SecCMP: A Secure Chip-Multiprocessor Architecture

Li Yang
Department of CSEE
University of Tennessee at Chattanooga
Chattanooga, TN 37405
Li-Yang@utc.edu

Lu Peng
Department of ECE
Louisiana State University
Baton Rouge, LA 70803
lpeng@lsu.edu

ABSTRACT
Security has been considered as an important issue in processor design. Most of the existing mechanisms address security and integrity issues caused by untrusted main memory in single-core systems. In this paper, we propose a secure Chip-Multiprocessor architecture (*SecCMP*) to handle security related problems such as key protection and core authentication in multi-core systems. Threshold secret sharing scheme is employed to protect critical keys because secret sharing is a distributed security scheme that matches the nature of multi-core systems. A critical secret is divided and distributed among multiple cores instead of keeping a single copy that is sensitive to exposure. The proposed *SecCMP* can not only enhance the security and fault-tolerance in key protection but also support core authentication. It is designed to be an efficient and secure architecture for CMPs.

Categories and Subject Descriptors
C.1.2 [**Multiple Data Stream Architectures (Multiprocessors)**]: *Interconnection architectures, Multiple-instruction-stream, multiple-data-stream processors (MIMD), Connection machines.*

General Terms
Algorithms, Design, Security

Keywords
Chip-Multiprocessor, Security, fault-tolerance, encryption

1. INTRODUCTION
Computer networking makes every computer component vulnerable to security attacks. Examples of such attacks include passive eavesdropping over communication between CPU cores and off-chip devices, injection of malicious code like buffer overflow, denial of service attacks by malicious code, and attacks from compromised off-chip or on-chip devices. Security has been considered as an important issue in processor design because pure software solutions are not secure enough. Many proposed work focus on hardware memory encryption and authentication in single-core systems [2, 8, 10, 11]. They usually consider the processor chip as one safe and trusted unit. However, because Chip-Multiprocessors (CMP) have become mainstream products, adversaries could steal on-chip shared critical secrets by compromising one core. Approaches to protect shared critical secrets for CMPs are demanded.

In this paper, we propose a novel *Secure Chip-Multiprocessor (SecCMP)* architecture which employs a distributed *Secret Sharing* [6] approach to protect and distribute critical secrets shared by all processor cores. *Secret Sharing* is a robust distributed security scheme with low overheads and improved fault-tolerance. The distributed security management matches multi-core architecture in CMPs very well. By employing a threshold *Secret Sharing* implementation, critical secrets can be protected safely in a CMP processor even when one or more processor cores are compromised. Moreover, confidentiality and authentication among cores are supported in *SecCMP*. The proposed *SecCMP* architecture enhances system security and fault-tolerance by critical secrets protection and core authentication. It is designed to be an efficient and secure architecture for CMPs. We use an application to demonstrate secure and remote critical information access and sharing supported by our *SecCMP.* Integrated with identity based cryptography [1], the *SecCMP* provides a secure and reliable way to generate and distribute encryption keys between local host and remote site when prior distribution of keys is not available.

The rest of this paper is organized as follows. Section 2 introduces related work. Section 3 states the attack and thread model of this paper. We discuss the *master private key* protection and core authentication of the *SecCMP* architecture in Section 4. An application of the *SecCMP* is to support critical information remote access and sharing is given in Section 5, followed by security and computational complexity analysis in Section 6. Finally, we summarize this paper and briefly introduce the future work in Section 7.

2. RELATED WORK
Lee et. al. [3] proposes a *"secret-protected (SP)"* architecture focusing on key protection and management, featured by secure on-line access of users' keys from different network computing devices. The keys are organized as a tree structured key chain rooted at a secret *"User Master Key"*. With helps from additional hardware features supporting *Concealed Execution Mode (CEM)* and

ASID'06, October 21, 2006, San Jose, California, USA.

Trusted Software Module (TSM), the *SP* architecture protects confidentiality and integrity of sensitive data transmitted between processor chip and off-chip devices. Our proposed mechanism can enhance the security for the SP processor architecture working on a CMP. With a threshold distributed secret sharing, even if one or more pieces of critical secrets are released, the adversaries still cannot obtain the secrets.

In [9], the authors present an integrated framework utilizing multi-core processors to detect intrusions and recover from infected states. The processor cores are divided as resurrectors and resurrectees and memory space is also insulated. Resurrectees cannot access resurrectors' memory but resurrectors can access all the memory space. Fine grain internal state logging for low privileged cores, resurrectees, is employed. Resurrectors dynamically check the states of resurrectees. If any suspicious intrusions are detected, a logged state will be recovered.

Other works [2,8,10,11] emphasize on memory encryption and authentication by efficient hardware approaches in single-core systems. Our proposed scheme focuses on on-chip secret protection in multi-core processors. Collaborating with the above memory protection and recovery schemes, our proposal will further enhance security and fault-tolerance of CMP systems.

3. ATTACK AND THREAT MODELS

In this paper, we focus on the remote exploit attacks from networks, which intrude on-chip critical secrets in multi-core processors. An attacker will try to learn critical secrets of authorized users or elevate his/her privilege or consume the victim's resources. Encryption technologies, firewalls and intrusion detection systems have been designed to protect the users from attacks such as password sniffing, password cracking, buffer overflow, deny-of-service etc. However, the adversaries still are able to launch attacks if the encryption key is not proper managed or databases of firewall or IDSs are not up to date.

