BIOS Security Analysis and a Kind of Trusted BIOS

ZhenLiu Zhou and RongSheng Xu

Network Security Laboratory of High Energy Physics, CAS, China
{zhouzl,xurs}@ihep.ac.cn

Abstract. The BIOS's security threats to computer system are analyzed and security requirements for firmware BIOS are summarized in this paper. Through discussion about TCG's trust transitivity, a new approach about CRTM implementation based on BIOS is developed. In this paper, we also put forward a new trusted BIOS architecture-UTBIOS which is built on Intel Framework for EFI/UEFI. The trustworthiness of UTBIOS is based on trusted hardware TPM. In UTBIOS, trust encapsulation and trust measurement are used to construct pre-OS trust chain. Performance of trust measurement is also analyzed in the end.

Keywords: Trusted Computing, Trust Measurement, BIOS, UEFI, TPM.

1 Introduction

The traditional information security mechanisms which are built on the level of operating system can not meet the requirements of the developing information security. The security of computer system requires protection to extend to the firmware level even the hardware level. The firmware BIOS is the software that the computer processor carried out at the earliest stage, so its security will affect directly the security of the whole computer system. The traditional design of BIOS does not consider the security problem, it exists many hidden risks. In 1997, William A. Arbaugh put forward a kind of computer security bootstrap architecture AEGIS [1]. AEGIS is based on the traditional BIOS of IBM PC, it ensures the integrity of the code of the firmware BIOS and improves the security of code at BIOS bootstrap process by using authentication. But AEGIS lacks hardware protection and trusted root of hardware. As a firmware, AEGIS also can't provide extent protection for operating system. Dexter Kozen put forward a language-based approach which checks the security of control flow safety, memory safety and stack safety during compiling process to improve the code security, and using this way to inspect malicious code for open firmware [2] [3]. But the language-based method is so complex that it works poor in practice.

Without a trusted bootstrap process or trusted BIOS the operating system and application cannot be trusted since it is invoked by an un-trusted process. This paper researches how to build a new trusted firmware BIOS.

1.1 UEFI

The Extensible Firmware Interface [9], or EFI, is the layer between the Operating System and platform firmware. EFI is the new model for the interface between

S. Qing, H. Imai, and G. Wang (Eds.): ICICS 2007, LNCS 4861, pp. 427–437, 2007.

operating systems and the platform firmware running them. EFI is the data tables (containing platform specific information), boot and runtime service calls available to the operating system and its loader. The end result of EFI is a standards compliant environment for running pre-boot applications and for booting the operating system.

The EFI specification was primarily intended for the next generation of IA architecture-based computers, and is an outgrowth of the "Intel® Boot Initiative" (IBI) program that began in 1998. In 2005 the Unified EFI Forum was formed. Using the EFI 1.10 specification as the starting point, this industry group is now responsible for developing, managing and promoting the ongoing evolution of the UEFI specification [6][10].

1.2 The Intel Platform Innovation Framework for EFI

The Intel Platform Innovation Framework for EFI (referred to as "the Framework for EFI/UEFI") is a product-strength implementation of EFI and UEFI. The Framework is a set of robust architectural interfaces, implemented in C that has been designed to enable the BIOS industry to accelerate the evolution of innovative, differentiated, platform designs. The Framework is Intel's recommended implementation of the EFI Specification for platforms based on all members of the Intel® Architecture (IA) family [7].

Unlike the EFI Specification, which focuses only on programmatic interfaces for the interactions between the operating system and system firmware, the Framework is an all-new firmware implementation that has been designed to perform the full range of operations that are required to initialize the platform from power on through transfer of control to the operating system.

Figure 1 shows the phases that a platform with Framework-based firmware goes through on a cold boot.

