</tr>
</thead>
<tbody>
<tr>
<td>EliteGroup Motherboard</td>
<td>/</td>
<td>ID=02/19/2002-SiS-745-6A61UE19C-00</td>
</tr>
<tr>
<td>Jet BIOS (Test)</td>
<td>ECS</td>
<td>Revision 0000.0015</td>
</tr>
<tr>
<td>AMD CPU ECS</td>
<td></td>
<td>Athlon(tm) processor, CPUID=000006A0h</td>
</tr>
<tr>
<td>Seagate Harddisk ATA</td>
<td></td>
<td>Capacity=40G, C=77545, H=16, S=63</td>
</tr>
</tbody>
</table>

Suppose set \texttt{DEVICE} includes all of the platform hardware, and a concrete device is \texttt{device \in DEVICE}. The SPDT is composed of many entries, and each entry is the following:

\texttt{Conf(device) = [device info, expected MeaInfo, extended]},

where \texttt{device info} represents the name of the device, the interface (such as IDE, SCSI and so on) to which the device connects; \texttt{expected MeaInfo} indicates the expected measurement result of the device, and \texttt{extended} contains the extended expected information, whether to use and how to use the \texttt{extended} part is determined by administrators.

The first entry of Table 2.1 is related to the Motherboard. From the \texttt{device info} part, we know that this is a Motherboard made by the Elite Group System Company. And the interface is "/", indicating that it is
not connected to any interface. The expected\_mea\_nfo gives the key expected measurement information. For example, the ID, the serial number, the bus frequency and so on. For the space limitation, we do not give the full expected measurement information here. There is no other extended expected information, so the extended part is not used. The second entry corresponds to our modified BIOS. The “ECS” indicates that it is connected to an EliteGroup Motherboard. The version of the BIOS is “Revision 0000.0015”, and its expected hash is not shown here.

Suppose set DEVICE indicates all of the platform devices, and the detected devices are included in the set Detected\_DEVICE \subseteq DEVICE . By comparing measured information against the SPDT, three kinds of attacks can be detected.

(1) A new hardware which does not belong to the standard configuration of the platform is connected to the platform (Attack\_add\_hardware ) by and attacker. Formally,

\[ \forall \text{device} \in \text{Detected\_DEVICE} \land \exists \text{Conf}(\text{device}) \notin \text{SPDT}. \]

The exact reason is that: for the device\_info of the detected device: \( \forall \text{Conf}[\text{device}] \in \text{SPDT} \Rightarrow \text{device}\_info \notin \text{Conf}. \)

(2) A critical hardware of the platform is disconnected by an attacker (Attack\_remove\_hardware ). Formally,

\[ \exists \text{device} \in \text{Detected\_DEVICE} \land \exists \text{Conf}(\text{device}) \notin \text{SPDT}. \]

(3) A critical hardware of the platform is maliciously attacked (Attack\_tam\_hardware ). For example, the firmware of the harddisk is tampered with. Formally,

\[ \exists \text{device} \in \text{Detected\_DEVICE} \land \exists \text{Conf}(\text{device}) \notin \text{PDT}. \]

The exact reason is that: for device\_info and expected\_mea\_nfo of the compromised device : \( \exists \text{Conf} \in \text{SPDT} \land \text{device}\_info \in \text{Conf} \Rightarrow \text{expected\_mea\_nfo} \notin \text{Conf}. \)

By now, the trust of hardware can be measured.

In this section, we propose the notion of measuring hardware trust, and discuss two problems based on this notion. (1) Some factors needed to be considered when we design an approach to measure the hardware trust. (2) With secured hardware, different measurement approaches can be designed.

In section 3, we show results of trust measurement of hardware, and further discuss its wide use.

3. Examples

3.1 Measurement results

Our approach is based on the trusted computer designed by us [10]. The difference between our trusted computer and common PC is that we tightly embed an Embedded Security Module (ESM) into the motherboard. The ESM is similar to a TPM, more details can be found in [10] [11].

BIOS: Trusted
Vendor: Jetway Information Security Corp.
Version: Revision 0000.0015
SHA1: CK

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HDD1</td>
<td>Warning</td>
</tr>
<tr>
<td>ID:</td>
<td>ST340015A</td>
</tr>
<tr>
<td>Serial Number:</td>
<td>5LAD1J8</td>
</tr>
<tr>
<td>CHS:</td>
<td>77545, 13, 63</td>
</tr>
</tbody>
</table>

!!!Warning: MBR is corrupted!!!
ErrorCode: 03
Description: this disk may connect to other platforms.

USB: Warning
Vendor: USB 2.0 (HS) Flash Disk
Serial Number: 00000000003D5

!!!Warning: a USB disk was connected at 19:23!!!

Figure 3.1 Measurement results

Though the approach in section 2.2 cannot be completely and directly implemented (necessary security enhancements are needed), we still implement it partly based on our trusted computer. Figure 3.1 gives a part of the report about hardware measurements.

From figure 3.1, we can find the BIOS is trusted. Meanwhile, there are also warning messages, indicating possible attacks.

