A Novel Secure Terminal System Based on Trusted Hardware: U-Key

Wei-Peng Liu and Jun Hu
State Key Laboratory of Information Security, Graduate school of Chinese Academy Science,
Beijing, 100039 China
Email: weipengliu0209@163.com

Xing Zhang
PLA Electronic Technology Institute of Information Engineering University, Zhengzhou, 450052 China
Email:zhangxing99@163.com

Abstract—**With the development of attack technique, new security problems have become more and more popular, traditional security means such as: firewall, IDS and so on expose to be limited, there is a basic consensus that solving the problem of information security is from terminal. Based on the deep analysis of the relation between the concept of "security" and "trusted" in computer system, this paper proposes a novel secure terminal system Based on trusted hardware: U-Key. It describes the architecture design of the whole secure system and analyses the important security function implemented. Moreover, the main performance overhead of system is analyzed. This paper gives illumination on how to take advantage of trusted hardware to enhance the security of terminal system。**

Index Terms—**terminal, U-Key, trusted computing, security function, security policy, security objective**

I. Introduction

The traditional method to protect information system security is to make use of security software and hardware, such as firewall, security router, security gateway, IDS (Intrusion Detection System) and vulnerability scanner to inspect and defend in the periphery of system. However, with the attack technology improving and new security question emerging, the traditional security method exposes to be limited. Nowadays, people have come to realize to solve information security problem from terminal. But on one hand, the architecture of most terminals used in practice is simple and some security functions provided by hardware is never used [1], so the configurations of hardware and the resource are easy to be tampered and misused. On the other hand, because of lacking of the protection from high level security operating system, especially the protection from mandatory access control mechanism, it is easy for mal-ware to attack and even more achieve the privilege of the super user by exploring the vulnerability of terminal. Furthermore, due to the lack of the guarantee of trusted mechanism, the correct running of terminal and security function can not be assured adequately. Recently, the rising of trusted computing provides new ideas and methods to enhance the security of terminal.

U-Key is a trusted hardware, the function of which mainly includes identity authentication; secure storage of sensitive or important data, symmetrical-key and unsymmetrical-key encryption and so on. After the system booting, by taking U-Key as the trusted root and building the trusted chain from U-Key to Os-loader, Os-kernel until to topper-application, thus it can assure that the booting process of whole system is trusted and that the security mechanism is not tempered and bypass. By using U-Key, it can authenticates user identity in the way of double-factor and grant the right to user based on the authentication; By storing cipher key in the U-Key, it can assure that the key is not tempered and not fall to the ground, thus it makes the security function relative to the key executed safely and reliably; The main work of this paper is to use main function of trusted hardware: U-Key to design and implement a secure terminal system.

The paper is arranged as follows: based on the analysis of the relation between the concept of "security" and "trusted" in computer system, the section II and III proposes and describes a novel secure terminal system based on the trusted hardware: U-Key; The section IV analyses the important trusted and security function implemented in system; Relative work is reviewed in section V; This paper concludes in section VI with a outlook of future research work.

II. The Architecture Of Secure Terminal System

"Security" and "trusted" are the focus (two important concepts) of information security in academe and industry, but the distinction between them is fuzzy. The architecture design of our secure terminal system is based on the analysis of "security" and "trusted", so we must try to make the difference between them clear.

TCG (Trusted Computing Group) [2] defines "trusted" in the manner of behavior and considers that when an entity is always achieving the expected objective in expected manner, then it is “trusted”. In the international standard ISO/IEC 15408 [3], it gives a definition of “trusted” from the view of system: a trusted component, operation or process is expected in the any operation and can withstand the attack or damage from application software, virus and physical interference. We think that the core of “trusted” is the expectation of behavior. But the behavior expectation to entity/component must be under a certain precondition; furthermore, it usually needs to receive input from the exterior in a certain state on a certain time, at last achieves a new state on a new time. So expectation, precondition, input, state and time are five essential factors for the concept “trusted”. We can define “trusted” as: under a certain precondition, a component receives input from the exterior in a certain state on certain time and achieves the expected state on new certain time.

