Enhanced Architecture of TPM

Fenghua Li
Key Laboratory of CNIS, Xidian University, Xi'an 710071, China
Graduate School, Beijing Electronic Science and Technology Institute, Beijing 100070, China
lfh@best.edu.cn

Wei Wang
Key Laboratory of CNIS, Xidian University, Xi'an 710071, China
wei_wang@mail.xidian.edu.cn

Jianfeng Ma
Key Laboratory of CNIS, Xidian University, Xi'an 710071, China
jfma@mail.xidian.edu.cn

Zhenguo Ding
Key Laboratory of CNIS, Xidian University, Xi'an 710071, China

Abstract

An enhanced architecture of TPM (Trusted Platform Module) is presented in this paper. Besides the previous components, a new special information I/O interface is added. By connecting with the various parallel or serial trusted devices outside TPM, the new special information I/O interface replaces the function of physical-presence and implements the preconfiguration, backup and restoration of information within TPM. At the same time, the service methods of new I/O component are proposed. By these methods, we can get the identity of the connecting device outside TPM, check the state of TPM, preconfiguration the initial or migratory information, backup or restore the sensitive information within TPM. Finally, an example shows that the security operations on the sensitive information within TPM can be implemented by the new special information I/O interface.

Keywords: TPM, security operations, special information I/O interface, sensitive information, service methods.

1. Introduction

As one of the information security modules, the Trusted Platform Module (TPM) is usually established on the chip with main processor, such as the motherboard. In the TPM Specification Version 1.2, TPM implements the specified functions, by connecting the motherboard with main processor and interacting with the software platform following the TSS commands. In 2005, the detailed design and applications of trusted computing platforms are presented [1]. In 2007, Shen et al. survey the security problems including trusted computing [2]. However, there are some problems in the trusted computer with TPM according to TCG Specification [3, 4].

(1) The initial information stored in non-volatile memories is configured by TPM manufacturers. Whereas, to use the TPM conveniently, the TPM Owner usually expects that some information in TPM are reserved after the Clear operation.

(2) Since a computer usually should be updated or replaced by another one, the computer operator or TPM user needs to copy the whole content in one computer to others. But if the backuped keys or restored keys in TPM are transmitted on the system bus, they may be intercepted and captured by the trojan horse or virus programs.

(3) In TPM, several commands, such as Clear, Deactivate and Disable, require physical presence at the platform before the command will operate. However, when the TPM Owner preconfigures, backups or restores the information in TPM, there are not any methods to achieve the security by physical presence.

Several improved architectures of TPM are presented in [5-8], but there still exist some limitations. For example, (1)If the TPM considers the user′s computer is not trusted, the files in the computer may not be opened by the computer user. (2)To achieve the trust, users will be forced to install hardwares and softwares signed by manufacturer of trusted computer (with TPM). (3)Mistakes and failures will not be avoidable when users use TPM. If these events happen,

978-0-7695-3398-8/08 $25.00 © 2008 IEEE
DOI 10.1109/ICYCS.2008.254

TPM users cannot restore the secret information in TPM and the data encrypted by this secret information.

In this paper, a new special information I/O interface is introduced and an enhanced TPM is presented. The new I/O component can replace the function of physical-presence and implement the functions of preconfiguration, backup and restoration of information within TPM.

2. The Enhanced TPM

2.1 The Architecture of Enhanced TPM

To overcome the shortcomings of TPM Specification Version 1.2, a new architecture of TPM and its service methods are presented. In Fig 1, the special information I/O interface interacts with the execution engine in the TPM, implements the information preconfiguration, backup and restoration by connecting various parallel or serial trusted devices, and replaces the ability of physical presence.