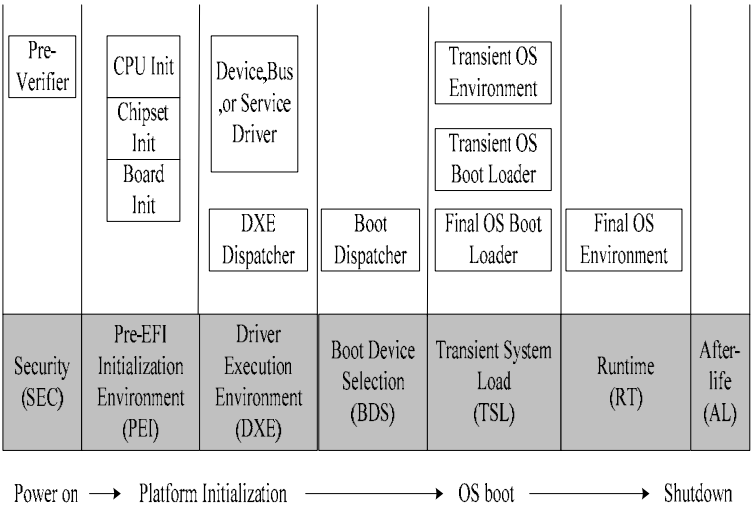

Fig. 1. Framework Firmware Phases

The SEC phase is defined as CRTM in Intel Platform Innovation Framework for EFI [7], but Intel gave an empty implementation of CRTM in Intel's EFI BIOS product. Furthermore, the smallest set of functions for CRTM is not discussed yet in Intel Platform Innovation Framework [7], and it is not sufficient to exclude PEI phase out of the CRTM. Section 4 of this paper discusses about this.

1.3 Trusted Computing

Trusted computing environment requires every component from hardware level to software level in whole system is trusted and excludes any component from system that can not be proved as trusted, such as malicious-codes, viruses, and Trojans etc. This computing and network environment is named as trusted computing ecological environment. For building this trusted computing environment, we should enhance the lower layers security of computer terminal such as hardware and firmware. In 2003, IBM, Intel, and Microsoft established international trusted computing organization TCG (Trusted Computing Group). The TCG is a not-for-profit organization formed to develop, define, and promote open standards for hardware-enabled trusted computing and security technologies, including hardware building blocks and software interfaces, across multiple platforms, peripherals, and devices.

2 Transitive Trust

The trustworthiness of trusted computing of TCG is based on the trusted hardware. The chain of trust is constructed through transitive trust between hardware and software [8] [13]. Any entities must be authenticated and validated about its integrity before be loaded and run. The process of trust transiting that is defined by TCG is illustrated in Figure 2.

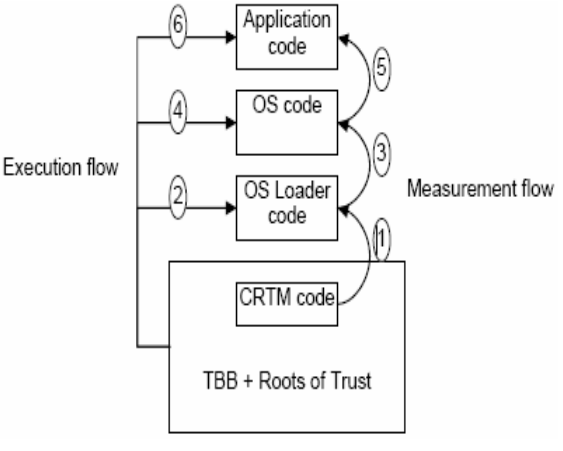

Fig. 2. Transitive Trust

TBB（Trusted Building Blocks）is the basic component of trusted computing platform. The core trusted root of TBB is the Trusted Platform Module (TPM). TPM [12] is a microcontroller chip affixed to the motherboard of computer that stores keys, digital certificates and trust measuring event logs. TPM also has computing capabilities such as digital signature, key exchange, encryption and decryption etc. Although these functions can be realized using pure software, TPM usually adopts independence hardware chips for satisfying sealed store requirement.

Core Root of Trust for Measurement (CRTM) is a trusted component which include the first instruction CPU picks up when power on. CRTM should be trusted non-conditionally.

Ideally, the CRTM is contained in TPM [13], which is illustrated as Figure 3.A. This demands CPU reset vector points to TPM, i.e., the first instruction which CPU picks and executes should be located in TPM after power on. This requires the existing computer architecture to be changed. Implementation decision of this paper is putting CRTM into BIOS firmware illustrated as Figure 3.B. Following the existing computer architecture, the first instruction picked and executed by CPU after Power on is located at address 0xFFFFFFF0 (32bit CPU) in BIOS firmware, so this implementation is more realizable. For this implementation, additional software and hardware protection should be taken to enhance the reliability and security of CRTM.

