International Journal of Security and Its Applications
Vol.8, No.5 (2014), pp.265-276
http://dx.doi.org/10.14257/ijsia.2014.8.5.24

A Method to Construct Dual-Kernel Trusted Computing Environment on Embedded System

Kong Xiangying[1,2] Chen Yanhui[1] and Chen Xuebing[1]

[1]Jiangsu Automation Research Institute, Lianyungang China
[2]College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China
[1]Kongxy716@aliyun.com

Abstract

Currently, with the increasing number of embedded applications, the research for the security of embedded systems has become popular. Considering the requirement of integrity and security, the characteristics and the design constraints of embedded systems, a method of constructing software trusted root and a trunk-branch trusted chain transfer model is proposed. The software trusted root is composed of boot loader and reliable kernel. SHA-1 engine which is included in boot loader (burned in boot flash), measures and loads the trusted kernel and protect the boot loader by prohibiting kernel and user-level application to write to flash. SHA-1 and trusted kernel together can be used as a trusted root to withstand non-physical attacks. Trusted kernel contains virtual trusted platform module (vTPM) module which can provide cryptographic functions and related services to the user kernel and guides flash to open up specific memory for platform configuration register (PCR) of vTPM. Application is running as a process of the trusted kernel. In the trunk-branch trusted chain transfer model, boot loader verify the trusted kernel, trusted kernel authenticate users kernel, and kernel users verify the application, thus the trust extents to the application layer. The proposed method not only avoids the trust attenuation problem which may occur in chain transfer, but also raises the low efficiency caused by using only ETPM to measure reliability in the star model. Finally, a prototype system is given, and the test results show that this method has built a trusted computing environment for embedded applications on existing hardware and software resources without additional hardware.

Keywords: *trusted computing, embedded system, dual-kernel, trusted root, trunk-branch model*

1. Background

In recent years, the rapid development of electronic and information technology leads the widespread use of embedded technology and products in aerospace, defense equipment, industrial control, medical equipment, consumer electronics and other fields. In the meanwhile security is become increasingly important. Unreliable devices may disclosure personal privacy, or even cause significant economic harm to national defense, endanger human beings. The malware threats report of the third quarter by 2013 of McAfee announce that: the quarterly growth of new personal computer (PC) malware samples is relatively stable, but attacks on the Android mobile operating system grew by more than 30%. Android malware samples library increased to 700,000, and the total number of samples is up to 2.8 million [1]. Embedded systems become new security offensive and defensive positions. So how to provide a trusted computing environment for embedded applications attracts many researchers.

On the one hand, similar to desktop system, it is difficult to fundamentally relieve security threats faced by embedded systems by traditional means of security firewalls and antivirus software. Trusted Computing Group (TCG) proposed an effective solution to the aforementioned problem -- to establish a trusted root in computer system, and ensure the integrity and security of trusted computing systems by the idea of trusted chain [2]. On the other hand, embedded systems have their own characteristics, and face security threats and design constraints other than desktop systems, so building embedded trusted computing environment must consider the following factors:

a) Each embedded system is customized with multi-peripheral, and its system memory address space, I/O address space, and system resources such as interruptions are different from each other. The configuration of these resources is only known by the designer and needs specific boot loader. This differs from the versatility required by a PC which BIOS supports dynamic expansion.
b) Embedded systems are typically designed for a particular type of application, the hardware and software of the system are fixed, especially those used for control or monitoring. And embedded systems are designed to need, so powerful functions on desktop operating system are not necessary.
c) Compare to PC, embedded systems have very stringent requirements on volume, cost, power consumption and reliable. Add more chips in a system means more cost, power consumption, size, and less reliability, thereby reducing competitiveness.

According to the aforementioned characteristics of embedded systems, drawing on ideas of trusted computing. In this paper a dual kernel embedded trusted computing environment is presented. Take the characteristics of embedded systems into consideration, and based on existing hardware like flash, CPU, etc. without adding additional hardware, the presented environment uses the boot loader of embedded vTPM together with the kernel as trusted root. The user applications and the kernel are running as a process of the trusted kernel. The trunk-branch trusted chain transfer model extended the trust from the boot loader to the application layer.