“Security” is not absolute; it is a relative concept that can be implemented by some certain technology and under a certain condition. NIST describes security objective as [4] “enable an organization to meet all of its mission/business objectives by implementing systems with due care consideration of IT-related risks to the organization, its partners and customers.” Security policy which is the formal description of security requirement in certain environment, servers for security objective and regulates that what rules it should comply with to achieve the security objective. There is no uniform security policy, but certain security policy is for certain security requirement and certain security objective. People customarily partitions security objective as: availability, integrity, confidentiality, accountability and assurance. “Security” in actual environment can be considered as the approach to the security objective by utilizing trusted mechanism and security mechanism. So we can define security as: in a certain environment, the approach to the security objective by utilizing trusted mechanism and security mechanism.

Based on the above analysis, we consider that “trusted” emphasizes much on the expectation and the validation of the system state in objectivity, it is internal factor and it is absolute, at this point, it is not same as “security”. “Security” focuses on the objective and the result, it is an explicit manifestation, it is not absolute, in actual system, security objective is very concrete and usually it is the reflection of security policy and security requirement. If some security objective, such as integrity, confidentiality and so on can be met, then we can say that it is secure.

We think that “trusted” is the base of security function for executing correctly, and the expectation and the validation of the system behavior is the prerequisite of system security. So the relation between “security” and “trusted” can be described as: “trusted” is the necessary condition to “security”, only if the system is trusted can the system achieve the security objective, furthermore the security. “Trusted” of system shows that the system itself is running correctly, at the time, it is the guarantee and the base to carry into security mechanism and security function execution correctly. In actual system, “trusted” servers for “security”, for example, it is not necessary for operating system to encrypt the code of itself, but it must assure the integrity of code, or it can not provide the topper-application with encryption service. So it can say that only if the system is trusted can it achieve security objective. On the contrary, if the operating system can not assure that it is trusted, for example, if the system itself is tampered, thus the base of security disappears, so it can not judge if the security mechanism implements security function correctly. Moreover, the relation between “security” and “trusted” is dialectic. “Trusted” provides the base for “security”, thus assures that the security mechanism executes correctly and achieves the final security objective, at the same time, “security” reacts to “trusted”. Usually, in actual system, it is difficult to achieve complete “trusted”, but by using some security mechanisms, it can alleviate the complex of trusted mechanism implementation. In the work done by IBM [5], it proposes an integrity measurement approach based on information flow integrity, by implementing CW-Lite [6] security model, it solves the question of the system redundant measurement and improves the efficiency. It gives illumination on the design of our architecture.

In paper [9], it points out that: “it has been generally accepted for some time that software alone cannot provide an adequate foundation for building a high-assurance trusted platform”. Usually, “security” can be implemented by security mechanism, and “trusted” also can be implemented by trusted mechanism. Classical security mechanism consists of: identity authentication, access control, encrypting and decrypting and so on. Classical trusted mechanisms consist of: trusted storage, trusted measurement and trusted report. Based on the above discussion, we propose the whole architecture of secure system based on trusted hardware: U-Key as figure1.The main design idea is based on the relation between the concept of “security” and “trusted” addressed above and pays more attention to the combining of “security” and “trusted”. This can guarantee that the security policy enforces correctly and realizes the whole security objective. In architecture, we have implemented four main security functions.

Trusted Hardware Layer: to assure that the basic hardware of the platform is trusted and provide the foundation for security function to execute correctly, the U-Key is the trusted root of the architecture, so it is trusted unconditionally.

Trusted Mechanism Layer: trusted booting is main trusted mechanism implemented. It can assure that the system boots in strict order and each component of the booting process is in a trusted state, thus it can provide a trusted environment for application. The trusted chain starts from U-Key, firstly it measures and validates the integrity of every stage of Grub, then Grub measures and validates Initrd, and then Initrd measures and validates operating system kernel and security module, thus it assures that the security function can not be tempered and bypass. As the expansion, how to implement trusted report mechanism [2] presented by TCG is next research direction.