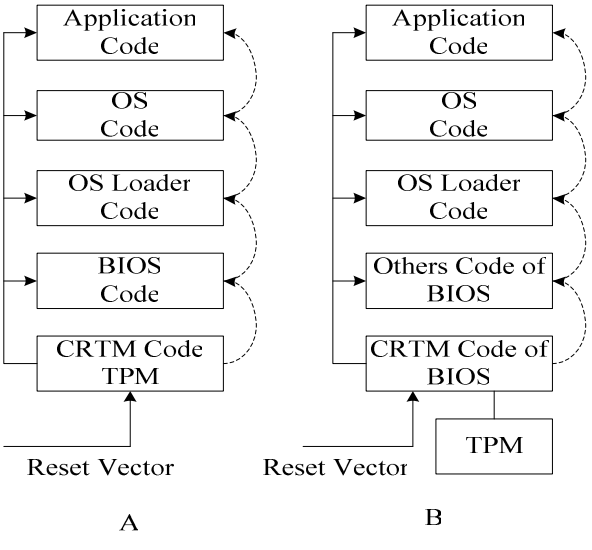

Fig. 3. Different Implementation of CRTM

3 Security Threats Against BIOS

Security threats against BIOS should be analyzed firstly to list security requirements of trusted BIOS.

For requirements of patching and updating BIOS product online, more and more mainboard manufacturers use FLASH ROM to store BIOS firmware. Using of flash ROM makes BIOS code can be read and flashed just by means of pure software without any assistant hardware device under operating system environment. So the third spiteful party can implant malicious codes such as virus and Trojan into BIOS [4]. And the increasing capacity of Flash chip also provides storage space for malicious codes.

Framework for EFI/UEFI even admits user load and execute EFI drivers or EFI applications under EFI Shell [7]. These EFI drivers or applications may come from other devices, such as USB disk or hard disk, than the Flash chip that contains the BIOS code. This causes increasing chances for malicious code being loaded and executed by BIOS.

Intruders can also juggle some bytes to destroy the integrity of BIOS code. This is a kind of denial-of-service attack that will make the computer system can not boot normally. The representative of this kind of attack is CIH virus and its variation.

4 Trusted BIOS

4.1 Security Requirements of Trusted BIOS

Trusted computing requires every step of execution can transit trust and build chain of trust based on trusted hardware from system lower-level to upper-level. These requirements emphasize the integrity protection of entity which is similar with the integrity model of Clark-Wilson [5]. Trust transitivity is implemented by trust measurement. Trust measurement is the process to verify integrity and authenticity of entities.

Trust measurement can prevent execution of illegal and malicious Option ROMs, EFI drivers and EFI applications which come from external to ensure bootstrap process only executes the code from trusted BIOS manufactures, devices driver manufactures, or trusted users. Although malicious code can be embedded into BIOS or BIOS code may be juggled, the trust measurement can prevent the execution of these illegal or malicious codes.

Trusted BIOS need to measure the following three kinds of code. One is BIOS own code of trusted BIOS except CRTM code. The second include option ROM code, EFI driver and EFI application loaded by trusted BIOS from external. The third is OS Loader Code.

When the integrity of BIOS code or data is juggled by unpredictable failure or attack, the trusted BIOS system must support the safe and reliable failure self-recovery mechanism to deal with denial-of-service attack. In order to ensure the trusted self-recovery mechanism may not be destroyed, the BIOS codes that used to implement the failure self-recovery mechanism must be protected by hardware. During the process of recovery, the recovery contents integrity must be measured too. This process is called trusted self-recovery.

4.2 Security Requirements of CRTM

As starting point of the chain of trust measurement, CRTM should be trusted unconditionally. To guarantee the source of trust transiting can be trusted, CRTM

should content with four preconditions: 1) CRTM should be protected physically from writing or flashing by means of pure software; 2) CRTM should have capability of trust measuring for the next step executive code; 3) CRTM should have trusted self-recovery mechanism when others component of trusted BIOS is juggled; 4) The code of CRTM should be the minimum set which can fulfill the functions of precondition 2 and 3, including other essential initial codes for CPU, chipset and platform.

5 UTBIOS Architecture and Boot Process

UTBIOS is a trusted bios building on Intel Platform Innovation Framework for EFI/UEFI. The SEC phase is defined as CRTM by Intel in [7], but Intel gave an empty implementation of CRTM in Intel's EFI BIOS product. Furthermore, it is not sufficient to fulfill CRTM functions if PEI phase defined by Intel is excluded out of the CRTM.

