2012 International Conference on Information and Network Technology (ICINT 2012)
IPCSIT vol. 37 (2012) © (2012) IACSIT Press, Singapore

A Systematic Method For Securing Memory Modules By Using Cryptographic Co-Processor

Ashutosh Singh and Nitesh Keswani [+]

Tata Consultancy Services

Abstract. Security and privacy are community-derived—one cannot hope to be secure even if one person on the network does not take the same defensive measures that one does. If everyone is secure and every computer 'trusts' every other machine, there is little reason to be afraid of a malicious hack or attack.
This is a content-protection concept that has spawned from the belief that the PC, as it currently stands, is not architecturally equipped to protect a user from the pitfalls and challenges that an all-pervasive network such as the Internet poses. A drastic change in the PC hardware is not feasible due largely to economic reasons. This paper hopes to introduce a minimal change that will serve the dual purpose of granting a computer user security, even as it maintains backward compatibility with existing applications. These changes will be reflected in the key components of the central processor, a motherboard chip (cryptographic coprocessor), input and output devices. It will not be a separate operating system; rather, it will be based on low-level changes to the OS kernel itself. Secondly, it is meant as an enforcer of security measures— measures that will either be defined by an end user or a content generator.

Keywords: cryptographic co-processor, gatekeeper, memory module, boot module, virtual memory module.

1. Introduction

This concept is meant to cultivate a computing environment, termed as 'trusted space' [5,6], which a user can rely on for security and privacy. At the heart of the system lies a secure cryptographic co-processor, that is, a tamper-resistant embedded smart card, which contains unique keys and a variety of cipher implementations. It thus acts as a unique identifier of a particular system and is the base on which security measures are implemented. Additional changes made to the CPU will allow the central processor to run in a special mode. Finally, to enable secure communication over a motherboard's data buses, the chip too will have to be modified. Built upon this hardware foundation and extending its functionality is a tiny piece of software termed the Nexus. It represents a change in the kernel of the operating system and takes the form of an independent micro-kernel[10]. The Nexus is run in an ultra-privileged mode of the CPU where certain areas of memory are strictly reserved for it and are restricted to other, potentially malicious, pieces of code. This micro-kernel dictates the rudimentary security policies that the rest of the system must follow.

Under the scheme, a Nexus can be changed, thus changing the security parameters and guidelines. These rudimentary policies can be changed by using a different Nexus. The Nexus is the manager of trusted memory and an arbitrator of access to the cryptographic co-processor[3,8]. When such a PC is booted for the first time, an inventory of the system hardware will be taken and authenticated. If a part of the hardware is later changed or replaced, the system will have to be re-authenticated via a central authority. Assuming that the system's hardware is recognised as legal, the Nexus is then loaded in an encrypted state into the restricted memory space. It will then reside there as long as the system is up, all the while dictating the system's security policies and ensuring that applications do not trespass into restricted memory or file systems. It will not be possible for any software to modify the Nexus or subvert the policies laid down by it.

[+] Corresponding author. Tel.: + 033-24912335.
E-mail address: keswani.nitesh3@gmail.com.

The Nexus is thus protected by the hardware, while it protects the software that forms the applications and data. While the Nexus maintains a low profile, managerial stance, entities known as Trusted Agents (TA) [6,7] do the legwork of interacting with the user. A TA can either be a – manufacturer's trusted application or an add-on to an existing program, imparting security awareness to it.

Although existing applications will run on such a system, they will not be able to use the security features without TAs. To better comprehend the relation between a Nexus and TAs, consider the analogy of a language. If the Nexus defines the rules of grammar for the language, the TA uses those rules to construct sentences as a means of communicating securely with other systems and users.

1.1. Spheres of trust

The Nexus and TAs together provide trusted data storage, encryption services for applications and facilities to enable hardware and software to authenticate themselves and each other. Central to the this concept are 'Spheres of trust'[5,7], wherein a user can not only define which program to trust but can also ask it to use only a certain set of data. Its like trusting one's doctor with very sensitive health information. It is unlikely, however, to share with him the particulars of one's bank account—details that the broker is aware of. The broker, in turn, is in the dark about the details of one's health.

