International Journal of Hybrid Information Technology
Vol.6, No.5 (2013), pp.237-248
http://dx.doi.org/10.14257/ijhit.2013.6.5.22

A Dynamic Trustworthiness Attestation Method based on Dual Kernel Architecture

Kong Xiangying[1,2], Chen Xuebing[2] and Zhuang Yi[1]

[1]*College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China*
[2]*Jiangsu Automation Research Institute, Lianyungang China*

Kongxy716@aliyun.com

Abstract

The existing trustworthiness attestation methods are not only difficult to be applied to the embedded system because they are mainly based on virtual machine technology, but have some problems such shat evidence is not obtained in time, protecting the privacy need trusted third party and trust measurement efficiency is low. In this paper, an embedded system dynamic trustworthiness attestation method based on dual-kernel (super kernel and normal kernel) operating system architecture is proposed. Super kernel is non-changeable, and it verifies the integrity of the critical data structures and kernel file in normal kernel. Super kernel can serve as a trusted third party which can dynamically verify whether the code segment changes in runtime. A system implementation is given in this paper, and the experimental data show that the behavior of the system can dynamically verify the behavior of program whether meets embedded trusted application demand or not.

Keywords: *Trusted Computing; Trustworthiness Attestation; Dual core; Dynamic Integrity Measurement; Embedded System*

1. Introduction

The wide application of embedded system is closely related to people's life and its safety also becomes hot topics in the study of computer technology, especially computing environment safety. To prove whether the embedded computing environment used and interacted with is credible or not becomes the focus of attention and this is the problem of computing environments certificate (Trustworthiness Attestation).Trustworthiness Attestation turns out to be process of measuring and verifying the state of computing platform and Trusted Computing provides credible solution to solve the trusted certificate. Attestation is the most important features of trusted computing, and the basis of trusted relationship. Many institutes and scholars have conducted a lot of research on trustworthiness attestation, and got a lot of achievements. TCG (Trusted Computing Group) proposed, based on TPM (Trusted Platform Module), binary attestation TBA (TPM-based Binary Attestation) method [1]. Based on TBA, IBM put forward IMA (Integrity Measurement Architecture) [2], which implements the integrity measurement of components before loading and supports remote attestation. But there are some problems such that lack of information privacy protection, difficult to adapt to system dynamic expansion requirements and low validation efficiency, in TCG integrity measurement method before loading and remote attestation. Sadeghi A R, Ahmad R S et al.

ISSN: 1738-9968 IJHIT

carried out the study on security attribute [3-6], and proposed PBA (Property-based Attestation) method, which to some extent, solved the privacy problem, but still did not solve the low efficiency and difficult to implementation of dynamic credibility and attestation. Li Xiaoyong et al. put forward trustworthiness attestation based on the behavior of the system [7, 8], while, Wang Shihua et al. proposed trusted atteatation method based on strategy[9]. The two methods respectively change the platform state attestation into trusted attestation of historical behavior sequence and feasible strategy, but they only consider part of the system security attributes, and the granularity of system behavior and strategy measurement is coarser. Virtualization is one of the hotspots in trusted computing research. Shi Guangyuan et al. put forward a dynamic credible attestation method based on safety Virtual Machine Introspection (VMI) [10], which use VMI technology in dynamic verification procedures on whether actual behavior is consistent with the expected behavior, to judge the credibility of the program. However, Xen and KVM virtual technology are still limited to I86 general processor, so it's difficult to be implemented on most embedded processor platform. Dual kernel technology is a kind of layered architectures proposed by RT Linux [11] and RTAI [12] in order to improve the Linux real-time performance. It, under the standard Linux kernel, realized a small real-time kernel, and the original non-real-time Linux kernel runs on the small kernel as a preemptable task, and all tasks run in the kernel address space. In this paper, using RT Linux Kernel technology [11] for reference, an embedded system dynamic trustworthiness attestation based on dual kernel (bi-Kernel TRAM, the biKernel-based Trustworthiness Remote Attestation Method) is proposed. We design reliable attestation architecture, use the dual kernel technology to solve the problem of proving system security, and verify the integrity on system important components and key data in program running, in order to improve the security of the system.

The paper is organized as follow: Section 2 gives the bi-Kernel architecture of the system; Section 3 describes the measurement and verification methods; Section 4 gives the system implementation and related results and Section 5 is the conclusion.