Requirements of CRTM are discussed in section 4.2. Based on these requirements, this paper reconstructs the SEC and PEI phase of Intel Framework for EFI, and adds other essential functions in. In UTBIOS, Intel's phase SEC and PEI are combined together to form CRTM. To fulfill trust measuring, hash computing code and TPM driver are embraced in CRTM. Here TPM must be driven as much as early to satisfy the demands of integrity verification and event logging. USB driver is immigrating from DXE phase into CRTM so that when self-recovery code is invoked, the USB storage device can be accessed. So CRTM of UTBIOS is constructed from following five parts: initial code for CPU, chipset, memory and stack; TPM Driver code; hash computing code; self-recovery code including USB driver; PEI Core Code. In UTBIOS, CRTM is

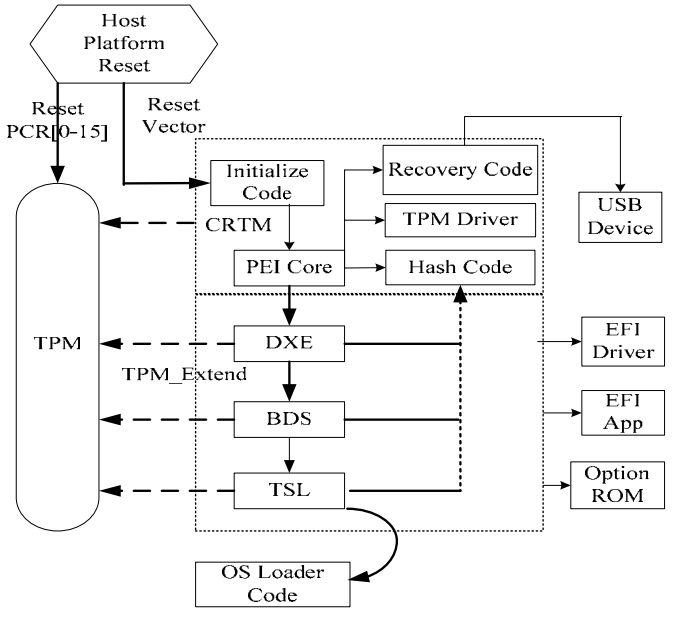

Fig. 4. UTBIOS Architecture and Boot Process

stored at the highest 64KB block of the flash ROM, and is protected physically to avoid being rewritten or flashed using pure software.

Figure 4 shows UTBIOS architecture and boot process.

The boot process is as following: 1) Initialize CPU, chipset, memory and Stack, build the environment of executing language C code; 2) Enter into PEI core, initialize TPM, install TPM protocol, self-recovery protocol including USB driver and Hash Protocol; 3) PEI core measures trust of DXE core, saves trust measuring event log [14][15]. If successful, loads and executes DXE core, else enters into trusted self-recovery mode; 4) According to the requirements of executing environment, DXE core loads internal or external EFI Drivers, EFI Applications, or Option ROM of other devices. During this procedure, DXE core measures trust of those loaded code and saves trust measuring event logs. If successful, loads and executes code, else enters into trusted self-recovery mode; 5) DXE core measures trust of BDS code and saves trust measuring event log. If successful, loads and executes BDS core, else enters into trusted self-recovery mode; 6) According to the requirements of executing environment, BDS core loads Option ROM of boot device, measures trust of the code and saves trust measuring event log. If successful, loads and executes the code, else enters into trusted self-recovery mode; 7) BDS core measures trust of TSL code, saves trust measuring event log. If successful, loads and executes TSL code, else enters into trust self-recovery mode; 8) TSL core measures trust of OS Loader code, saves trust measuring event log. If successful, loads and executes OS Loader code, else enters into trusted self-recovery mode.

When finishing OS loader trust measurement, the UTBIOS transfers system control to OS Loader and the trust measurement of OS Kernel will be measured by OS Loader. As so far, the pre-OS chain of trust is constructed successfully.

After enter into trusted self-recovery mode, the self-recovery protocol that installed in CRTM should be executed and with responsibility for reading trusted BIOS image except CRTM from USB device and writing it back to flash ROM. During the process of trusted self-recovery, the loaded BIOS image need to be trust measured.

6 UTBIOS Trust Encapsulation and Trust Measurement

6.1 Principle of Trust Encapsulation and Trust Measurement

UTBIOS uses digital signature and message digest for trust measurement of various entities. In the process of producing the UTBIOS, vendors are required to bind each component together with the digital signature of message digest of the component. This procedure is called trust encapsulation. All components except CRTM are required to be encapsulated. Trust measurement is the reversed procedure of trust encapsulation. In trust measuring procedure, digital signature is decrypted to get the message digest being signed, and then message digest of the entity being measured is calculated again and compared with the decrypting result to verify the integrity of the entity. The public and private keys used in this procedure are corresponding unique, so it is feasible to judge whether the entity is come from trusted owner or not, and whether the entity is juggled or not. The principle of trust encapsulation and trust measurement is illustrated in Figure 5 and Figure 6.