Subsequent parts of the paper is organized as follows -- the second part describes the current research results, the third part gives the architecture of the dual-kernel embedded trusted computing environment, the fourth part shows a prototype system , and the fifth part gives the test results and analysis, and finally made the outlook for future study.

2. Related Research

Existing embedded trusted computing model mainly rely on the trusted embedded hardware. According to the number of processor chips the model can be divided into two categories, one is multi-chip architecture; the other is single-chip architecture.

Multi-chip architecture solve the problem that the main processor does not have the authorization and authentication function by adding a trusted platform module to identify the system and protect information. The trusted computing model given by TCG is a multi-chip architecture. For mobile platforms, TCG gives mobile platforms reference and several mobile trusted computing module specifications, which can be used by mobile chips and mobile phone manufacturers as well as mobile application developers [3, 4]. To enhance the CPU scheduling and fulfill the demand for higher system reliability in embedded systems, Huanguo Zhang from Wuhan University has designed an embedded TPM (ETPM) which adds bus arbitration module, symmetric cryptography engine and backup recovery module in the original TPM. ETPM, which is based on a star measurement model and can reduce the loss of trust in the transfer process [5], is also multi-chip architecture. Based on TPM or ETPM, scholars have conducted many researches and build some embedded trusted computing environment [6-10].

Single-chip architecture trusted solution refers to integrating trusted module into the main processor, without additional chips. ARM's TrustZone [11-13], MIT's security processor AEGIS [14], Stanford University’s XOM (execute only memory) [15] are using single-chip architecture. TrustZone applies AMBA AXI system bus and specific IP for system expansion, and its security features are built inside of the processor, which can effectively protect the memory, the encryption block, the keyboard and screen and other peripherals from software attack. TrustZone core CPU provides a special level of security—trust zone, which allow kernel mode and user mode program to run, and also provides an independent module to monitor switch between trust and non-trust areas. Monitoring module is a small, non-reentry program that protects content in context switch, monitors real-time operations of processors. The system can only use limited commands on monitoring module. Using TrustZone, Johannes Winter from Austria Institute of information processing and communication has to implement an embedded trusted computing environment on Linux [16]. AEGIS is a single-chip security processor designed by the MIT Artificial Intelligence Laboratory. By applying a random physical number generation function and off-chip memory protection mechanisms, AEGIS divided the system boot into several layers. In the transfer of each layer, the integrity is verified. And AEGIS enhances the robustness of the system by providing integrity restore mechanism [14].

The aforementioned studies are based on hardware to guarantee the reliability of computing environment. Multi-chip architecture requires additional reliable chip (TCM, TPM or ETPM), which leads the increase of volume; power consumption and cost thus reduce the competitiveness of the product. Single-chip architecture is suitable only for a particular chip, rather than most of the embedded chip in the current market.

3. Dual kernel Embedded Trusted Computing Environment

Trusted computing environment based on dual-kernel (ETCEDK) system is divided into three layers, and the architecture is shown in Figure 1. The bottom -- trusted root is combined by trusted boot loader (TBL) with SHA-1 engine and trusted kernel (TK) with vTPM; the intermediate layer is embedded TPM software stack (TSS) of the user's operating system kernel, the top is application layer.

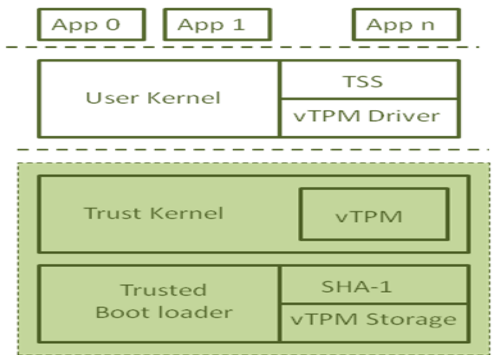

Figure 1. Architecture of Dual Kernel Embedded Trusted Computing Environment

3.1. Trusted Root

The core idea of trusted computing is to introduce TCM（Trusted Computing Module）as a trusted root to protect the vulnerable hardware and software (BIOS, operating system, memory, *etc.,*). Base on TCM to establish a chain of trust in the order of the first measure and then load the module. By this way trust is transmitted to each module, thus establish the trust of the system.

The definition of a trusted root is given as follows:

Definition 1 Trusted root is a reliable component of a computer system which meets the following criteria:

1) It should be able to withstand the non-physical attack;

2) It should be able to provide cryptography and certificate services to other components in the system;

3) It should be capable of extending trust to other components in the system.