2. The Trustworthiness Remote Attestation System based on Dual Kernel

Attestation is the procedure of providing evidence or (and) logical reason to the Challenger to proof its some properties. A typical attestation process includes three participants: AT (Attestation), the party proposing service requesting, CH (Challenger), the party providing the service, and a trusted third party acting as mediator AP (Appraiser), as shown in Figure 1.

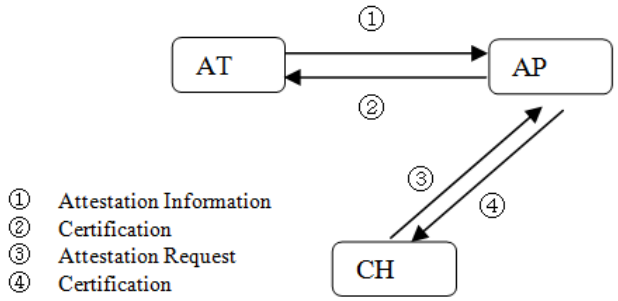

Figure 1. A typical trustworthiness attestation system

AP is a stand-alone machine, and it can arbitrate credibility of all ATs in the system. In such a system, AP is easy to become the bottleneck of the whole system.

In the bi-Kernel TRAM architecture, as shown in Figure 2, the AP acts as a super kernel (called AP-Kernel) running on the ordinary kernel (called AT-Kernel) and AT sojourns on the same machine, but AP-Kernel and AT-Kernel are stored physically isolated.

AP-Kernel is unchangeable which is similar to TPM and the nature of its composition is static (non-changing). AP-Kernel with system privileges cannot be accessed by any user and thusly, CH must trust the AP-Kernel. AT-Kernel is running as an AP-Kernel process therefore AP-Kernel can control AT-Kernel running, get AT-Kernel running state, or direct access and check all resources of the AT-kernel to determine if a violation of the rules such as an attack occurs. AP-Kernel can also check the sequences and variables called by the applications so that it can find the untrusted behaviors in the program running due to an attack. Meantime, AP-Kernel provides the integrity measurement on key date structure such as kernel module list, and it can detect whether there is hidden suspicious malicious module loaded into AT-kernel, thus checking AT and sending the results to CH.

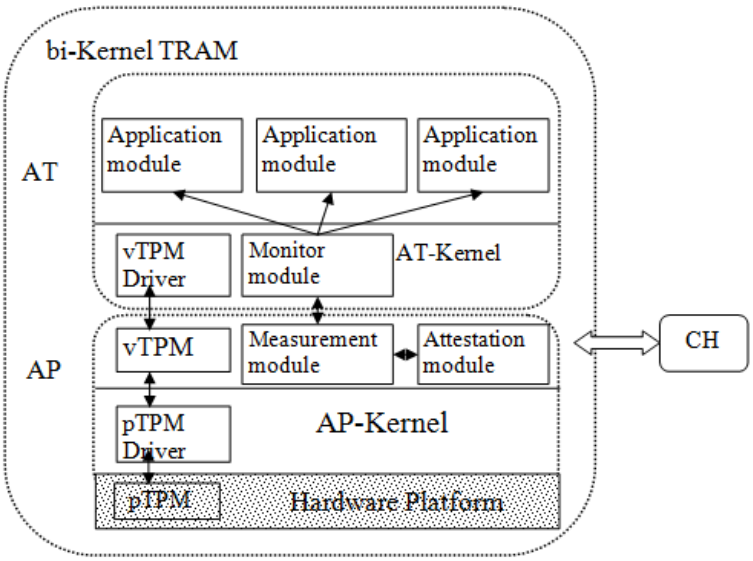

Figure 2. bi-Kernel TRAM trustworthiness attestation system

AT-Kernel runs as an AP-Kernel process, but a program in AT-kernel runs the same as on a common kernel, that is, dual kernel architecture is transparent to the program in AT-Kernel. For the convenience of AP-Kernel monitoring the operations of the AT-Kernel system, a monitoring module is added in the AT-Kernel kernel, which intercepts system calls of AT-kernel and records the related parameters then send to AP-Kernel through a particular data channel.

AP-Kernel only measures AT-Kernel after receiving request from CH, while does nothing in the normal system running, therefore the entire CPU is substantially occupied by AP-kernel process in AT-Kernel so that the application system performance is not affected.