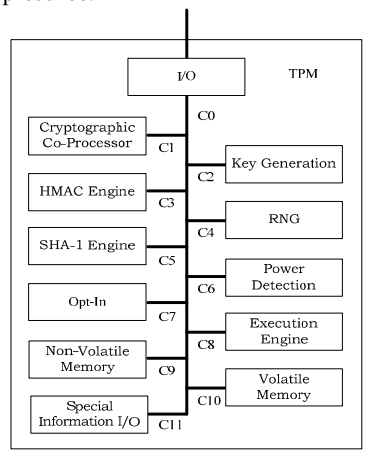

Fig 1. Enhanced TPM architecture

The special information I/O interface is a component interacting with trusted devices outside TPM but not with the CPU on motherboard. So, the program executed by CPU cannot access the special information I/O interface. Except the security, there are no additional limits on the special information I/O interface. The special information I/O interface may be various parallel or serial interfaces, such as USB, smart card, UART, PS/2, RS232, RS422, RS485, and so on. When connecting with the trusted devices, let the state of physical presence is TRUE. Thus, the special information I/O interface replaces the ability of physical presence.

2.2 The Service Methods of Enhanced TPM

From Fig 2, the service methods of the enhanced TPM are as follows:

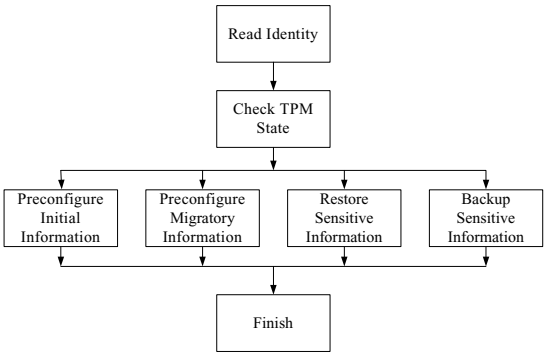

Fig 2. Service methods of the enhanced TPM

Step 1. Read the identity

After receiving the request of the TPM Owner, the TPM executes the TPM_ReadSpecialIOID command. The TPM Owner checks the validity of the connecting device by the identity information ID and its HASH value SHA_ID returned from the application software. If the validity is confirmed, the state of SpecilIO is set to TURE, otherwise FALSE.

Step 2. Check the state of TPM

Firstly, the TPM checks the state of SpecilIO. If the state of SpecilIO is FALSE, the TPM does nothing; otherwise, the TPM performs the following operations. Then, the TPM checks the state of the special information I/O interface, if the state is not active, the TPM does nothing; otherwise, the TPM performs the Step 3, 4, 5 or 6. Here, the state of special information I/O interface can be set by commands TPM_Set SpecialIOActived or TPM_SetSpecialIODeactivated.

Step 3. Preconfigure the initial information

The TPM Owner firstly specifies the public information needs to be transmitted. Then, the TPM performs the preconfiguration operation by command TPM_InitConfigSpecialIOContext and creates the related keys according to the public information. After that, the state of TPM is specified by the TPM Owner and a message TPM_InitConfig_Success is returned. By these operations, the TPM Owner can customize the initial information of TPM, and avoid the TPM manufacturer controlling the key of TPM Owner.

Step 4. Preconfigure the migratory information

To operate on the same key and encrypted data by different TPM, it is necessary to transfer the information in a TPM to another one, we called it migration. Firstly, the TPM Owner inputs the password and specifies the public information needs to be transmitted. Then, the TPM checks the password. If the password is correct, TPM decrypts the transmitted information by the key encrypted by the password and executes the command TPM_MigConfigSpecialIO Context, which transmits the public information from the connecting device to the

non-volatile memory of TPM. After that, the state of TPM is specified by the TPM Owner and a message TPM_MigConfig_Success is returned.

Step 5. Backup the sensitive information in TPM

Firstly, the TPM Owner inputs the password and specifies the information needs to be backuped. Then, the TPM checks the password. If the password is correct, TPM encrypts the transmitted information by the key encrypted by the password and executes the command TPM_BackupSpecialIOContext. After transmitting the information from the non-volatile memory of TPM to the connecting device, a message TPM_Backup_Success is returned.