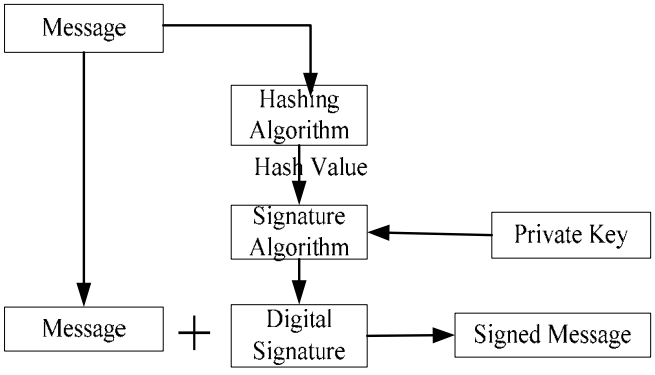

Fig. 5. Trust Encapsulation

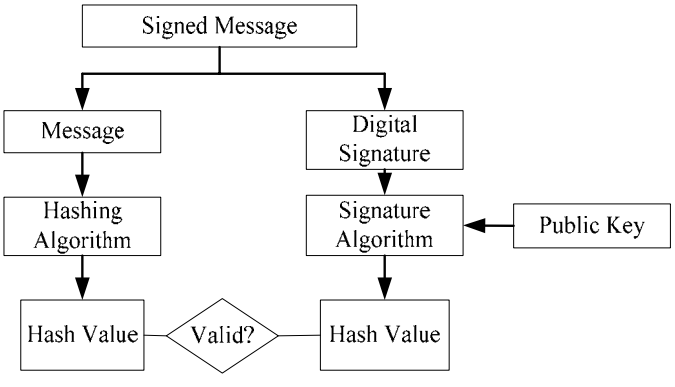

Fig. 6. Trust Measurement

Think about a kind of attack, attackers sign malicious code using their own private key, and replace the trusted BIOS verification public key with their own public key. Then when the malicious code is trust measured, the measuring result would be successful. In this situation, the attackers can embed malicious code into BIOS and run them successfully. In order to prevent this kind of attack, the public key, which is used for trust measurement, can not be stored at the same place with measured module, and should be stored sealed. The best way to prevent this attack occurring is storing public key in TPM. In the trust measurement process, the encapsulated signature of the measured entity is taken apart and sent to TPM, and the verification computing is finished in TPM.

6.2 Trust Initialization

Legacy Option ROM and OS Loader Code maybe not be encapsulated and signed when the system executed at the first time the computer is used by user, so there need a trust initialization procedure when the system run firstly. Following is the way of UTBIOS trust initialization. UTBIOS would self-create a couple of keys automatically when the

system runs at first. UTBIOS computes the hash value of exterior entity and uses the private key to sign the hash value of the entity, and discards the private key after the signature. The public key would be sealed and protected by TPM, and the digital signatures of the external entities are preserved in BIOS Flash's NVRAM data area. Once the trust initialization finished, subsequent bootstrap process after that can use the initializing result (including the public key in TPM and the digital signatures in NVRAM) to measure the trust of exterior entities such as unsigned Option ROM and OS Loader.

On the platform in which all exterior entities are signed, there is no need for using trust initialization.

7 Performance

Comparing UTBIOS with Legacy BIOS, the performance of bootstrap speed is affected mainly by trust measurement. The main operations of trust measurement include hash value computing and signature verification. The SHA-1 hash algorithm and RSA Verification whose key length is 2048bits are used in UTBIOS. Signing a message takes much longer than verifying the signature does, but in the process of bootstrap of UTBIOS, there is only verification operation, no signing operation.

We can compute the estimated increased extra time consumed by trust measurement in UTBIOS using following equation:

$$T = T(L1) + T(L2)$$

$$T(L1) = \sum_{i=1}^{n} t(m_i) = \sum_{i=1}^{n} (t(H(i)) + t(V(i)))$$

$$T(L2) = t(H(L2)) + t(V(L2))$$

T (L1) is the total time that consumed by UTBIOS to load the EFI Driver, EFI Application and Option ROM for internal and external entities. T (L2) is the time consumed by trusted BIOS to measure OS Loader Code. t (mi) is the time to measuring the ith entity. t (H()) means the time to compute SHA-1 digest for the ith entity, t (V()) means signature verification time of the ith entity.