Thus, the trust one lends to the physician does not extend into the financial life, just as broker's 'sphere of trust' does not encompass information on health. Similarly, such a system divides user data into separate spheres or vaults. Each trusted application is granted access to either a section or an entire vault.

1.2. Information security

A Using the Nexus and the supporting hardware, such a system is cryptographically verifiable to the user and to other computers or services. The system can ascertain that other computers are trustworthy before engaging them in a transaction involving sensitive data. This is termed as attestation and it forms a cornerstone of successful security and privacy measures. Take the instance of a bank operation. Banking software has to be highly trusted and moreover, it must be ensured that the computer dealing with it does indeed belongs to us and not to some malicious party masquerading under one's identity. The hardware should thus trust the banking software and vice-versa. This virtual handshake to exchange and verify credentials is made possible by attestation. Once attestation is done, exchange of data takes place. It is important that data is only accessible to applications that have been identified as trusted. Once our personal data is transmitted to the bank, we could so arrange the transaction that only particular software can read such information.

Such control is made possible by a mechanism called sealed storage[9]. Sealed storage is done by encryption or decryption techniques using the cryptographic co-processor. To seal a piece of data, it is appended with the hash of the system's Nexus identifier (a 160-bit number). It is then encrypted with a private key—a secret key unique to each system. When a sealed data is given to such a system, the cryptographic coprocessor attempts to decrypt it using its platform-specific private key. If successful, the hashed Nexus identifier is extracted and decrypted to determine which Nexus is permitted to receive the data. If the Nexus that currently resides on the computer is the same as the one that was present while the data was sealed, the data is unsealed and read.

Two things emerge from this: 1. You cannot take a sealed file and transfer it to another machine to be unsealed there since the platform-specific key will change and the process will fail at the first step. Such a measure is essential if you wish to store sensitive data on a non-trusted hard disk or over a network. For this reason there will be software extensions that will implement one of three migration policies: (a) Migration is entirely prevented by default setting, (b) Migration is permitted upon entry of a valid password, (c) Migration is allowed with the consent of a third party, for instance, permission is required while downloading an MP3 file from a music company. 2. The same Nexus must be present while sealing and unsealing the data. If you have changed the micro-kernel after encrypting a file, you will have to go back to the older one to read the file. Thus, applications can only decrypt sealed data if they run on the same hardware and under an identical software environment (as dictated by the Nexus) under which the data was sealed. This ensures that under a trusted Nexus, an application can decrypt and process the data but cannot

subvert a TA's policy. If you're not running under a trusted Nexus, you could run software that undermines the policy set by a TA, but the application will no longer be able to decrypt the sealed data since the Nexus has changed. The concept does not require user information for authentication purposes; instead, it secures the hardware and software environment. The Nexus identifier is not unique to a system, since it is likely that the majority of people will use the same Nexus. The only identifier unique to the system will thus belong to the cryptographic coprocessor [8].

1.3. Proposed method

Now we require a systematic method that allows for a secure memory architecture where the memory module is not directly accessible. In accordance with today's needs , a systematic method has been described which substantially eliminate or reduce disadvantages with previous systems and methods for securing the memory modules(and the system).A secured memory module including a memory module, a virtual memory module, and a gatekeeper (hereafter identified as cryptographic coprocessor) allows for a memory module that is accessible via the virtual memory module and therefore not accessible where access of the memory module requires the authentication of the key by the gatekeeper.

In accordance with one aspect of the present disclosure an information handling system is provided. The information handling system includes a processor, a memory module, a virtual memory module, a boot module and a cryptographic coprocessor. The boot module accepts a key and one or more requests for the memory module and provides the requests and the key to the virtual memory module. The virtual memory module is externally accessible by the boot modules and accepts the key and one or more requests from boot module[1,4]. The virtual memory module transmits the one or more requests and the key from the boot module to the cryptographic coprocessor. The cryptographic coprocessor receives the key from the virtual memory module and authenticates the key and does not allow modification of the memory module if the key is not authenticated.