In PC, BIOS is the first code been executed after the system start up. To ensure the openness of the PC, BIOS not only allow expansion through firmware according to certain specifications, but also provide the appropriate interfaces that allow applications to access and modify the upper configuration and even replace its original version. Take display BIOS and network BIOS as examples, during loading BIOS initiate the graphics card and network card, or even possible to boot system from the network according to BIOS settings. However, the openness reduces the credibility, thus cannot meet the first requirement of the trusted root, so the TPM chip needs to be introduced as a trusted root to resist non-physical attack.

For embedded systems, devices are usually designed for special applications. The BIOS is usually the boot loader which is burned in the corresponding flash. Boot loader is used to establish runtime environment, initialize the necessary hardware, memory and CPU, and provide the basic environment to load and run the operating system. Usually boot loader of ARM embedded devices is vv or uboot; MIPS devices are PMON; PowerPC devices are uboot. The open source of vv, uboot, and PMON includes support for most of the development board, and provides abundant of device drivers, powerful configuration and debugging functions. Users can modify these boot loader based on the actual configuration of the hardware device and remove useless codes. Therefore, by cutting, boot loader can prohibit itself from been modified by specific peripheral devices.

To improve the Linux real-time performance, layered architecture dual-kernel is proposed by Victor Yodaiken and Michael Barabanov from Department of Computer Science of New Mexico University. Dual-kernel is to add a simple real-time kernel under the standard Linux kernel. The original non-real-time Linux kernel runs on top of this real-time kernel as a preemptive task. The access to the hardware is provided by the underlying kernel [17, 18]. Based on such idea, authors have implemented a dynamic credible proof embedded system. Since all kernel-level hardware access are through the underlying kernel system call, based on dual kernel architecture the underlying kernel can prohibit access to specific hardware, including the boot flash direct write access operation, thus eliminating possible malicious changes in the boot flash applications of upper kernel.

Based on the considerations mentioned above, we have modified the boot loader, and made full use of existing hardware and software resources to build a trusted computing environment for embedded applications.

ETCEDK trusted root contains two parts, TBL and TK, the interaction between the two modules ensures trusted root cannot be tampered.

Guide flash is divided into two parts -- the lower part stores the binary code of TBL, and the higher part is used as a non-volatile memory to store PCR and other constants, like endorsement key (EK) which is assigned uniquely to the embedded device and pre-computed hash value TK SHA-1.

TBL is a boot loader including SHA-1 algorithm stored in the boot flash. It mainly complete the board-level hardware initialization and clear the PCR register according to specification TCG1.4, and use the SHA-1 algorithm to measure the TK module, namely calculate TK hash value. Then it compares the measured value with the stored constant of TK. If the two values are consistent, TBL loads TK and gives the CPU control to TK, otherwise it will alarm and stop the execution. The use of PCR is in line with specification TCG1.4. See in Table 1.

Table 1. Definition of PCR

PCR Number	Usage	Reset Value
0	TBL	0
1	Hardware configuration	0
2	TK	0
6	Status	0
8-23	UK kernel usage	

TK includes vTPM, operating system kernel for memory management, interrupt management, I/O management and process management. User Kernel (UK) is running as a process of TK. According to the size of TK binary module and capacity of the boot flash, TK can be stored in the boot flash or external memory. Note that in order to ensure the integrity of TK, we compile TK into an image file.

vTPM is a software simulator of the TPM, which has essential functions of the TPM -- symmetric / asymmetric encryption, integrity metrics, secure storage and signature authentication. vTPM structure is shown in Figure 2. The left box is the required data area of vTPM which is stored in flash.