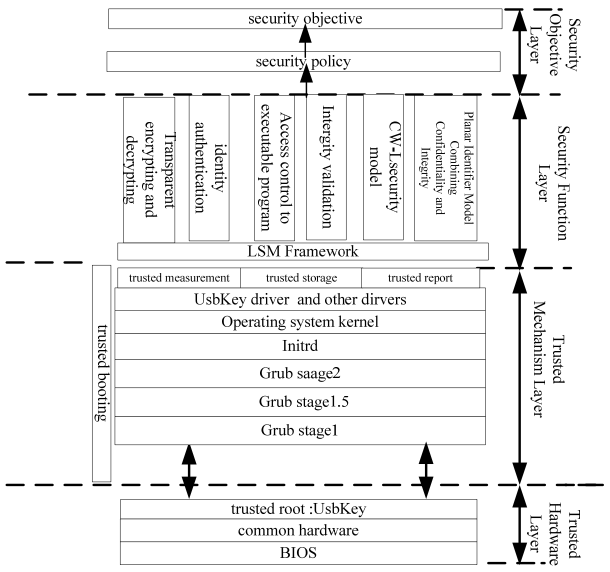

Figure1. The architecture of secure terminal system based on trusted hardware: U-Key

Security Function Layer: Identity authentication is the base of security function and it can assure the validity of user identity. Security function enforces security policy and security policy servers for security objective. Different application environment has different security objective. But confidentiality usually can be considered as one of the most important security objective. As to confidentiality, the function mainly composes of transparent encrypting and decrypting to data; For the high level security operating system, usually mandatory access control mechanism is necessary, so we designs and implements a mandatory access control mechanism for executable program. The security function implementation is based on LSM [13], it is a flexible and general access control framework that supports many classical security policy. It also supports the coexistence of multi-security policy, and can extend the security function easily according to the change of menace and environment.

Security Objective Layer: security policy comes from security requirement and security menace, and at the same time, security policy serves for security objective. Usually, confidentiality and integrity are considered as two important security objectives.

III. The Constitution And Work Flow Of System

The secure terminal system mainly composes of three parts: installing part, configuration and management tool and security module. There are two kinds of U-Key used in the whole system: 0 level key for administrator and 1 level key for ordinary user.

A. Installation Part

The installer can simply and fast finish the whole security system installation by executing installation script in installation CD. The key point of system installation part is finished by modifying Initrd. Initrd is "initialized RAM disk" in short. So before operating system kernel booting and accessing the genuine file system, it will first access Initrd file system which is loaded to system memory by Grub. Thus the kernel booting will be divided into two stages, the task of the first stage is to complete loading driver module and the second stage is to execute /sbin/init which is in root file system. By modifying /init in Initrd file system and copying U-Key driver module, identity authentication module, and security module and so on to /lib which is in Initrd, the whole installation is finished. For the security consideration, the system installer must hold 0 level U-Key. In installation, the system reads information from the U-Key and creates a default administrator user. This user can use configuration and management tool to finish configuring and managing the secure system. At last, it needs to modify the grub configuration file: /boot/grub/menu.list to load the modified Initrd which is used in secure system.

B. Configuration and Management Tool

The main function of configuration and management tool is to finish adding or deleting user and resource. The tool development is based on Linux operating system and uses QT as a graphics development tool. The graphics interface is friendly, convenient and easy to use. Before using the tool, the administrator must first insert 0 level U-Key and login successfully. Before adding a user, it also must insert a user U-Key, when the input password is correct, the operating of adding user is permitted .After successfully adding a user, a private encrypting dir which is used to store private data will be created in /home/ and the "dir name" is the same as "username". After adding a user and adding certain resource to the user, the administrator push "confirm" button to take the configuration into effect.