Step 6. Restore the sensitive information in TPM

Firstly, the TPM Owner inputs the password and specifies the information needs to be restored. Then, the TPM checks the password. If the password is correct, TPM decrypts the transmitted information by the key encrypted by the password and executes the command TPM_RestoreSpecialIOContext. After transmitting the information from the connecting device to the non-volatile memory of TPM, a message TPM_Restore_Success is returned.

2.3 The Storing Style of Data in outside Device

The information transmitted by the special information I/O interface are stored in trusted devices and are organize by file format, where the file types are as follows:

(1) identification file: consisting of IDI (the identity information), N_ID (the length of user information), UI (the user information, N_ID bytes) and SHA_ID (the hash value of identification file).

(2) key file: encrypted by the cryptographic co-processor with user's password. This file consists of IDK (the identity of key file), N_KEY (the length of encrypted key), C_KEY (the encrypted keys, N_KEY bytes) and HMAC_KEY (the HMAC value of key file).

(3) TPM information file: encrypted by the key stored in key file, consisting of following components:

(a) NC (the number of encrypted information), IDC_1 (the identity of encrypted information), LC_1 (the length of IDC_1), C_1 (the encrypted information corresponding to IDC_1, LC_1 bytes), $HMAC\_C_1$ (the HMAC value of related information with IDC_1), …, IDC_{NC} (the identity of encrypted information), LC_{NC} (the length of IDC_{NC}), C_{NC} (the encrypted information corresponding to IDC_{NC}, LC_{NC} bytes), $HMAC\_C_{NC}$ (the HMAC value of related information with IDC_{NC});

(b) NM (the number of public information), IDM_1 (the identity of public information), LM_1 (the length of IDM_1), M_1 (the public information corresponding to IDM_1, LM_1 bytes), $HMAC\_M_1$ (the HMAC value of related information with IDM_1), …, IDM_{NM} (the identity of public information), LM_{NM} (the length of IDM_{NM}), M_{NM} (the public information corresponding to IDM_{NM}, LM_{NM} bytes), $HMAC\_M_{NM}$ (the HMAC value of related information with IDM_{NM}).

When the TPM Owner performs the preconfiguration, backup and restoration operations, he only needs to find the ID value of related information in the TPM information file, and then execute the corresponding operations.

2.4 The Data Structure of Transmitted Information and Parameters of Commands

The data structure of information transmitted by the special information I/O interface is shown in Table 1. From Fig 3, there are seven extend commands corresponding to the special information I/O interface, so as to preconfigure the initial information and migratory information, backup and restore the sensitive information in TPM. The related commands are listed as follows.

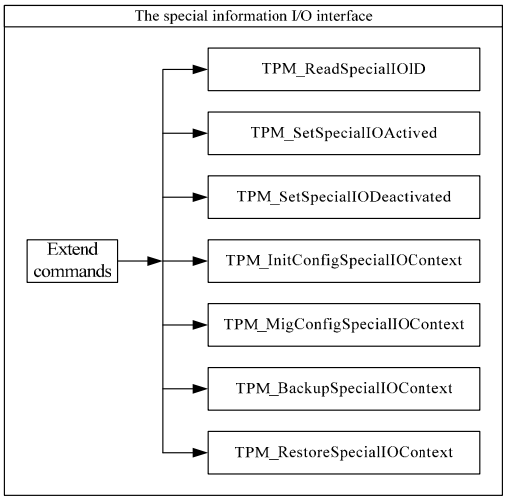

Fig 3. Extend commands corresponding to the special information I/O interface

The TPM Owner reads the identity of devices connecting with TPM by command TPM_ReadSpecialIOID, which is shown in Table 2. The format STRING is a character string ended by bit 0 and defined by users.