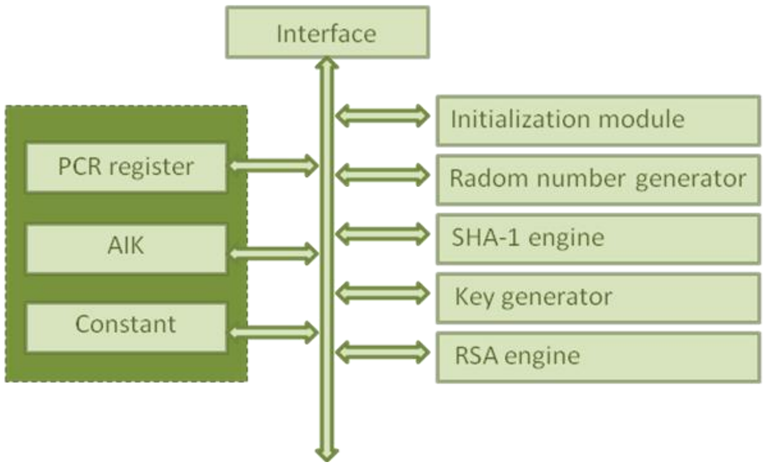

Figure 2. Structure of vTPM

Following paragraph proves that the trusted root constructed above meets definition 1.

Theorem 1: TBL and TK together constitute a trusted root.

Proof: Since the UK is running as a process of TK, all its access to hardware resources must call the system call of TK. It is designed to shield modification to flash from UK and upper applications. On the other hand, which is already mentioned before, the measured value and the stored constant of TK will be compared. If two values are consistent, load TK and give the CPU control to TK, otherwise start alarm and stop the execution. These two methods interact with each other can withstand software attacks and ensure the integrity of the trusted root, thus meets the first requirement of the definition 1.

In the meanwhile, TK has cryptography and certificate services, and can pass the trust to the upper layer (see section 3.2). Thus meet the second and third requirements definition 1.

Therefore, all the requirements of definition 1 are satisfied.

3.2. User kernel

User kernel is a common operating system kernel, which is running as a process of trusted kernel and provides trusted service to applications by TSS. The structure is shown in Figure 3.

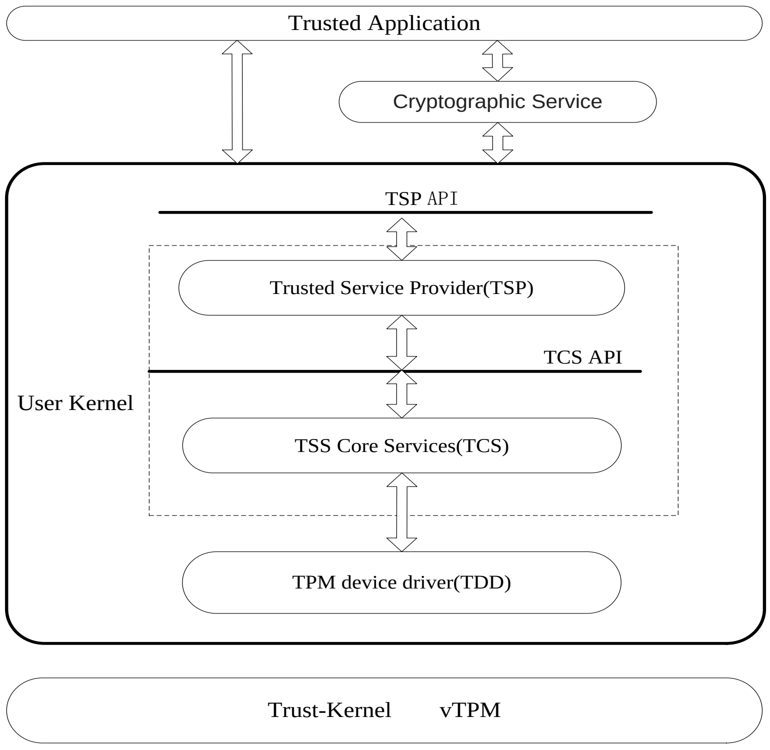

Figure 3. Structure of UK

TSS lies between vTPM and application software, which provides interface to applications to access vTPM, and contains vTPM management functions. TSS is divided into three layers: trusted service provider (TSP), TSS core services (TCS) and kernel mode TPM device driver (TDD).

TSP provides application programming interface to TPM, application context management and cryptographic operations services. It runs as a user process resides in user space. TCS provides a set of standard API for TSP. A TCS can serve multiple TSP, including four core services: thread context management; storage and platforms relevant certificates and keys management; event logs and access the corresponding PCR register measurement management; TPM command serialization, synchronization and parameter block production. TDD is a device driver, which enables access to vTPM.