The attribute of resource includes "encrypting" and "reservation". If the resource has the attribute of "encrypting", the tool will encrypt the resource. The attribute of "reservation" is used to be extended. Each resource in the system has a level including 0 and 1. The level is mainly used to the granting of executable program. 0 level resource can only be executed by 0 level administrator and other user can not execute. While 1 level resource can be executed by 0 level administrator and can also be executed by 1 level user who has been granted.
In the system, the management of resource is based on "group", "group" is the "bridge" between user and resource, it is the same as the concept of role in RBAC model. Each resource can belong to different group, each group can consist of different resource; each user can belong to different group, each group can consist of different user. The granting of resource based on group is

easy to manage resource effectively. In the initialization of tool, it must read information from os210_user.rec (userlist), os210_group.rec (grouplist) and os210_least.rec(least resource list) to build the data in interface, and after the management finished, it will rewrite those files according to actual granting condition. Furthermore, after the management, it will rewrite os210_userlist and os210_reslist file, which is the interface file to security kernel. os210_userlist is the user list file, the secure system bases it to validate user identity. os210_reslist is resource granting file. The security system reads the two files to enforce security function.

C. *Security Module*

The security module is the core of the whole system, which is "insmod" by Initrd. The design and implementation of security module is based on LSM framework. Security module can dynamically register and delete by using Linux Module Mechanism. The security module builds the user "linked list" according to os210_userlist, and then reads os210_reslist to build the security resource "linked list", the whole security function can be completed by comparing and checking between two "linked list". Furthermore, to implement user authentication based on U-Key, we modify login program and use Linux PAM module mechanism。

IV. Security Function Implementation

The whole secure terminal system is developed based on Linux2.6.11 kernel, and uses LSM framework and PAM module mechanism. The main secure and trusted function comprises of: trusted booting, user identity authentication, transparent encrypting and decrypting to data and mandatory access control to executable program.

A. *Implementation of Trusted Booting*

In the computer system, the process of booting is base of the other system behaviors; it begins after the post of BIOS and ends after loading the operating system kernel successfully.

So whether or not the booting is trusted is important to the security of the whole system. Trusted booting which is based on the concept of "trusted chain" was first presented by W.Arbaugh [10], it is defined as: during the system booting, if the integrity measurement value of component is the same as expected measurement value, then the system boots normally, or the system must adopt some remediation such as: system booting halt or trusted recovery. Grub is the default boot loader of Linux operating system, but Grub is an untrusted booting loader, In order to assure "trusted" of system, we design and implement a Trusted Grub: TGrub.

The execution of Grub is divided into three stages: stage1, stage1.5 and stage2. Stage1 mainly comprises of a piece of assemble code which is called by BIOS; Stage1.5 comprises of two piece of assemble code: start.s, asm.s and some piece of C codes. The C code stage1_5.c will load stage2. Stage2 also comprises of two pieces of assemble code: start.s and asm.s and some pieces of C codes. The C code stage2.c will load Initrd and operating system kernel. So as to finishing measuring and validating, we implement a real-mode U-Key driver which is inserted into system after the stage1. The concrete step of trusted booting is as follows:

1. During the installation of TGrub, the system creates the integrity measurement value of various stages (except stage1) and stores them into the U-Key, at the same time, backups the various stages (except stage1) into the U-Key for trusted recovery.

2. In the stage1, the system calls the validation module to validate the integrity of start.s, if successes, then the control right is passed to start.s, or calls the recovery module to recover start.s and the system reboots.

3. In start.s, before the control right passed to the residual of stage1.5, the system calls validation module to validate it, if successes, then the control right is passed to it, or calls the recovery module to recover it and the system reboots.

4. In C code of stage1_5.c in stage1_5，the system calls the validation module to validate stage2, if successes, then the control right is passed to stage2, or calls the recovery module to recover stage2 and the system reboots.

5. In C code of stage2.c in stage2, the system calls the validation module to validate Initrd and operating system kernel, if successes, then the control right is passed to Initrd and kernel, or calls the recovery module to recover Initrd and kernel and the system reboots.