More specifically in one embodiment, a stored key is stored in programmatically inaccessible location within either the cryptographic coprocessor or the memory module .When the cryptographic coprocessor receives the request and the key from the boot module via the virtual memory module, it retrieves the stored key from the programmatically inaccessible location and compares the key provided by the boot module with the stored key. If the key provided by the boot module and the stored key are the same, then the key is authenticated and the request is processed within its dedicated memory. If the execution of the request exceeds permitted memory or processor utilization then the request is denied. If the request is found to be safe then it is dispatched to the memory module and processed. The present disclosure provides a number of technical advantages. One important technical advantage is the inaccessibility of the memory module . With the addition of the cryptographic coprocessor, the boot module no longer operates as the guardian for the memory module and the boot module no longer writes directly to the memory module. Instead, the boot module writes to the virtual memory module believing that the virtual memory module is the memory module. Read and write requests are only executed in the memory module by the cryptographic coprocessor if the key provided when accessing the boot module matches the key stored in the programmatically inaccessible location. Therefore, the memory module is not directly accessible by the boot module and therefore not accessible to any outside users and is difficult to hack for unauthorized changes. In addition, since the key is stored in a programmatically inaccessible location within either the cryptographic coprocessor or the memory module, neither of which are directly accessible, the key is not accessible to any outside users and cannot be easily retrieved by an unauthorized user in order to make unauthorized changes to the memory module.

Another important technical advantage is that the secured memory module architecture functions with existing information handling system components. The memory module[2], cryptographic coprocessor, and virtual memory module are designed so as to not require any changes to the other components of the information handling system[1,2]. The virtual memory module is designed so that it appears to the boot module as the memory module and the boot module does not have to be redesigned in order to communicate requests to the virtual memory module instead of the memory module. Therefore, when the boot module

accesses the virtual memory module, the boot module believes that it is interacting with the actual memory module and not the virtual memory module.

A significant technical advantage is that even if an authorised user inadvertently or deliberately requests execution of malicious code, the system denies the request. Therefore the system is equipped even against intentional sabotage.

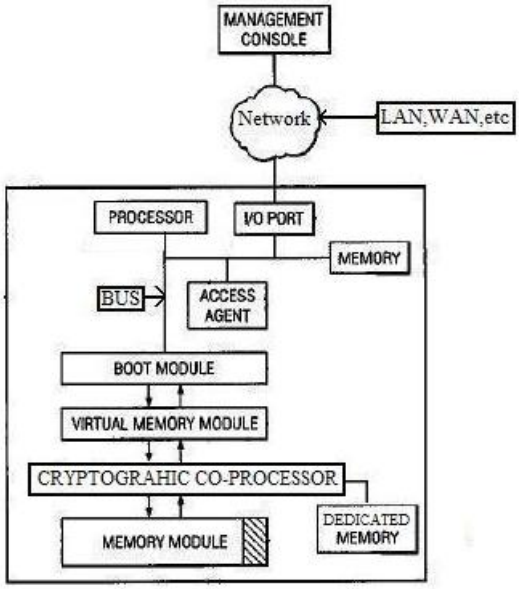

Fig. 1: Block Diagram

FIG. 1[modified from 1] illustrates a block diagram of a system including information handling system. In the example embodiment, information handling system includes respective software components and hardware components, such as processor, memory, input/output ports, access agent, boot module, virtual memory module, memory module, cryptographic coprocessor with its dedicated memory and those components work together via bus to provide the desired functionality. Bus provides a communication link between the above components and is typically used as a control, diagnostic or power management bus with interconnects and transfers of information between the various components of information handling system. The various hardware and software components may also be referred to as processing resources. Information handling system may be a personal computer, a portable computer, a personal digital assistant, a server, or any other appropriate information handling device.

Network may be LAN, a WAN, the Internet, intranets, or any other type of communication network. Memory module is a memory device storing various information regarding information handling system. The memory module stores information such as boot order and system configurations for information handling system. Memory module may be a non-volatile memory such as CMOS or flash memory or any appropriate type of memory module like RAM.