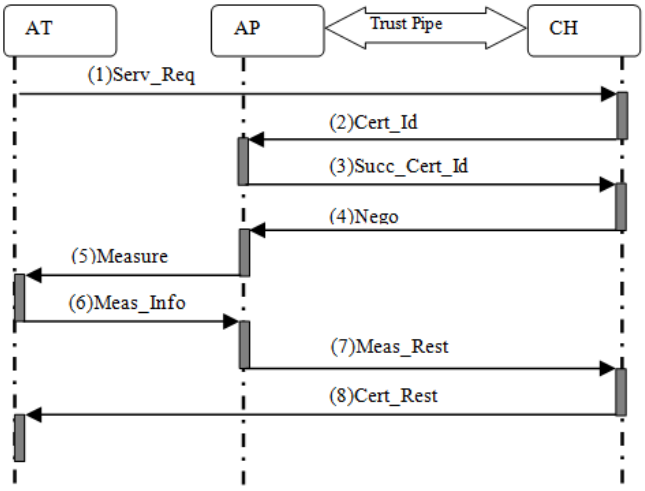

Figure 3. the attestation interaction of bi-Kernel TRAM architecture

bi-Kernel TRAM architecture does not affect the Consultant CH. CH can still think that the arbiter AP is an independent existence, and that AP-Kernel is completely trustworthy. For after the system is deployed AP-Kernel is no longer allowed to vary, CH can also verify credibility of AP-kernel by integrity measurement. bi-Kernel TRAM allows users to dynamically change AT-kernel configuration according to demand.

In bi-Kernel TRAM architecture, AT, AP and CH tripartite processing attestation interaction is shown in Figure 3.

AT sends service requests Serv_Req to Consultant CH to ask access to CH service resources; CH verifies the identity of the AP Cert_Id after receives the request; if certificate validation is successful, then CH and AP will establish a credible pipeline Trust_Pipe, otherwise mark AP exception and attestation terminates; CH negotiates Nego with AP, that is CH sends the request to AP to measure the integrity of AT; AP measures AT according to the measurement requirements proposed by CH and gets the information cared by CH; AP returns the prove results to CH by trusted pipeline Trust_Pipe and CH validates attestation results; if the attestation results validation is successful, then CH allows AP to access its services, or refuses to provide service.

3. Measurement and Verification Methods

As similar to virtual machine technology architecture, the bi-Kernel TRAM architecture designed for embedded system can support a variety of trusted measurement mechanisms, for example, the model of trusted program behavior attestation described in [10] can be introduced in the architecture.

3.1 Static Integrity Measurement

In bi-Kernel TRAM architecture, AT-Kernel, which runs user programs, need to measure the integrity of user programs, but AT-Kernel is running on top of AP-kernel and cannot directly access physical TPM, so we introduce a software virtual TPM (vTPM) into the AP-Kernel, which is a kernel process of AP-kernel; for AT-kernel, vTPM can be accessed and used as a real TPM. The trusted software architecture based on bi-Kernel TRAM with vTPM is shown in Figure 4.

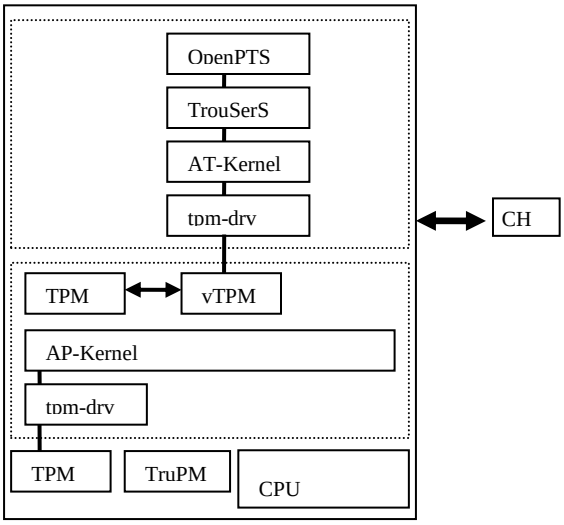

Figure 4. The trusted software architecture of bi-Kernel TRAM with vTPM

The BIOS of bi-Kernel TRAM system adopts modified PMON (TruPMON), which supports Trusted Boot. The TruPMON structure is shown in Figure 5. TruPMON divides PMON into CRTM and T-Loader to enhance the design. CRTM, as a root of trusted measurement of computers, partly realizes platform initialization sub-module, TPM driver sub-module, trusted measurement module and trusted call sub-module; CRTM mainly accomplishes basic platform initialization, including the initialization of CPU, memory, Cache and stack, establishment of T-Loader program to minimize operating environment, initialization of TPM hardware and building the Trusted Root (Measurement Root, Storage Root, Report Root); T-Loader is used to load the operating system.