(1)The first warning message indicates that the Harddisk1 may be connected other platforms. As we know, an attacker who can physically access the platform may disconnect a harddisk from a trusted computing platform and then connect it to another common platform, so that he can bypass the defense of the trusted computing platform to steal data. Because data stored in Harddisk1 is not encrypted, this attack is possible and dangerous. One way to defend this attack is to initiate encrypt data stored in harddisk; the other way is to passively detect this attack as ours. In our platform, we modify MBR such that MBR must interact with the ESM so as to boot up normally. Otherwise if the harddisk is connected to another platform, the MBR will be corrupted automatically if the unique ESM cannot be found.

(2) The second warning message indicates that a USB disk has ever been connected to this platform. We prescribe that USB devices cannot be used in this
platform for security concerns. The Operation System (OS) is enhanced to monitor all of the USB device connections. Once a USB device is connected, the behavior is audited. Likewise, we can also set to monitor any wireless device connections. Then all of the unauthorized wireless connections will be recorded. OS can further automatically react to wireless connections according to policies, once it detects a connection requirement. Thus, the wireless attacks which is hardly found by visual inspection [1], can be easily detected, because the OS can automatically monitor and react to all of the wireless operations without any intervention of humans.

The first possible attack in figure 3.1 was found in the stage of platform power-up, and the second possible attack was found after OS is running. Obviously, they can not take the place of each other. Therefore, it is strongly recommended that we should implement both kinds of hardware measurement in different stages.

Furthermore, in section 3.2, more applications of hardware measurement are listed.

3.2 Other applications

In this section, some other applications of hardware measurement are shown.

(1) Detecting maliciously increasing/decreasing a device.

If a device that does not belong to a platform is connected, this attack can be immediately found when TPM compares the device information against the SPDT. Similarly, if a critical device is disconnected from a platform, it can also be detected based on this kind of comparison.

(2) Detecting tampering with a device.

In most situations, a device is tampered with by attacking its firmware. To defend this kind of attack, we need to save the firmware hash, for example, SHA1, of the device first. Then, after comparing the saved hash against the measured one, we can detect the tampering attack. The hash of the firmware can be stored in the expected_meaning part of SPDT.

However, the firmware of a device is usually under the protection of the copyright of vendors. Reading (modifying) firmware is strictly prohibited. So, in order to save the firmware SHA1 of a device, necessary hardware enhancement or modification of that device is needed in the future.

(3) Binding a device.

In some companies, for security concerns, managers prescribe that confidential data can only be accessed by specific devices and using these devices outside the company is absolutely prohibited. By applying the hardware measurement, this problem can be solved flexibly. For example, on the one hand, TPM and OS can check the identity of a device when it requires accessing the confidential data. If identity checking passes, it can access the data. Otherwise, the request is denied and audited. On the other hand, data flowing from a platform to the device should be automatically encrypted by TPM and decrypted for the opposite direction. Thus, the confidentiality of the data is guaranteed, because an attack can get nothing but the encrypted data if the specific device is used outside of the company.

(4) Choosing a trusted path.

In [2], the authors show possible hardware measurements result as Figure 3.2. Figure 3.2 is copied from [2] and as follows.

![Figure 3.2 Hardware Measurement](image)

After hardware measurement, we can choose the trusted path. For example, if there are two Cameras in a platform: an USB one and a SCSI one. As we can see, the USB camera is not trusted (marked with a cross). Suppose there is another trusted SCSI camera. Thus, when a user requests the camera service, the platform will choose the trusted SCSI camera automatically, inform the user untrusted devices and interfaces in the path from CRTM to the USB camera. Moreover, if the user insists on using the untrusted camera, the platform can deny the request, or allow it but audit the compulsive choosing behavior.

(5) Supplement to existing security applications.

The hardware measurement can be a supplement to existing security applications. For example, some researchers try to use U-key to reinforce a system [12] [13]. This method is questionable if there is not a related supplement. For example, if an attacker duplicates a legal U-key, or cracks it by tampering data stored in it, the defense provided by that U-key can be bypassed. However, if we implement the hardware measurement, these problems can be solved. For example, as we know, the identities of two U-keys are different. So, duplications can be found by identifying the identities of U-Keys, and the identifying process can be performed by hardware measurement.
Meanwhile, the crack can also be detected by comparing the hash of data stored in a U-Key against the expected one saved in advance.

(6) Confidentiality and integrity improvement.

The hardware measurement can also help to improve system confidentiality and integrity. In figure 3.2, for confidentiality consideration, any data, as long as they flow through an untrusted device or interface, should be encrypted. For example, if there is an IDE harddisk Disk3 in figure 3.4, because the IDE interface is not trusted, any data flow from Disk3 to a PCI device should be encrypted. For integrity consideration, for example, since the SCSI Disk2 is not trusted, any data coming from Disk2 should be assigned with a low integrity level.

(7) Some other security problems.

All of the security problems in [1] [2] can be solved by hardware measurement. Furthermore, there are some other applications, and we do not give a detailed discussion here because of space limitations.

4. Conclusion

In this paper, we point out that hardware trust is pretty important to a trusted computing platform. We further propose the notion of measuring hardware trust. Then, we give our measurement result and discuss some other applications of measuring hardware trust. Finally, we conclude that on the one hand, necessary security enhancement must be applied in the common devices so that they can be used in a highly trusted computing environment; on the other hand, engineers should pay sufficient attentions to the trust of hardware.

In short, trust needs to be built based on both hardware and software. A true trust cannot be achieved if either of aspect is absent. Furthermore, measuring and managing hardware trust of a platform are strongly recommended.

5. References