Gatekeeper is a cryptographic coprocessor that regulates and controls access to memory module. Because gatekeeper regulates access to memory module, only gatekeeper can access and modify memory module. It also has a dedicated memory which is isolated (inaccessible) from rest of the system. The key is a symmetric key or password that is required by gatekeeper for access to memory module. Secure location is a programmatically inaccessible location that is accessible only to gatekeeper. Virtual memory module is a hardware component that appears as actual memory module to all components of information handling system except gatekeeper and memory module.

FIG. 2 (below) illustrates a flow diagram of an example method for securing access to memory modules. A user accesses information handling system[1] to control it while management console remotely accesses information handling system to allow for remote control and operation of information handling system. When the user accesses information handling system and boot module, the user generally desires to provide a request to memory module. In previous systems, when the boot module received a request from the user, the

module would execute the request in the memory module often without verifying the validity of the request or the user.

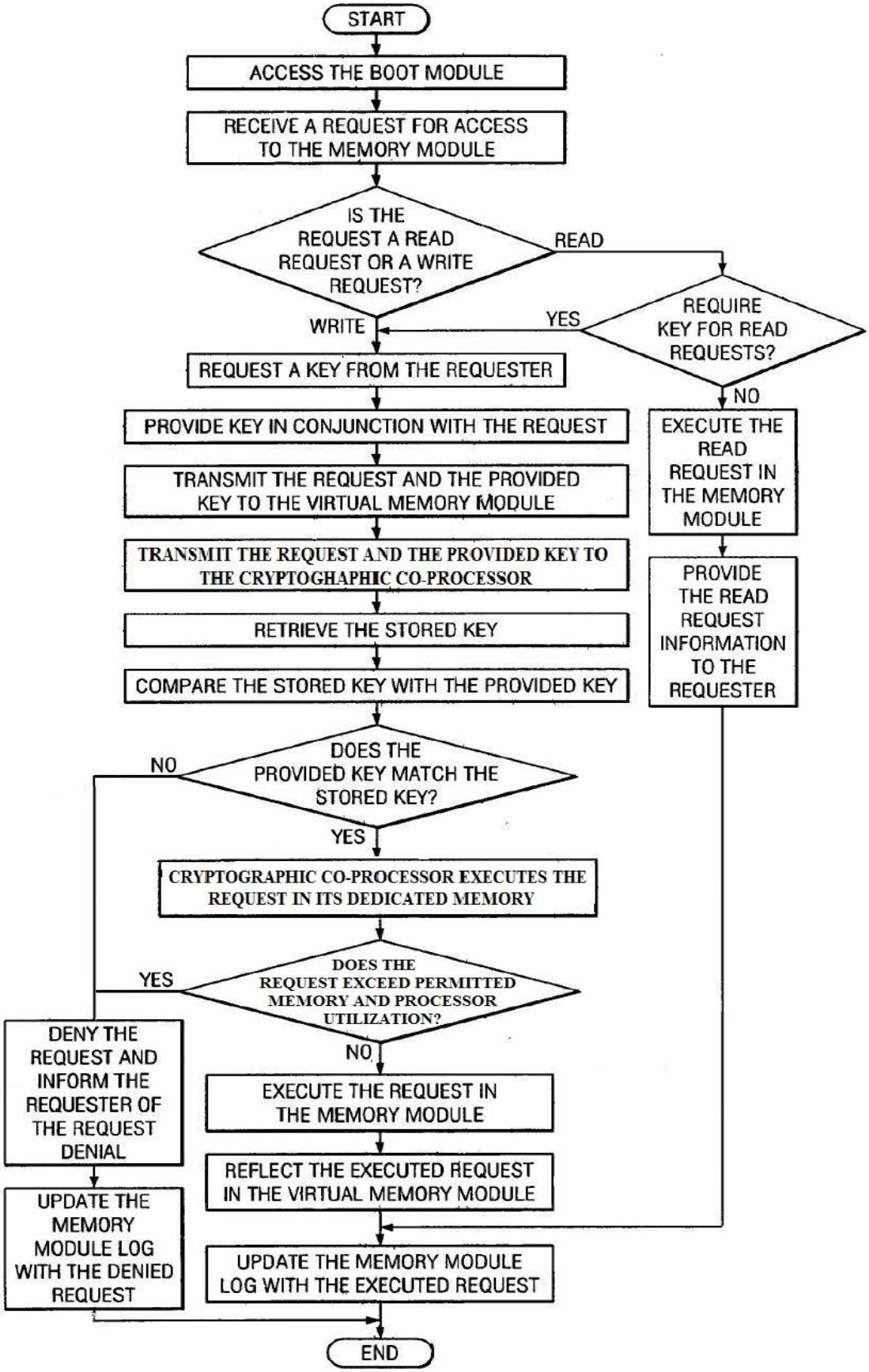

Fig. 2: Flow Diagram

In the present disclosure, when boot module receives the request, it determines if the received request is a request to write to memory module or to read from memory module. If the request is a write request, then the boot module requests from the user the key. As previously described, the key is a symmetric password or secret code that is required by gatekeeper (cryptographic co-processor) in order to access memory module. In order to process the request within memory module, gatekeeper must first authenticate the key provided by the user. To authenticate the provided key, the cryptographic co-processor compares the provided key with a stored key stored in secure location. Hashes of the key may also be utilized instead of the actual key. Creating and retrieving hashes of the provided key and stored key allows for a greater level of security.