B. *Implementation of User Identity Authentication*

User identity authentication is a basic security function which is necessary in secure operating system. Taking the higher security into account, we implement Double-Factor user identity authentication mechanism. Double-factor authentication requires that the system user must provide two factors in order to successfully login into the security system, it is more secure than only depending on username and password.

One authentication factor is identity authentication based on U-Key. First, the user must hold correct U-Key and knows correct U-Key password, only if the user insert correct U-Key and input correct U-Key can he login the system successfully. Based on the U-Key driver module, we implement a user identity authentication module, before the secure system booting, the module is inserted by Initrd. The other authentication factor is based on "Username", "Passwd", "U-Key name", and "U-Key level" which are assigned to user by using configuration and management tool. In implementation, we use PAM mechanism. PAM is "Pluggable Authentication Module" for short. It is a center mechanism providing the authentication to all the service and it almost suits to all system applications, such as login, su and so on.

System administrator can modify PAM configuration file to set down different authentication policy.

Application program developer can use PAM API to call authentication method. PAM interface library reads configuration file so as to relate application program to authentication method.

When using configuration and management tool to add user, the system will bind U-Key name and U-Key level together, then creates the os210_userlist file. By adopting PAM mechanism and implementing a pamlogin.so dynamic library for login. When the user tries to login the system, the system compares the U-Key name and U-Key level that read from U-Key with the corresponding information in os210_userlist, if the same, then user logins successfully. The original authentication mechanism of Linux operation system based on Username and Passwd is still available. In implementation, only allowing 0 level user to login as "root". We modify /etc/pam.d/login file in system and add a line below so as to bring PAM module into effect:

```
auth required  /lib/security/pamlogin.so.
```

C. Implementation of Transparent Encrypting and Decrypting

Transparent encrypting and decrypting mechanism is an effective method to protect the sensitive data in the level of operating system. There are two kinds of key stored in each U-Key, one is working key, the other is private key. The U-Keys in the same working group (assigned in the distributing key) have the same working key, but have different private key. Working key is corresponding to public encrypted dir, while private key is corresponding to private encrypted dir. In the security system, data stored in public encrypted dir is in the manner of cryptograph actually, but for group user who has correct working key, "see " the data in public encrypted dir is "clear text", while for the user who has correct private key ,"see " the data in private encrypted dir is "clear text", but for the other users who has not correct private key, "see " the data in private encrypted dir is "cryptograph". Because the encrypting and decrypting mechanism is implemented in operating system kernel level, and it is "transparent" to user, so we call it transparent encrypting and decrypting mechanism.

In Linux operating system, the operation of "read" or "write" has cache. Usually it puts the file cache into the "inode" data structure of the file. There is a pointer in data structure "inode" which points to the data structure "address_space". "Cache queue" is in the data structure "address_space" in the manner of "memory page". In the data structure "address_space", queue head "pages" is used to maintenance "cache page queue", while the pointer "a_ops" points to the data structure "address_space_operations", which gives the concrete operation between cache page and file system, for example, "read" ,"write" and "lseek" cache pages from concrete file system. For Linux file system EXT2，the corresponding data structure is "ext2_aops"：

```
struct address_space_operations ext2_aops= {
readpage: ext2_readpage;
readpages: ext2_readpages;
prepare_write: ext2_prepare_write;
commit_write: ext2_generic_commit_write;}
```

The function of "readpage" is to read single page from cache pages, "readpages" is to read multi-pages from cache pages, "prepare _write" is to prepare "writing" for a certain cache page; After prepare_write finishes preparing for "writing", then there is a open road from cache to record in device. After writing to cache, it must hand those cache pages to kernel thread kflushd. While the above task is finished by the function of "commit write".