On the machine with 1GHz Pentium CPU, creating SHA-1 digest for 1Mbytes message needs 13ms. Table 1 is the time used for SHA-1 algorithm and public keys with different length to check 1500000 Bytes data for one time [17].

Table 1. Digital signature verification times

Hash function	Public key sizes(bits)	Verification time(seconds)
SHA-1	512	0.093
SHA-1	768	0.093
SHA-1	1024	0.094
SHA-1	2048	0.098

SINOSUN SSX3B TPM is used in the implementation, the verification time for 2048 bit of SSX3B TPM is less than 40ms [16].

For the ith entity with 1Mbytes length:

t (H(i))=0.013 seconds
t (V(i))=0.040 seconds

This means the Trust Measurement of measuring 1Mbytes entity costs 0.053 seconds. In UTBIOS，the sum of entity is less than 100 and the size of entity is less than 0.5 Mbytes. So T (L1) is estimated to be 2.7 seconds. The size of OS Loader Code is always less than 10MBytes, the time required to verify OS Loader is about:

t (H(L2))=0.13 seconds
t (V(L2))=0.40 seconds

And T (L2) is estimated to be 0.53 seconds. The T is estimated to be 3.23 seconds. Actually, the sum of entities and it size that needed to be measured by UTBIOS are far less than the maximum number used to be estimated; thereby the actual increase extra time is less than this estimated time.

Experiment results show that the time of UTBIOS boot without Trust Measurement is 11 seconds and the time of UTBIOS boot with Trust Measurement is 16 seconds. Because of device accessed and data exchanged, the actual Trust Measurement extra time is about 4 seconds, which is longer than computed estimated time 3.23 seconds.

8 Conclusions

Without changing the current computer architecture, BIOS should become the Core Root of Trusted Measurement and the root of transitive trust. Based on this idea, this paper proposed a trusted BIOS architecture, implemented a trusted BIOS-UTBIOS which is build on Intel Innovation framework for EFI/UEFI. UTBIOS can not only defend against effectively the security threats faced by firmware, but also can construct pre-OS trust chain successfully. To build a trusted computing terminal and trusted network connection, there have two problems to be solved next: one is to establish practicable trust management mode, the other is to construct post-OS chain of trust.

References

1. Arbaugh, W.A., Farber, D.J., Smith, J.M.: A Secure and Reliable Bootstrap Architecture. In: Procedings, 1997 IEEE Symposium on Security and Privacy (4-7 May 1997) pp. 65–71 (1997)
2. Kozen, D.: Efficient Code Certification. Technical Report98-1661,Computer Science Department, Cornell University (January 1998)
3. Adelstein, F., Stillerman, M., Kozen, D.: Malicious Code Detection for Open Firmware. In: Computer Security Applications Conference. Proceedings 18th Annual (9-13 Decembr 2002) pp. 403–412 (2002)
4. Heasman, J.: Implementing and Detecting an ACPI BIOS Rootkit, http://www.ngssoftware.com/jh_bhf2006.pdf

5. Clark, D.D., Wilson, D.R., Comparison, A.: A Comparison of Commercial and Military Computer Security Policies. In: Proceedings of the 1987 IEEE Symposium on Security and Privacy, IEEE Computer Society Press, Los Alamitos (1987)
6. The Unified, E.F.I.: Forum. Unified Extensible Firmware Interface Specification Version 2.0 (January 31, 2006), http://www.uefi.org
7. Intel Corporation. Intel Platform Innovation Framework for EFI Architecture Specification Version 0.9 (September 16, 2003)
8. TCG. TCG Infrastructure Architecture Version 1.0. https://www.trustedcomputinggroup.org/specs/
9. TianoCore, https://www.tianocore.org/
10. UEFI, https://www.uefi.org/
11. Intel Corporation. Intel Platform Innovation Framework for EFI Firmware File System Specification Version 0.9 (September 16, 2003), http://www.intel.com/technology/framework/
12. TCG. TPM Main Specification Part 1,2,3 Version 1.2 (March 29, 2006) https://www.trustedcomputinggroup.org/specs/
13. TCG. TCG Specification Architecture Overview, https://www.trustedcomputinggroup.org
14. TCG.TCG EFI Platform Version 1.0 Final Revision 1.00, https://www.trustedcomputinggroup.org
15. TCG.TCG EFI Protocol Version 1.0 Final Revision 1.00, https://www.trustedcomputinggroup.org
16. Sinosun. SSX35, T.P.M.: Datasheet Version 1.2
17. Menasce, D.A.: Security Performance[J]. IEEE Internet Computing 7(3), 84–87 (2003)