3.3. Trust Chain Transfer Process

System trust chain transfer process is shown in Figure 4. After the system is powered on, TBL first get control of the system and complete the relevant initialization. Then get TK from flash or external memory, use SHA-1 algorithm to measure the integrity, and compare the value with the expected value stored in flash. If two values not equal, start alarm and stop execution, otherwise, pass the trust to TK. TK get executed right, initialize the trust the root operating environment, and start vTPM to prepare the execution environment. Then follow the UK's start loading order, TK measure related modules from the external memory and extend the results to the PCR, and complete loading UK. Afterwards trust boundaries are extended to UK. Finally UK metrics each application, extended the result to the PCR, and then starts the application. By the way the trusted boundary will be extended to the entire system.

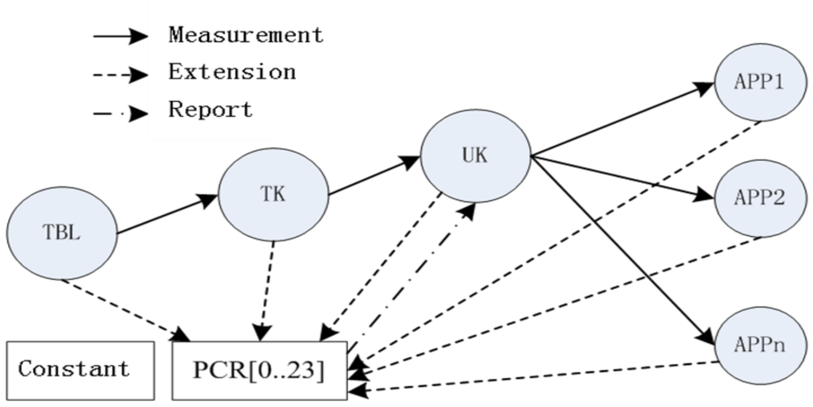

Figure 4. Trunk-branch Trusted Transfer Model

The trunk-branch trusted extension model uses a combination of chain and star model and gets advantage of both. Transfer between TK and TBL use the chain structure, and is called a trunk. Transfer between UK and applications uses a star model, and is called a branch. The problem of chain model is the gradual decay of trust, while star model avoids the decay by using ETPM to measure all components in the system, but take very long time because performance of ETPM processor is usually low. In trunk - branch model, TK is measured by TBL, UK by TK and application by UK. All the calculations are completed by CPU, so greatly reducing the measurement time.

4. Prototype Implementation

Lab environment is built on Godson 2F, Linux and Qemu virtual machine. Godson 2F is MIPS architecture, clocked at 600MHZ, with ATI M9 graphics controller, Intel82257 network controllers, 64G ATA hard drive, and PMON as the system bootloader. PMON takes charge of CPU operating environment preparation, board-level and PCI bus initialization, peripherals detection, and loads the operating system from the external memory and CPU and give control to the operating system. PMON here is cut, leaving only the device drivers ATA hard drive, while increasing the SHA-1 algorithm engine.

Flash boot using a 1MKB Norflash SST49LF008a chip, which is used to store PMON binary code and related environmental variables (configuration data) in lower 768K, higher 256K store PCR, the key and the SHA-1 hash TK value. As shown in Figure 5.

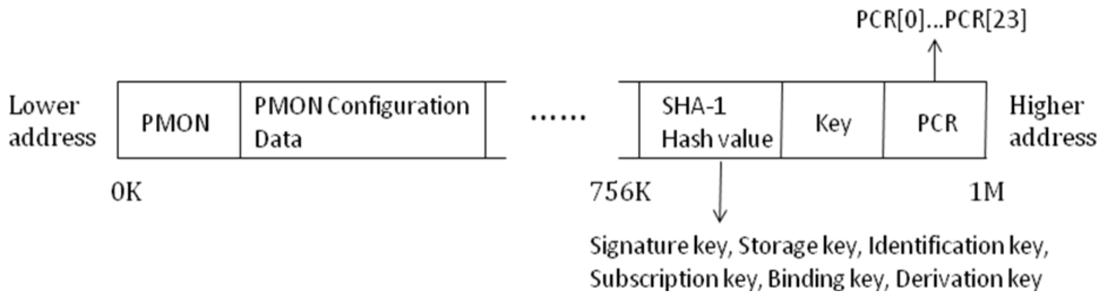

Figure 5. Flash Storage Allocation

RTLinux is not chosen when design the trusted kernel, because it supports the real-time kernel by dynamic loading, interruption, and related modules scheduling, but it is difficult to compile the real-time kernel and non-real-time kernel into two independent images. Therefore, virtual machine is used to verify the proposed construct method.