(1) The system operator executes the command TPM_SetSpecialIOActived to permit the information preconfiguration, backup and restoration operations between TPM and trusted devices, executes the command TPM_SetSpecialIODeactivated to forbid the information preconfiguration, backup and restoration operations. These two commands are shown in Table 3.

Table 1. The data structure of information transmitted by the special information I/O interface

Definition

```
typedef struct tdTPM_TRANSPORT_ID{
        TPM_STRUCTURE_TAG tag;
        UINT32 lengthENCID;
        TPM_TRANSPORT_ENC[] transENCID;
        UINT32 lengthPUBID;
        TPM_TRANSPORT_PUBLIC[] transPUBID;
} TPM_TRANSPORT_ID;
```

Parameters

Typedef	**Name**	**Description**
TPM_STRUCTURE_TAG	tag	TPM_TAG_TRANSPORT_ ID
UNIT32	lengthENCID	The array length of identities IDC_x corresponding to encrypted information
TPM_TRANSPORT_ENC	transENCID	The array of identities IDC_x corresponding to encrypted information
UNIT32	lengthPUBID	The array length of identities IDM_x corresponding to public information
TPM_TRANSPORT_PUBLIC	transPUBID	The array of identities IDM_x corresponding to public information

Table 2. The parameters of extend command TPM_ReadSpecialIOID

Incoming Parameters and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RQU_COMMAND
2	4			UINT32	paramSize	Total number of input bytes including paramSize and tag
3	4			TPM_COMMAND_CODE	ordinal	Command ordinal, fixed value of TPM_ORD_ReadSpecialIOID

Outgoing Parameters and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RSP_COMMAND
2	4			UINT32	paramSize	Total number of output bytes including paramSize and tag
3	4			TPM_RESULT	returnCode	The return code of the operation
4	<>			STRING	deviceInformation	The related information of connecting device

Table 3. The parameters of extend command TPM_SetSpecialIOActived and TPM_SetSpecialIODeactivated

Incoming Parameters and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RQU_COMMAND
2	4			UINT32	paramSize	Total number of input bytes including paramSize and tag
3	4			TPM_COMMAND_CODE	ordinal	Command ordinal, fixed value of TPM_ORD_SetSpecialIOActived
4	1			BOOL	state	The state is active or not.

Outgoing Parameters and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RSP_COMMAND
2	4			UINT32	paramSize	Total number of output bytes including paramSize and tag
3	4			TPM_RESULT	returnCode	The return code of the operation

Incoming Parameters and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RQU_COMMAND
2	4			UINT32	paramSize	Total number of input bytes including paramSize and tag
3	4			TPM_COMMAND_CODE	ordinal	Command ordinal, fixed value of TPM_ORD_SetSpecialIODeactivated
4	1			BOOL	state	The state is deactive or not.

Outgoing Parameters and Sizes

PARAM		HMAC		Type	Name	Description
#	SZ	#	SZ			
1	2			TPM_TAG	tag	TPM_TAG_RSP_COMMAND
2	4			UINT32	paramSize	Total number of output bytes including paramSize and tag

Table 4. The parameters of extend command TPM_InitConfigSpecialIOContext

Incoming Parameters and Sizes

PARAM #	PARAM SZ	HMAC #	HMAC SZ	Type	Name	Description
1	2			TPM_TAG	tag	TPM_TAG_RQU_COMMAND
2	4			UINT32	paramSize	Total number of input bytes including paramSize and tag
3	4			TPM_COMMAND_CODE	ordinal	Command ordinal, fixed value of TPM_ORD_InitConfigSpecialIOContext
4	1			BOOL	keepHandle	Indication if the handle MUST be preserved
5	4			UINT32	infoSize	Total number of bytes of preconfigured initial information
6	<>			TPM_TRANSPORT_ID	preconfInitInfo	The ID set of preconfigured initial information

Outgoing Parameters and Sizes

PARAM #	PARAM SZ	HMAC #	HMAC SZ	Type	Name	Description
1	2			TPM_TAG	tag	TPM_TAG_RSP_COMMAND
2	4			UINT32	paramSize	Total number of output bytes including paramSize and tag
3	4			TPM_RESULT	returnCode	The return code of the operation
4	4			TPM_HANDLE	handle	Handle of the initial information being preconfigured.