So as to implement the function of transparent encrypting and decrypting, we redirect the function in data structure "ext2_aops". For the file which has encrypting attribute, before using readpage() or readpages() to read file, it must decrypt first; and before using prepare_write()and commit_write(), it must encrypt first. To implement the above function, we implement redirection function: os210_cip_inode_redirectop ():

```
os210_cip_inode_readirectop {
new_a_ops->readpage=os210_cip_readpage;
new_a_ops->readpages=os210_cip_readpages;
new_a_ops->prepare_write=os210_cip_prepare_write;
new_a_ops->commit_write=os210_cip_commit_write; }
```

The whole transparent encrypting and decrypting function is based on LSM, the kernel object that needs to add security attribute is inode, super_block. In order to set security attribute and redirect "read" and "write", we use and rewrite some hook functions.

D. Implementation of Mandatory Access Control for Executable Program

The discretional access control mechanism of Linux operating system itself is not enough to protect the security of executable program. Though the super user can manage and use the system conveniently, it will bring much menace and hidden trouble to system. Moreover, complicated network attacks can success by using executable program and attack script. So the programs that are only granted can be allowed to execute in secure system. In implementation, user can be divided into two level: 0 level is administrator and 1 level is normal user; And executable program can also be divided into two level: 0 level is privileged executable program and 1 level is normal executable program. Furthermore, each user has an "access control number" and each executable program has a "access control string". Executable program mandatory access control policy is :0 level user can execute 0 level and 1 level executable program, 1 level user can not execute 0 level executable program, and can execute 1 level executable program whose "access control num" meets "access control string" of executable program.

Linux_binprm is a important data structure in Linux operating, it records parameter and environment variable before the new program loading. LSM provides hook function bprm_check_security (struct linux_binprm * bprm) to check executable program right before it executes. We rewrite the hook function and implement the mandatory access control policy.

E. Performance Results

In the implementation of TGrub, we use SHA-1 to measure the integrity of important component of trusted booting including Grub(of various stages), OS-Initrd and OS-Kernel, at the same time, use U-key to finish integrity recovery, thus it would lead performance overhand. If we ignore the performance overhand from integrity recovery, then the performance overhand of trusted booting can be computed as:

$$\Delta t = \sum t(S_i), (i = 1, 2, 3, 4, 5) \quad (1),$$

$t(S_i)$ is the time overhand to validate various stage of Grub, Os-initrd and Os-kernel. Our system configuration is P4 1.8G, memory is 256M, and Fedora4.0 operating system is running on it, boot partition is ext2 file system, and graphical interfaces is not chosen. The whole booting time (including measurement and validation) is about 4s and the integrity measurement and validation time is about 67ms. The table 2 below shows that concrete time performance overhead of each component

The performance overhead resulting from trusted booting is about 1.7 % , and so it almost can be negligible. In paper [13], Chris Wright gives the overhead about the system using LSM framework, the operation of "read" or "write" with or without LSM leads almost nothing overhead, so in the implementation of transparent encrypting and decrypting mechanism, the main overhead is as the result of encrypting and decrypting algorithm.

Table 1
THE TIME OVERHEAD OF INTEGRITY MEASUREMENT AND VALICATION

Component Name	Time
1.start.S	**2000us**
2.stage1.5 residual part	**4999us**
3.stage2	**3185us**
4./boot/initrd\-2.6.11\-1.1369_FC4.img	**24392us**
5./boot/vmlinuz\-2.6.11\-1.1369_FC4	**32961us**

V. Relative Work

Much work has been done to enhance the security of Linux operating system, though different projects adopt different implementation methods and implement different security function, access control is a main security function, and the main method is kernel patch and LKM (Loadable Kernel Module).

SeLinux[16][17] is a security-enhanced operating system which is sustained and participated by NSA, Co. Mitre, NAI lab and the Research Group of Kernel of Utah University. The objective of SeLinux is to provide a basic support to MAC and by flexibly configuring access control policy to achieve security requirements and security objective, and then validate the feasibility of the application of MAC to mainstream operating system. The main security function implemented in SeLinux based on LSM [13] framework includes RBAC (Role Based Access Control), TE (Type Enforcement) and MLS (Multi-Level Security).

