Effective Implementation of the Cell Broadband Engine™ Isolation Loader

Masana Murase
IBM Research,
Tokyo Research Laboratory
1623-14 Shimotsuruma
Yamato-shi, Kanagawa-ken
242-8502 Japan
mmasana@jp.ibm.com

Kanna Shimizu
IBM Corporation,
IBM Systems & Technology Group
One Rogers St, Cambridge,
MA 02142
kannas@us.ibm.com

Wilfred Plouffe
IBM Research,
Almaden Research Center
650 Harry Road San Jose, CA
95120
plouffe@almaden.ibm.com

Masaharu Sakamoto
IBM Research,
Tokyo Research Laboratory
1623-14 Shimotsuruma
Yamato-shi, Kanagawa-ken
242-8502 Japan
sakamoto@jp.ibm.com

ABSTRACT

This paper presents the design and implementation of the Cell Broadband Engine™(Cell/B.E.) isolation loader which is a part of the IBM Software Development Kit for Multicore Acceleration [14]. Our isolation loader is a key component in realizing secure application boot and encrypted application execution. During the application load process, the isolation loader fetches, validates, and decrypts a Synergistic Processor Element (SPE) executable, establishing a chain of trust from the hardware to the application. Since not all applications are SPE executables, we also introduce a general solution. This is a verification service framework in which all applications including system functions can be verified by the isolation loader immediately before execution.

We have applied several novel implementation techniques to the isolation loader. The countermeasure implemented in our isolation loader against the substituted-ciphertext attack is given and our staging technique to allocate contiguous working areas for applications is also introduced. The load overhead of this loader including application fetch, validation (RSA-2048/SHA-1), and decryption (RSA-2048 and AES) is less than 50 milliseconds on the 2.8 GHz IBM PowerXCell 8i processor. This overhead is reasonable compared with the 500-millisecond 2048-bit RSA signing needed by the Trusted Platform Module chips [3].

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection

General Terms

Design, Performance, Security

Keywords

Cell Broadband Engine™, encrypt-then-sign, isolation, multi-core, substituted-ciphertext attack

1. INTRODUCTION

Although modern CPUs with multiple cores are the predominant trends for enhancing performance and power efficiency, the multi-core feature has another aspect for computer usage – *isolation*. In 2005, Percival [6] reported on a security flaw of Hyper-Threading Technology by which malicious software could steal secret data such as encryption keys or passwords from caches or registers. This is because two threads, one a legitimate thread handling a secret and the other a malicious thread observing the legitimate thread, can share caches and registers when they are running on the same processor core. To address such security risks, it is necessary to prevent illegal and unexpected access to shared resources. The multi-core system allows us to confine each thread to the corresponding cores with concurrent execution of those threads. In particular, if there is no dependency among threads, we can benefit from both performance and security enhancements. We assume three hardware security features to realize such a secure environment: (1) on-chip memory isolation, (2) runtime secure boot, and (3) decryption during the software boot.

The Cell Broadband Engine™(Cell/B.E.) [4] processor is one of the multi-core processors supporting such capabilities. As introduced in [21], the security features are called the secure processing vault[1], the runtime secure boot, and the hardware root of secrecy, respectively. Unlike competing security solutions, this design is unique in that even if the

[1]Also referred to as the Synergistic Processor Unit (SPU) isolation mode

supervisory software such as the operating system or the hypervisor is compromised, the process isolation is guaranteed. In contrast, most security architectures rely on the perpetual integrity and security of their supervisory software to protect and separate the processes. With these features, we can create an isolated and secure domain on each core which is independent of the traditional insecure domain where the operating system or the hypervisor is running.

While [21] focuses on the security hardware architecture of the Cell/B.E. processor only, we present the detailed design and implementation of our security software stack for the Cell/B.E. processor in this paper. Our work provides a secure software-based application loader minimizing the application load overhead on top of the Cell/B.E. processor. The Cell/B.E. processor validates and decrypts the Synergistic Processor Element (SPE) isolation loader first. Afterward, the authorized loader validates and decrypts a signed and encrypted application (secure application) every time the secure application is initiated. In this way, a chain-of-trust is established and maintained from the hardware layer at the bottom to the application layer. This layered structure gives us flexibility and portability for application development. The software-based application loader provides separation of the hardware and the application layer so that if one changes (for example, a different cryptographic algorithm is used), the other is not affected.