Table 5. The parameters of extend command TPM_MigConfigSpecialIOContext

Incoming Parameters and Sizes

PARAM #	PARAM SZ	HMAC #	HMAC SZ	Type	Name	Description
1	2			TPM_TAG	tag	TPM_TAG_RQU_COMMAND
2	4			UINT32	paramSize	Total number of input bytes including paramSize and tag
3	4			TPM_COMMAND_CODE	ordinal	Command ordinal, fixed value of TPM_ORD_MigConfigSpecialIOContext
4	1			BOOL	keepHandle	Indication if the handle MUST be preserved
5	4			UINT32	infoSize	Total number of bytes of preconfigured migratory information
6	<>			TPM_TRANSPORT_ID	PreconfMigInfo	The ID set of preconfigured migratory information

Outgoing Parameters and Sizes

PARAM #	PARAM SZ	HMAC #	HMAC SZ	Type	Name	Description
1	2			TPM_TAG	tag	TPM_TAG_RSP_COMMAND
2	4			UINT32	paramSize	Total number of output bytes including paramSize and tag
3	4			TPM_RESULT	returnCode	The return code of the operation
4	4			TPM_HANDLE	handle	Handle of the migratory information being preconfigured.

Table 6. The parameters of extend command TPM_BackupSpecialIOContext

Incoming Parameters and Sizes

PARAM #	PARAM SZ	HMAC #	HMAC SZ	Type	Name	Description
1	2			TPM_TAG	tag	TPM_TAG_RQU_COMMAND
2	4			UINT32	paramSize	Total number of input bytes including paramSize and tag
3	4			TPM_COMMAND_CODE	ordinal	Command ordinal, fixed value of TPM_ORD_BackupSpecialIOContext
4	4			TPM_HANDLE	handle	Handle of the information being backuped
5	<>			TPM_TRANSPORT_ID	backupInfo	The ID set of backuped information

Outgoing Parameters and Sizes

PARAM #	PARAM SZ	HMAC #	HMAC SZ	Type	Name	Description
1	2			TPM_TAG	tag	TPM_TAG_RSP_COMMAND
2	4			UINT32	paramSize	Total number of output bytes including paramSize and tag
3	4			TPM_RESULT	returnCode	The return code of the operation
4	4			UINT32	infoSize	Total number of bytes of backuped information

(2) The system operator preconfigures the initial information of TPM by the command TPM_SetSpecialIOActived which is shown in Table 4.

(3) The system operator preconfigures the migratory information by the command TPM_MigConfigSpecialIOContext shown in Table 5.

(4) The system operator backups the sensitive information in TPM by the command TPM_MigConfigSpecialIOContext shown in Table 6.

(5) The system operator restores the sensitive information from the trusted device to TPM by the TPM_RestoreSpecialIOContext shown in Table 7.

Table 7. The parameters of extend command TPM_RestoreSpecialIOContext

Incoming Parameters and Sizes

PARAM #	PARAM SZ	HMAC #	HMAC SZ	Type	Name	Description
1	2			TPM_TAG	tag	TPM_TAG_RQU_COMMAND
2	4			UINT32	paramSize	Total number of input bytes including paramSize and tag
3	4			TPM_COMMAND_CODE	ordinal	Command ordinal, fixed value of TPM_ORD_RestoreSpecialIOContext
4	1			BOOL	keepHandle	Indication if the handle MUST be preserved
5	4			UINT32	infoSize	Total number of bytes of restored information
6	<>			TPM_TRANSPORT_ID	restoreInfo	The ID set of restored information

Outgoing Parameters and Sizes

PARAM #	PARAM SZ	HMAC #	HMAC SZ	Type	Name	Description
1	2			TPM_TAG	tag	TPM_TAG_RSP_COMMAND
2	4			UINT32	paramSize	Total number of output bytes including paramSize and tag
3	4			TPM_RESULT	returnCode	The return code of the operation
4	4			TPM_HANDLE	handle	Handle of the information being restored.