TK uses Linux2.6.x with improved vTPM. These two together are compiled into a binary image. TK will load and start a Qemu virtual machine, and build a virtual platform for the UK kernel.

To extent the trust on virtual platform, Stefan Berger from IBM proposed to use hardware virtualization, in order to make TPM to support multiple virtual machine environment. Mario Strasser from Switzerland has developed a software simulator of TPM, it supports for UNIX platforms and can be used to test and related trust construction methods [20]. This TPM simulator consists of three parts: The TPM Emulator Daemon (TED), TPM device driver Library (TDDL) and compatible TPM device driver devices (TDD Dev) [20]. TK only uses TED. TDDL and TDD Dev are on UK, in order to provide TED interface to TSS. We conducted a simple modification TED to map PCR and its associated key to the corresponding guide flash address space.

Qemu is a cross-platform virtual machine developed by Fabrice Bellard's [21, 22]. Its performance is almost close to the actual hardware environment by using kqemu accelerator [23]. In Qemu, TED is virtualized into a TPM character device and can be enumerated after UK started. TED is virtualized by PCI devices, can be simulated on the TPM chip. To add vTPM.c file in Qemu the hw directory, the step is as follows:

1) Define the management structure vTPM.

```
Static PCIDeviceInfo vtpm_info={
      .qdev.name=“ vTPM”,  //Device name
      .qdev.desc=“ vTPM for qemu”,//Device description
      .qdev.size=sizeof(TPM State), //Store status of TPM
      .init=vTPM_init,   //Definition of initial function
}
Typedef struct TPMState_t{
   //pci device information
   PCIDevice dev;
     //TPM status
   Unsigned char status;
     //package
  Unsigned char buf[TPM_MAX_DATA];
} TPMState;
```

vTPM_init function will be called when the QEMU started. vTPM_init is used to initialize the vTPM properties, including the setting of PCI vendor number and device number, size of PCI BAR and port mapping function vtmp_map etc., and registered the read/write function of TPM State in vtmp_map.

Examples are as followed:

```
/ / Call the pci-regiser_bar in qemu to register PCI IO space port mapping function
pci-regiser_bar (dev, 0, SIZE, PCI_BAR_ADDRESS_SPACE_IO, vtmp_map);
/ / Call the register_ioport_write of qemu in vtmp_map function to register the write
/ / function of TPMState, call the register_ioport_read of qemu to register the read
/ / function of TPMState.
TPMState  *s;
register_ioport_write(start_addr,length,size,vtpm_state_write,s);
register_ioport_read(start_addr,length,size,vtpm_state_read,s);
```

2) Since vTPM of PCR register is stored in flash, read PCR will be directly mapped to the read physical address of flash.

Implementation of the structure management registration functions of vTPM;

```
static void vTPM_register_devices(void)
{
pci_qdev_register(&vtpm_info);
}
```

(3) Add the vTPM to QEMU device list;

```
device_init (vTPM_register_devices);
```

(4) Mount vTPM to pci bridge;

```
Pci_create_simple (pci_bus,-1,"vTPM);
```

Trusted stack of UK has use TrouSerS [24], and a reliable service platform -- OpenPTS[25]. System block diagram is shown in Figure 6.