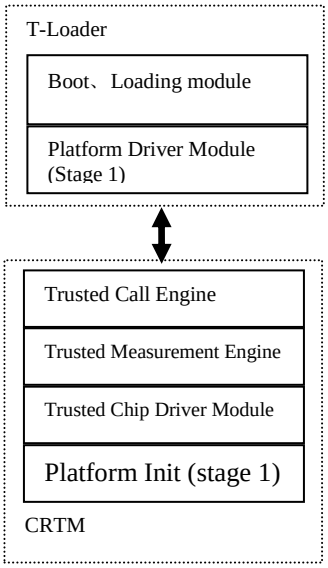

Figure 5. TruPMON architecture

The trusted protocol stack uses open source software TrouSerS [13] and trusted services platform uses OpenPTS [14]. The introduction of vTPM makes the introduction of trusted

root to AT-Kernel. The system startup process is as follows: After the system powers on, TruPMON boots; firstly, it measures AP-Kernel, and extends the results to hardware TPM, then AP-Kernel starts; secondly, AP-Kernel measures AT-Kernel and stores the results to corresponding PCRs in vTPM, then AT-kernel starts; AT-kernel completes the measurements of follow-up loading application modules, and extends the results to corresponding PCRs in vTPM.

bi-Kernel TRAM takes TPM and TruPMON as trusted root, and gets the credibility of platform by continuously submitting credible measurement reports to match an entity to its expected value. The Root of Trusted Measurement (RTM) and Root of Trusted Report (RTR) join up to provide current measurement data of the platform, and compare these measurement results with their expected values to determine whether the platform operations meet the expectations. The trusted chain transfer process is shown in Figure 6.

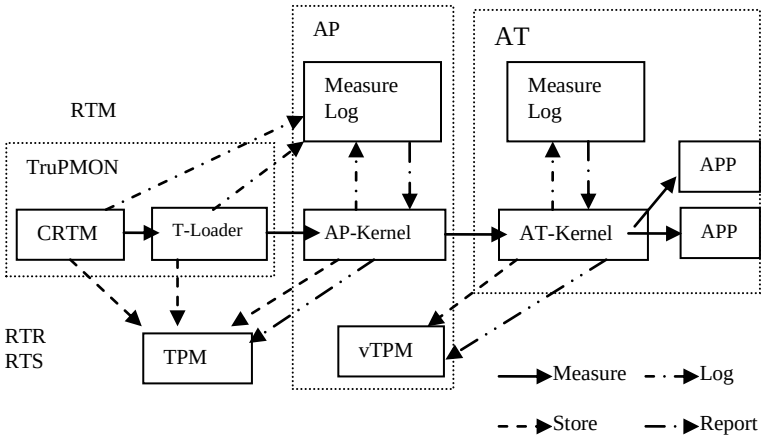

Figure 6. Trust Chain Transfer of bi-Kernel TRAM

3.2 Dynamic Integrity Measurement

However, the credibility of program in loaded time is not equal to that in runtime. Paper [15] presents a self-modifying code technology SMC (Self-Modifying Code) which can let program modifies its code during operations. SMC makes the static binary representation of the program different between loaded and running. SMC behaviors are usually divided into two steps: first, modify the read and write permissions of memory pages which store the program code; second, cover corresponding specific code to achieve its purpose to modify the code. Paper [16] gives a dynamic tracking process instruction technology for dynamic integrity measurement of the process, but it does not give the impact on system performance by such an instruction. [17, 18] introduce Intel TXT (Trusted Execution Technology) based on the Intel VT (Virtualization Technology), however it is difficult applied to other architectures.

For dynamic trusted measurement of the program, we design a dynamic integrity measurement method: analysis and measure the binary file of the program running on AT-kernel before loading, and store the measurements to corresponding PCRs in vTPM; in Linux system, every loaded program corresponds to a process, and for every process system maintains a memory descriptor mm_struct which records all address space information of the process, and in mm_struct, unsigned long variable start_code, end_code, start_data and end_data respectively points to starting or ending address of program code and data. When

need to attest the running program, we re-measure the program and compare the results with that measured in pre-loaded time to determine whether the code segment modifies, so that we can detect the occurrence of the attacks. The dynamic integrity measurement architecture (DynIMA) is shown in Figure 7.