RSBAC [15]is a security-enhanced operating system developed by Hamburger University of Germany which uses GFAC (Generalized Framework for Access Control) which is a classical multi security policy supporting Framework to manage and organize the access control policy, and at the same time, uses the LKM mechanism to support the access control policy to be extended dynamically. If the developer adds new security policy to system, it only needs to implement corresponding policy module according to interface specification, then inserts the module into the system and registers it, thus the security function can enforce. RSBAC supports classical security policy such as BLP [7], Biba [8] model. Besides, it has the function of intrusion detection and can scan virus in Linux operating system.

TrustedBSD [14] is a security-enhanced operating system developed by FreeBSD Group according to Common Criteria based on FreeBSD operating system. TrustedBSD is based on the uniform supporting framework of mandatory access control policy and can support the security policy module developed by third party, such as security auditing, intrusion detection, and the management of system by various privilege.

The above work is innovative and significant, so we adopt as the reference to our work. The same as SeLinux, our system is based on LSM, so it has the characteristic of flexibility, high-efficiency and easy-development. Though SeLinux has implemented various security functions, the configuration and management of security policy is complex, so it is not easy for normal user. But our system use graphics interface to finish configuring and managing, it is easy using. Besides, those studies differ from ours in that our proposed architecture is based on the analysis of the relation between "secure" and "trusted", by leveraging trusted hardware: U-key, it thus provides high assurance for the enforcement of security policy.

Another line of work focuses on using trusted hardware: such as TPM (Trusted Platform Module) to enhance the security of system. BIND [11] enhances those studies by providing a fine-grained attestation service for a process by isolating only some "important" sections of a process, instead of the whole application. As each application has to identify where to call the BIND interfaces in a program's code, it is not a transparent approach for legacy applications. Also, it is difficult for this approach to support collaborations where communication and information sharing are required during the running of an isolated section. SecureBus [18] is a trusted computing architecture， it is based on trusted hardware technology such as LT Technology presented by Intel or SEM technology presented by AMD. The architecture regards TPM as trusted root and in turn passes the trust to Secure Kernel and then to SecureBus, thus assures that the running of SecureBus is trusted. EMSCB [12] is European Multilateral Secure Computing Base for short, its main objective is to provide open-resource trusted computing platform and can solve some security question.

The above architectures also give some lights to our design. SecureBus and EMSCB uses TPM as trusted root, while our system uses trusted hardware: U-Key as trusted root, the difference is “start” of trusted chain. Nowadays, though the price of TPM chip is cheap, most of PCs have not been equipped with TPM, so ours is an effective resolution in a way, in some sense, U-Key can be considered as a mobile TPM.

VI. Conclusion

The rapid development of network technology facilitates the intercommunion of information and the share of resource. But the risk that the computer is attacked is increasing evidently. As the software base of information security, so people pay more attention to the security enhanced techniques for operating systems, and some security-enhanced technology and system is emerging. Based on the deep analysis of the relation between “security” and “trusted”, this paper enhances these studies by providing a trusted hardware: U-Key to the secure terminal system. It gives the description of the architecture design, and discusses the concrete implementation. Furthermore, the performance overhead of system is also analyzed. The paper gives illumination on how to use security hardware to enhance the terminal system security.

Acknowledgment

This paper is supported by The State Key Laboratory of Information Security of Graduate School of Chinese Academy of Science and The Trusted Computing Laboratory of Beijing University of Technology. This work is supported by the National High Technology Research and Development Program (863 Program) under grant No. 2006AA01Z440 and the National Base Grand Research fund (973 Program) grant No. 2007CB311100.

References

[1] Jason F. Reid William J. Caelli DRM, Trusted Computing and Operating System Architecture Australasian Information Security Workshop 2005 (AISW2005), Newcastle, Australia. Conferences in Research and Practice in Information Technology, Vol. 44. P. Montague, R. Safavi-Naini, Eds.