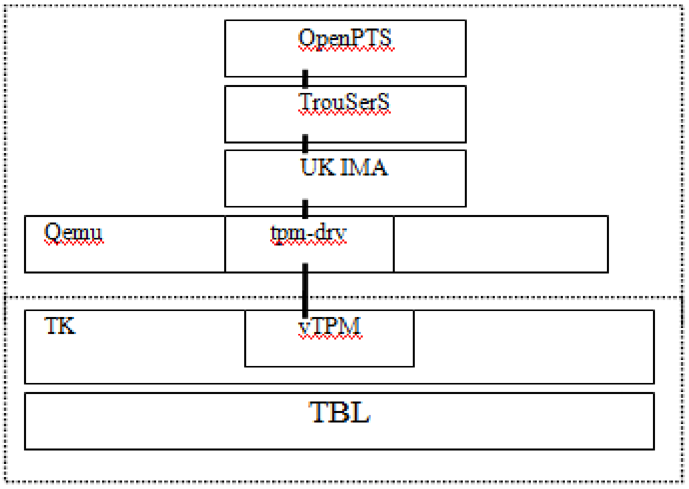

Figure 6. Structure of Prototype

In order to optimize the system startup time, we have improved the algorithm of loading the Linux kernel. Before loading modules, the system firstly read module code into memory, and measure it with SHA-1 (A), then the improved algorithm load the directly from memory, thus re-read the disk and shorten the system start-up time.

5. Performance Measurement and Analysis

Based on the above prototype system, we tested the system start-up time, system measurement time consuming, and application execution efficiency. The size of each module is shown in Table 2, and system start-up time in Table 3.

Table 2. Size of Each Module in Prototype System

Number	Module	Size of binary code（KB）
1	TBL（Pmon）	379.332
2	TK	972.076
3	Qemu	7323.292
4	UK	12707.686
5	App	4572.330

Due to the usage of dual-core architecture, TK and Qemu are introduced, and the size increases 8295.368KB.

Table 3. Start-up Time for Each Module in Prototype System

Number	Operation	Timing（ms）	Time consuming（ms）
1	TBL start	0	4133
2	TK read	4133	239
3	TK measure	4372	1667
4	TK start	6039	3066
5	Qemu read	9105	1799
6	Qemu measure	10904	12572
7	Qemu start	23476	2820
8	UK read	26296	3124
9	UK measure	29420	21817
10	UK start	51237	6071
11	App read	57308	1123
12	App measure	58431	7843
13	App start	66274	

From power on to the first application begins spends 66.274 seconds, during which the measure hash value takes 43.899 seconds, an average measurement time is 0.246s/MB, and due to the introduction of TK takes 22.163 seconds. In another platform which using ETPM (33MHz processor of choice is a C * Core) to achieve trusted computing, only one measure of a 3.6MB kernel modules takes about 15.6s, measure the rate is about 4.33s/MB. The proposed solution is nearly 20 times faster because our program uses a higher frequency CPU. In addition, as we have improved the module loading algorithm, duplication read the disk by measure and load operation is avoided which also shortens system start-up time.

Kernel and user applications are running on Qemu virtual machine. We selected three types of applications for efficiency test: fixed float calculation time, I/O performance of network communication, and average polygon filling time, the test results are shown in Table 4.

Table 4. Prototype System Performance Test Results

Subject	Real environment	Qemu virtual environment	performance
Float calculation	3966ms	4305ms	0.92
Network communication	10.23MB/s	8.71 MB/s	0.85
Polygon filling	0.124ms	0.224ms	0.55

From the test results, using Qemu virtual machine, the computing performance is approximately 92% of the original system, the communication performance is about 85%, and about 55% screen mapping capability of the original system. Because there is no hardware virtualization instruction support for Godson 2F, Qemu virtual graphics system does not support hardware accelerated graphics operations.

6. Conclusion

According to the reliable demand for embedded systems, this paper presents an embedded trusted environment model, and realized it. The main innovation of this paper is as follows:

1) Based on the characteristics of embedded systems, proposed pure software trusted root model and demonstrate its compliance with the definition of a trusted root, and meet the requirement for size, power and cost of embedded system.

2) The proposed trunk - branch trusted chain transfer model, which has advantages of both chain and star model, reduce delivery time while avoiding the loss of trust;

3) Improve the trusted system loading algorithm to avoid re-read the disk and reduce system loading time;

4) Build a prototype system to verify the validity of the model and make the embedded terminal trustful.

In future study, based on the prototype, we will extent the trust from terminal to the network to establish a trusted network models of embedded systems. On the other hand, we will modify RTLinux, so that it can be compiled into two separate modules, which will further reduce the size of the trusted root and the probability of flaw.

References

[1] “Mcafee Labs threat report”, the third quarter of 2013. http://www.mcafee.com/cn/resources/reports/rp-quarterly-threat-q3-2013-summary.pdf, **(2013).**

[2] “TCG”, TCG Specification Architecture Overview, Reversion 1.4, **(2007),** http://www.trustedcomputinggroup.org/files/resource_files/AC652DE1-1D09-3519 ADA026A0C05CFAC2/TCG_1_4_Architecture_ Overview.pdf.