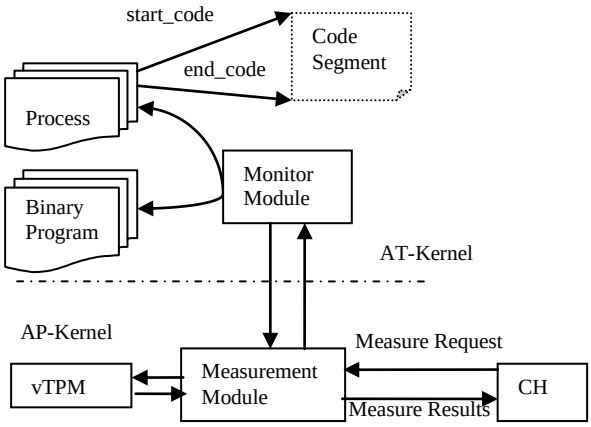

Figure 7. DynIMA Architecture

The attestation process is as follow:

1) Measurement module receives the request from CH party to attest certain program;

2) Measurement module communicates with monitor module which parses corresponding process name of the program in AT kernel, and finds the process in process list, gets the process handle, and then gets the address where the code segment stores; and then finishes corresponding address conversion;

3) Measurement module use vTPM to hash and sign the contents of the corresponding code segment;

4) Measurement module returns the information signed by vTPM to the challenge party CH;

5) The challenge party CH validates the measurement results to decide the next action, and returns the results.

4. System Implementation and Testing

4.1 Implementation

We, by the ideas of RTLinux realization, implemented bi-Kernel TRAM on Godson 2F platform. As shown in Figure 8, similar to RTLinux, AP-Kernel is a real-time nanokernel; AT-Kernel is a modified version of standard Linux kernel 2.6.18.

In interrupt management, AP-Kernel intercepts the hardware interrupt of Godson IP2-IP7 [19], and AT-Kernel is isolated with interrupt controller by soft interrupt. The concrete approach is that replace standard Linux kernel interrupt switch function (cli, sti), interrupt return function (ret_from_irq), interrupt handler (do_IRQ) and interrupt vector table, etc. [20], AT-kernel's interrupt process is replaced by simulated soft interrupts, that is to say, with a set of variables to record the interrupt switch and the occurrences, all hardware interrupts

are intercepted by AP-kernel, and according to the interrupt types and flags it determines whether or not to assign interrupt to AT-Kernel.

For task scheduling, AP-Kernel redefines the scheduling function. AT-Kernel runs as a low-priority task of AP-kernel, but normal user process can still run on the AT-Kernel and use the variety services provided by AT-Kernel.

The measurement module runs as a real-time process on AP-kernel, and communicates with AT-Kernel by using the Mbuff communication mechanism of RTLinux.

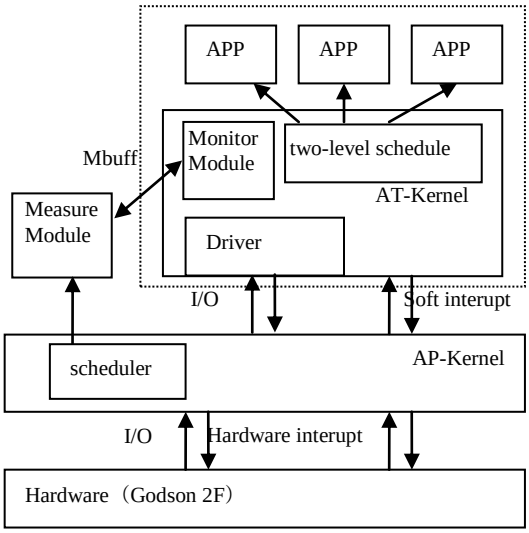

Figure 8.Dual kernel architecture based on Godson 2F

4.2 Testing

First, we tested the SMC attack detection ability. On the principle of SMC, we designed and implemented a program SMC_demo which can attack by dynamically modify their own code. The program started by printing a message to show it has started. After a random period of time, the program filled code segment from back to front with 0xaa by 4 bytes (enough to form a system call address). The system uses DynIMA mechanism, and the results of the attestation process output are shown in Figure 9. From the test results, we can see that DynIMA can detect changes in the code segment and make up the defect of static integrity measurement.