If the provided matches the stored key, then the provided key and accordingly the user has been properly authenticated and gatekeeper assumes that user is an authorized user of information handling system. The Cryptographic co-processor has a dedicated memory in which it test-runs the execute request for malicious behaviour and it denies the request which exhibit malicious behaviour. If the provided key does not match the stored key, then the gatekeeper denies the request and does not execute the request within memory module. If the request is a read request the gatekeeper determines whether the key is needed for a read of the memory module. If a key is required for a read request, the gatekeeper requests the user for the key.

If gatekeeper receives one or more simultaneous requests, whether read, write or combination, from boot module at same time it serializes the requests first in, first out sequence. In other embodiments, an importance or urgency value may be associated with each request and therefore gatekeeper processes the requests from most important or urgent to least important or urgent.

2. Conclusion

Therefore a method has been proposed to defend a system against any threat. Not only does it prevent unauthorised access but also the execution of malicious request even if an authorized user makes such a request. Hence the memory module is protected through the cryptographic co-processor which makes sure that only safe and authorized request are provided access to the memory module.

3. References

[1] US Patent no. US 7308102 B2 - Douglas M. Anson et al., December 2007.

[2] US patent no. US 5844986 – Dl Davis, December 1998.

[3] US Patent 2002/0099950 A1 – Smith, July 2002.

[4] US Patent 2005/0033970 A1 – Douglas M. Anson et al, 2005.

[5] Trusted Computing, Digital Rights Management, and the Fight for Copyright Control on Your Computer, Fall 2003, 2003 UCLA J. L. Tech. 8 by Ryan Roemer.

[6] Security & Privacy, IEEE, Felten, E.W.; Princeton Univ., NJ, USA, Issue Date: May-June 2003 Volume: 1 Issue: 3 On page(s): 60 – 62.

[7] Flexible OS support and applications for trusted computing, Tal Garfinkel, Mendel Rosenblum, Dan Boneh. Proceeding HOTOS'03 Proceedings of the 9th conference on Hot Topics in Operating Systems - Volume 9.

[8] FPGA implementation of RSA public-key cryptographic coprocessor, Hani, M.K.; Tan Siang Lin; Shaikh-Husin, N.; TENCON 2000. Proceedings On page(s): 6 - 11 vol.3.

[9] Operating System Security: Microsoft Palladium for Dr. Chris Taylor by Karl Heins, February 3, 2003.

[10] Certifying program execution with secure processors, Benjie Chen, Robert Morris, Proceedings of the 9th conference on Hot Topics in Operating Systems - Volume 9 Pages 23 – 23.