[3] “TCG. TCG Mobile Trusted Module Specification Version 1.0”, **(2007),** http://www.trustedcomputinggroup.org/specs/mobilephone/tcg mobile trusted module1.0.pdf.

[4] “TCG”, TCG Mobile Reference Architecture Version 1.0, **(2007),** http://www.trustedcomputinggroup.org/specs/mobilephone/tcg mobile reference architecture 1.0.pdf.

[5] Z. Huanguo, L. Jing, P. Danling and Z. Bo, “Trusted Platform Module in Embedded System”, Journal of Computer Research and Development, vol. 48, no. 7, **(2011).**

[6] Z. Yu, H. Dake and H. Ming, “Trusted computing based user authentication for mobile equipment”, Chinese Journal of Computers, vol. 29, no. 8, **(2006).**

[7] C. Shuyi, W. Yingyou and Z. Hong, “Conceptual design of trusted mobile platform”, Journal of Northeastern University: Natural Science, vol. 129, no. 8, **(2008).**

[8] W. Yu, W. Zhenyu and Y. Lining, “Design and implementation of TPM extension and trusted bootstrap on embedded platform”, Computer Engineering and Design, vol. 30, no. 9, **(2009).**

[9] B. Zhao and H. Zhang, "The System Architecture and Security structure of Trusted PDA", Chinese Journal of Computers, vol. 33, no. 1, **(2010).**
[10] K. Xiangying, C. Xuebing and Z. Yi, "A Dynamic Trustworthiness Attestation Method based on Dual Kernel Architecture", International Journal of Hybrid Information Technology, vol. 6, no. 5, **(2013).**
[11] T. Alves and D. Felton, "Trust Zone: Integrated hardware and software security (enabling trusted computing in embedded systems)", Information Quarterly, vol. 3, no. 4, **(2004).**
[12] "ARM Limited", Designing with TrustZoneTM: Hardware Requirements. Product brief, **(2005)** August, http://www.arm.com/pdfs/TrustZone Hardware Requirements.pdf.
[13] http://www.arm.com/zh/products/processors/technologies/trustzone.php.
[14] http://people.csail.mit.edu/devadas/pubs/aegis-istr-august6-2005.pdf.
[15] http://theory.stanford.edu/~jcm/papers/xom-asplos.pdf.
[16] J. Winter, "Trusted Computing Building Blocks for Embedded Linux-based ARM Trust Zone Platforms", ACM STC'08 Proceedings of the 3rd ACM workshop on Scalable trusted computing, **(2008).**
[17] M. Barabanov, "RTLinux Thesis", New Mexico Tech., **(1996).**
[18] C. E. Hall, "A real -time Linux system for autonomous navigation and flight attitude control of an uninhabit edaerial vehicle", Proceedings of the 20th Digital Avionics Systems Conference Proceedings, Daytona Beach, **(2001)**, FL, USA.
[19] S. Berger, R. Caceres and K. A. Goldman, "vTPM: virtualizing the trusted platform module", Proceedings of the 15th USENIX Security Symposium (USENIX Security 2006), **(2006),** Canada.
[20] M. Strasser and H. Stamer, "A software-based trusted platform module emulator", Proceedings of the 1st International Conference on Trusted Computing and Trust in Information Technologies: Trusted Computing - Challenges and Applications (TRUST 2008), **(2008,)** Villach, Austria.
[21] https://www.usenix.org/legacy/event/usenix05/tech/freenix/full_papers/bellard/bellard.pdf.
[22] http://bellard.org/qemu.
[23] F. Bellard, "Qemu Accelerator Module", **(2006),** http://fabrice.bellard.free.fr/qemu/qemu-accel.html.
[24] "TrouSerS", http://trousers.sourceforge.net/ v0.3.6, The open-source TCG Software Stack.
[25] "openpts", http://sourceforge.jp/projects/openpts/wiki/FrontPage Open Platform Trust Services.

Authors

Kong Xiangying, born in 1972. He is a PhD candidate of Nanjing University of Aeronautics & Astronautics. He research interests include Software Engineering, Information Security, Real-Time Operating system, *et al.*,.