We also tested the time consumed for the measurement by testing the static integrity measurement time and dynamic integrity measurement time of four different magnitude modules (as shown in Figure 10 and Figure 11). The static module size, code segment size and measurement time of the four test program are shown in Table 1 (static test and measurement did not consider the disk read time). As outlined in Table 1, the measurement time is linear to the size, and the relation is about 62us/MByte. The time consumption is very small and does substantially no effect on the system operation because vTPM is virtual software and the measurement of vTPM is calculated by CPU and PCR operations are memory operations, thus its operational performance is far superior to physical TPM.

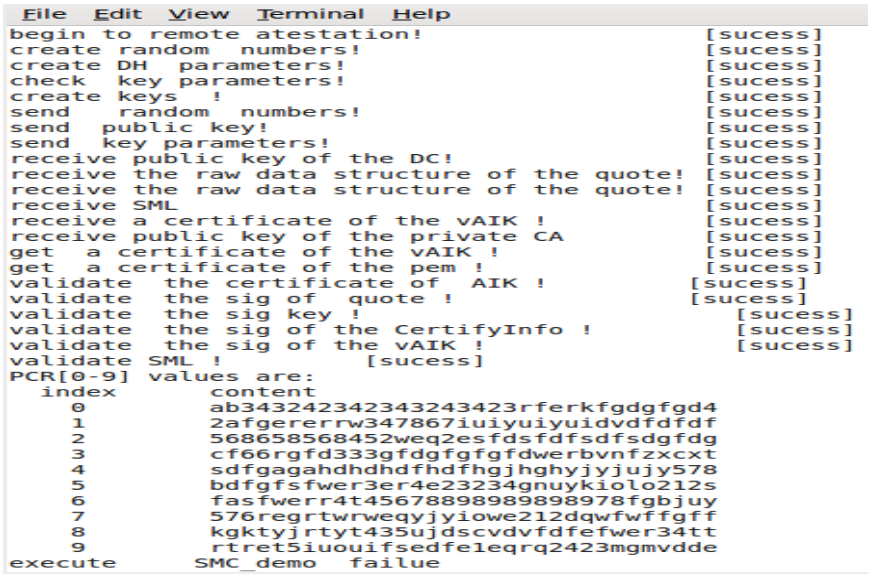

Figure 9. The outputs of attestation

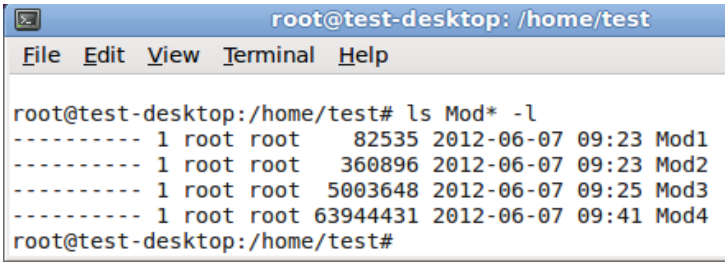

Figure 10.Four different magnitude modules

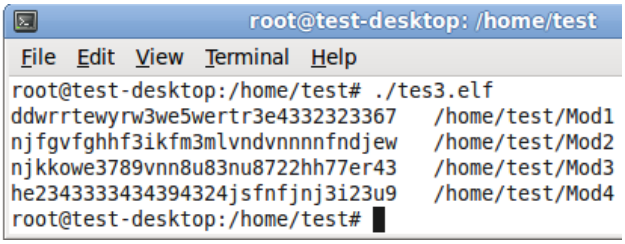

Figure 11. The hash of four different magnitude modules

Table 1. Time of Measurement

Medule N.O.	Static Length(KB)	Code Segment Length(KB)	Time of Static Measurement(us)	Time of Dynamic Measurement(us)
Mod1	80.6	61.216	5	4
Mod2	352.437	237.522	22	15
Mod3	4886.375	1003.358	303	62
Mod4	62445.733	15573.834	3872	966

5. Conclusion

For trustworthiness attestation demand of embedded systems, we take the reference of RTLinux dual-core technology, and apply super kernel as trusted third party. Through code dynamic measurement make up the defect of static integrity measurement method and improve the credibility of system operations in runtime. Based on TrouSerS, OpenPTS, PMON et al. open source technologies, modify PMON structure by reference to TCG-BIOS requirements and implement on Godson 2F platform. The test results show that the system can effectively deal with self-modifying code technology attack and measurement time also meets the demand of real-time systems. In the future, we will further study the dynamic measurement of data section method.