[2] TCG Specification Architecture Overview Specification Revision 1.2 28 April 2004

[3] International Standard ISO /IEC 15408 2001

[4] Gary Stoneburner Underlying Technical Models for Information Technology Security NIST Special Publication 800-33

[5] T. Jaeger, R. Sailer, U. Shankar. PRIMA: Policy-Reduced Integrity Measurement Architecture. Proceedings of the 11th ACM Symposium on Access Control Models and Technologies (SACMAT 2006), 2006.

[6] Umesh Shankar, Trent Jaeger, and Reiner Sailer. Toward automated information-flow integrity for security-critical applications. In Proceedings of the 13th Annual Network and Distributed Systems Security Symposium. Internet Society, 2006.

[7] David E. Bell and Leonard J. LaPadula. Secure Computer System: Unified Exposition and MULTICS Interpretation. MTR-2997 Rev. 1, The MITRE Corporation, Bedford, MA, USA, Mar 1976.

[8] K.J.Biba. Integrity considerations for secure computer systems. Technical Report MTR 3153, The Mitre Corporation, April 1977

[9] R.Sandhu and X.Zh Peer to Peer Access Control Architecture Using Trusted Computing Technology SACMAT’05, June 1–3, 2005, Stockholm, Sweden.

[10] W.Arbaugh, D.Farber, J.Smith. A Secure and Reliable Bootstrap Architecture. IEEE Symposium on Security and Privacy [C]. Oakland: IEEE Computer Society Press, 1999,pages 65-71.

[11] E. Shi, A. Perrig, and L. Van Doorn. Bind: a fine-grained attestation service for secure distributed systems. In Proceeedings of IEEE Symposium on Security and Privacy, pages 154–168, Oakland, CA, USA, May 8-11 2005.

[12] Ahmad-Reza Sadeghi, Christian Stüble, Norbert Pohlmann European Multilateral Secure Computing Base http://www.emscb.de/content/pages/Einleitung.htm

[13] Chris Wright, Crispin Cowan, Stephen Smalley, James Morris, and Greg Kroah-Hartman. Linux Security Modules: General Security Support for the Linux Kernel.11th USENIX Security Symposium, San Francisco, CA, August 2002.

[14] TrustedBSD project. http://www.trustedbsd.org/.

[15] Ott A. The rule set based access control (RSBAC) Linux kernel security extension http://www.rsbac.org/documentation,2001.

[16] P. Loscocco, S. Smalley Integrating flexible support for security policies into the Linux operating system1 FREENIX Track : 2001 USENIX Annual Technical Conf1 (FREENIX’ 01) , Berkeley , CA , 2001

[17] T. Jaeger, R. Sailer, XL. Zhang Analyzing integrity protection in the SELinux example policy The 12th USENIX Security Symposium , Washington , DC , USA , 2003

[18] X. Zh, SQ Ch, Michael J. Covington, and Ravi Sandhu SecureBus: Towards Application-Transparent Trusted Computing with Mandatory Access Control In Proceedings of ASIACCS'2007. pp.117~126

Wei-Peng Liu: male, born in 1980, he received B.E. degree in Guangzhou Communication Institute of PLA in 2002 and M.E. degree in PLA University of Technology in 2005 in Nan Jing. He is currently pursuing PH.D degree in State Key Laboratory of Information Security of Graduate School of Chinese Academy of Science. His main research interests include security operating system, security model and trusted computing key technology and theory.

Jun Hu: male, born in 1972, teaching assistant of BeiJing University of Technology. He received B.E. degree in China University of Science and Technology in 1994, M.E. degree in of State Key Laboratory of Information Security of Graduate School of Chinese Academy of Science in 1997, Ph.D degree in 2008 from State Key Laboratory of Information Security of Graduate School of Chinese Academy of Science in 2008. His main research interests include the key technology and theory of security operating system.

Xing Zhang: male, born in 1966, senior engineer of Beijing University of Technology. He received B.E. degree in Electronic Technology Institute of PLA Information Engineering University in 1988, M.E. degree in Electronic Technology Institute of PLA Information Engineering University in 1996. Now he is engaged in PH.D degree in Electronic Technology Institute of PLA Information Engineering University .His main research interests include trusted computing key technology and theory.