Unlike the prior approaches, we use the encrypt-then-sign policy, but not a naive one for the application signing and encryption. This policy has a large advantage in the response time to detect tampering. Since a verifier with this policy validates the target application prior to the decryption, that verifier can release the CPU resource owned by this process to another process as quickly as possible when it detects tampering. In contrast, the sign-then-encrypt policy needs both decryption and validation operations to detect tampering. Sometimes, the decryption process can be time-consuming, which occurs in our experiments. We also introduce a novel staging technique to allocate contiguous working areas for applications. This technique is much of importance when the isolated memory space is limited.

Prior research on secure-main processors such as AEGIS [22], XOM [16], and Cerium [5] also provided the same hardware security features. Unlike our approach, this work used the sign-then-encrypt policy to generate a secure application. The sign-then-encrypt policy is an effective way to cope with a substituted-ciphertext attack [9], but it takes a long time to detect tampering because both decryption and validation must be performed to detect tampering in the sign-then-encrypt approach. This approach is reasonable if a verifier is implemented in the hardware and the overhead of the cryptographic operations can be ignored. However, we need to care about the performance overhead in addition to the security if we use a software authentication layer to provide flexibility and portability of applications. The secure main processor work does not address on this concern.

Prior work on secure co-processors [27, 18, 26] is another approach to implement the isolated domains. The main purpose of this approach is to validate the software stack without changing the existing main processors. Once it has been ensured that a system has not been tampered with, then the authorized supervisory software such as the operating system or the hypervisor divides the runtime environment into two or more domains: one is the secure domain, and any others are non-secure domains. Thanks to the ring protection mechanism [13] implemented in the main processor, the secure domain is guaranteed to be isolated from the other domains. Compared with the secure main-processor work, the secure co-processor approach does not care about malicious supervisory software which might be hacked or compromised. Also, these secure co-processor systems are missing the encrypted application execution. How to implement signed and encrypted application execution in an efficient way is still an open question.

The remainder of this paper is structured as follows. In Section 2, a brief introduction to the Cell/B.E. processor is given as background for this paper. In Section 3, a secure application load technique and a contiguous working area allocation technique are proposed to realize a high-performance and secure application loader. Section 4 describes the performance evaluation of the SPE isolation loader and a code verification service, which is an extension of the SPE isolation loader, is implemented in Section 5. Section 6 clarifies the difference between our approach and the existing technologies. Finally we summarize our work in Section 7.

2. CELL BROADBAND ENGINE™ OVERVIEW

The Cell/B.E. is a multiprocessor core architecture (see Figure 1) [4]. The cores are heterogeneous and there are two kinds of cores on each chip. The principal core, the 64-bit Power Processor Element (PPE), is a PowerPC processor that has the supervisory role. It is the PPE that executes the operating system and manages the allocation of most system resources, including the SPEs. The other type of core on a Cell/B.E. is the SPE consisting of a Synergistic Processor Unit (SPU) and a Direct Memory Access (DMA) engine. In the current implementation, there are 8 SPEs per chip. The SPEs are the computational workhorses: a RISC-style single instruction, multiple data (SIMD) instruction set, wide and large (128 128-bit) register files, and 256 KB of physically dedicated private memory, called the Local Store (LS), for each SPE [10]. The high bandwidth Element Interconnect Bus (EIB) connects these processor cores to each other and to the off-chip system memory and I/O.

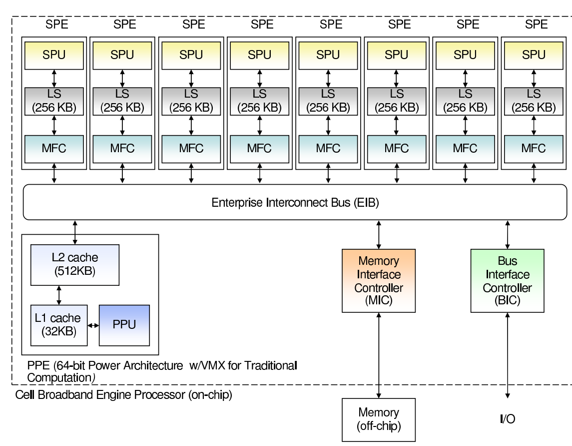

Figure 1: The diagram of the Cell Broadband Engine architecture