References

[1] TRUSTED C G. TCG Specification Architecture Overview, Reversion 1.4, **(2007)**.
[2] R. Sailer, X. Zhang, T. Jaeger, *et al.*, "Design and implementation of a TCG-based integrity measurement architecture", Proceedings of the 13th USENIX Security Symposium, San Diego, CA, USA, **(2004)**.
[3] A. R. Sadeghi and C. Stuble, "Property-based attestation for computing platforms: caring about properties, not mechanisms", The 2004 New Security Paradigms Workshop, Virginia Beach, VA, USA, **(2004)**.
[4] J. Poritz, M. Schunter, E. V. Herreweghen and M. Waidner, "Property attestation scalable and privacy friendly security assessment of peer computers", IBM Research Report RZ 3548, http://domino.watson.ibm.com/library/cyberdig.nsf/papers/215E33CB2B4F7FA485256E97002A0D6C/$ File /rz3548.pdf, **(2004)**.
[5] U. Kuhn, M. Selhorst and C. Stuble, "Realizing property-based attestation and sealing with commonly available hard- and software", Proceedings of the 2007 ACM Workshop on Scalable Trusted Computing, New York, NY, USA, **(2007)**.
[6] K. Rene, R. S. Ahmad, S. Christian, *et al.*, "A practical property-based bootstrap architecture", Proceedings of the 2009 ACM Workshop on Scalable Trusted Computing, Chicago, Illinois, USA, **(2009)**, pp. 29-38.
[7] X. Y. Li, C. X. Shen and X. D. Zuo, "An efficient attestation for trustworthiness of computing platform", Proceedings of the 2006 International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP'06), **(2006)**.
[8] X. Y. Li, X. D. Zuo and C. X. Shen, "System behavior based trustworthiness attestation for computing platform", Acta Electronic Sinica, vol. 7, no. 7, **(2007)**, pp. 1235-1239.
[9] S. H. Wang and X. Y. Li, "Policy Based Trustworthiness Attestation for Computing Platform", Acta Electronic Sinica, vol. 4, no. 4, **(2009)**, pp. 900-904.
[10] G. Shi, C. Shen and Y. Liu, "Dynamical attestation for trust based on secure virtual machine introspection", Journal of China institute of communications, vol. 11, no. 11A, pp. 24-38.
[11] C. E. Hall, "A real-time Linux system for autonomous navigation and flight attitude control of an un inhabited aerial vehicle", Proceedings of the 20th Digit al Avionics Systems Conference Proceedings, Daytona Beach, FL, USA: IEEE Press, vol. 2001, **(2011)**, pp. 1A11-1A19.
[12] G. Zhang, L. Chen and A. Yao, "Study and comparison of the RTHAL-based and ADEOS-based RTAI real-time solutions for Linux", Proceedings of the First International Multi-Symposiums on Computer and Computational Sciences, Hangzhou, China: IEEE Press, **(2006)**, pp. 771-775.
[13] TrouSerS, http://trousers.sourceforge.net/ v0.3.6, The open-source TCG Software Stack.
[14] openpts, http://sourceforge.jp/projects/openpts/wiki/FrontPage, Open Platform Trust Services.
[15] Y. -d. Wu, Z. -g. Zhao and T. -w. Chui, "An attack on SMC-based software protection", Springer Berlin/ Heidelberg, **(2007)**, pp. 232-248.
[16] L. Davi, A. -R. Sadeghi and M. Winandy, "Dynamic integrity measurement and attestation: towards defense against return-oriented programming attacks", Proceeding STC '09 Proceedings of the 2009 ACM workshop on Scalable trusted computing, **(2009)**, pp. 49-54.
[17] M. Gillespie, "Intel® Trusted Execution Technology: A Primer", http://software.intel.com/en-us/articles/intel-trusted-execution-technology-a-primer/, Intel Software Network, **(2009)** June 1.
[18] S. Choinyambuu, "A Root of Trust for Measurement", MSE Project Report, **(2011)** June 15.
[19] Institute of computing technology, Chinese academy of sciences. User Guide of Godson 2F Platform v0.2 [Z], **(2007)**.
[20] Linux kernel source, http://www.kernel.org/pub/linux/kernel/v2.6.

Author

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