3. Application Example

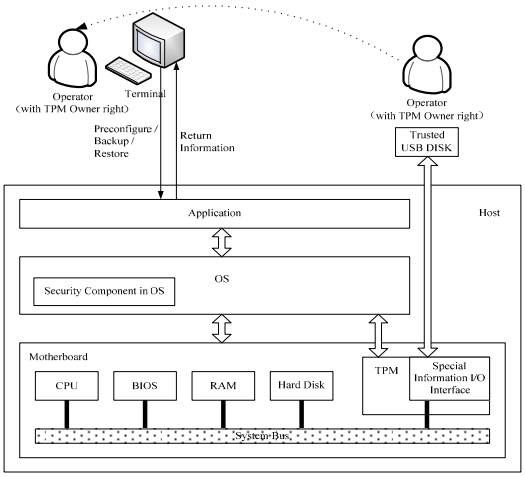

Fig 4. Application example of the enhanced TPM

The example of the enhanced TPM is shown in Fig 4. Firstly, the operator with TPM Owner's right connects the trusted USB DISK with the special information I/O interface. Then, the identity information of the USB DISK is read. The operator checks the validity and correctness of USB DISK by the returned identity information. Thereafter, the state of special information I/O interface is active by the command TPM_SetSpecialIOActived. After specifying the preconfigured, backuped or restored information by the application software, the corresponding operations are performed. Based on the requested operation, the related step in section 2.2 is carried out. If TPM finds that an operation makes a mistake, an error message is returned to the operator. If all the operations are executed correctly, a success message is returned. From the example, the functions of preconfiguring, backuping and restoring the sensitive information in TPM are implemented by the special information I/O interface.

4. Conclusions

An enhanced TPM is presented in this paper. The new I/O component can replace the function of physical-presence. After describing the data structure of information transmitted by the special information I/O interface and related commands, the preconfiguration, backup and restoration of information within TPM are proposed.

Acknowledgements

This work is supported by the National Natural Science Foundation of China(60633020, 60702059 and 60573036), and the National High Technology Research and Development Program of China (2007AA01Z472, 2007AA01Z429 and 2007AA01Z482).

References

[1] S. W. Smith, "Trusted Computing Platforms: Design and Applications", Springer. 2005.

[2] C. X. Shen, H. G. Zhang, D. G. Feng, et al, "Survey of information security", *Science in China Series F*. 2007, 50(3), pp. 273-298.

[3] TCG, "TPM Main Part 2 TPM Structures", Specification Version 1.2. www.trustedcomputing group.org. 2005.

[4] TCG, "TPM Main Part 3 Commands", Specification Version 1.2. www.trustedcomputing group.org. 2005.

[5] J. Camenisch, "Better Privacy for Trusted Computing Platforms", *In proceedings of ESORICS 2004*. LNCS 3193. Springer-Verlag. 2004, pp. 73-88.

[6] T. Eisenbarth, T. Güneysu, C. Paar, "Reconfigurable Trusted Computing in Hardware", *In proceedings of STC'07.* ACM Press. 2007, pp. 15-20.

[7] P. James, D. Rivera, "Systems, Methods, and Media for Accessing TPM Keys", US2006129824. 2004

[8] Z. Yan, P. Cofta, "A Mechanism for Trust Sustainability Among Trusted Computing Platforms", *In proceedings of TrustBus 2004.* LNCS 3184. Springer-Verlag. 2004, pp. 11-19.