During an eavesdropping attack, an attacker tries to learn critical secrets that he/she was unable to access such as the root password. Buffer overflow attack happens when a process attempts to store data beyond the boundaries of a fixed length buffer, the extra data overwrites adjacent memory locations. By doing so, possible malicious code is injected into an execution path. If executed, the injected malicious code grants attackers unauthorized privileges. Therefore, both eavesdrops and buffer overflow try to increase privileges of an attacker and expose the critical information to unauthorized attackers.

We describe countermeasures of the proposed *SecCMP* by an application of remote critical information access and sharing in Section 5.

4. SECURE CHIP MULTIPROCESSOR ARCHITECTURE

Each processor has a pair of *master public key* and *master private key*. The *master public key* is available to all other network devices and hosts. The *master private key* is protected by a threshold secret sharing scheme in the *SecCMP*. Protection of the *master private key* allows critical information related to an application to be stored and accessed over public network. Each core in the processor also employs new hardware features to supported trusted applications. Trusted applications generate and distribute application related keys and encrypt, decrypt the remotely shared critical information. Because all the key computations, distributions and critical information encryptions and communications are protected by the trusted applications, an adversary can not observe other cores' secret share even when one or more cores are compromised. Most importantly, the *SecCMP* comprises two unique components: protection of a master (chip) private key and core authentication.

4.1 Protection of the Master Private Key

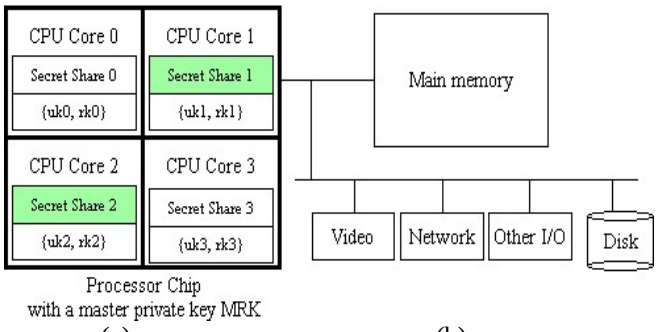

Figure 1. A Four-Core CMP System with Secure Chip Multiprocessor (*SecCMP*) Architecture.
(a) Processor Chip with a *Master Private Key* MRK.
(b) Off-chip Devices

The *master private key* is used to generate and an application private key (we call it as an *account private key* in Section 5) and decrypt the application related critical information stored on-line and accessed over public network, i.e., banking PIN, PGP keys. To avoid a central failure point in a single processor system like [3], the *master private key* is divided and distributed among multiple CPU cores in a processor chip. In (*k, n*)-threshold secret sharing, each core c_i holds a secret share ss_i, and any *k* of these *n* cores can reconstruct the *master private key*. Any collection of less than k partial shares can not get any information about the *master private key*. Here, *k* is the threshold parameter such that $1 \leq k \leq n$. Each processor will authenticate itself in fine grained intervals. Therefore, it is difficult for adversaries to obtain *k* or more pieces of secret shares during a short time.

Figure 1(a) is an example of (2, 4)-threshold scheme among 4 cores where *a master private key MRK* is divided into 4 unique pieces (Secret Share 0, 1, 2, 3), such that any 2 of them can be used to reconstruct the *master private key MRK*. Traditional Shamir's secret sharing scheme suffers from the requirement of a trust authority and the absence of share verification. We employ a scheme based on [6], which is an extension to Shamir's secret sharing without the support of a trust authority. We also deploy the verifiable secret sharing [7] to detect the invalid share that some shareholders generate to prevent reconstruction of the *master private key*.

4.2 Core Authentication

One or more cores may be compromised and their secret share is exposed to an attacker. We assume that an honest core will present the correct share to authenticate itself, and a compromised core will present a random number instead of the correct share. The attacker learns the secret shares from compromised cores and interrupt the *master private key* reconstruction. Failure of *master private key* reconstruction will result in a denial of service attack (DoS). In order to exchange secret share securely, each core c_i hold a public/private key pair $\{<uk_i, rk_i>\}$ to encrypt the secret share and authenticate each other. Each core signs its secret share and hash code with its private key (digital signature). Then the signed message is encrypted with requesting core's public key. The requesting core decrypts the message with its private key. Then requesting node checks the signature to authenticate the sender, checks the hash code to make sure the integrity of the secret share. The key pair is created during core installation based on each core's identity. An adversary is not able to observe the encrypted share without a correct key pair. The private key also serves similar function as *Device Master Key* in *CEMs* discussed in Lee's work [3] and is used to encrypt the critical key sent to and from off-chip devices.

We not only passively protect the *master private key*, but also actively detect the compromised core. In order to detect the compromised cores, we design a series of *m master private key*s such that none of the participants knows beforehand which is correct. The *master private key*s are ordered incrementally based on their values, except for the real key. The participants combine their shares to generate one key after the other, until they create a correct key that is less than the previous key. This helps us to expose the compromised core early, before the correct *master private key* is generated. The detection and prevention of cheaters in threshold schemes [4] will be adopted in our approach. Once the compromised core is detected, we isolate the compromised one. The work in [5] allows a new sharing scheme to be activated instantly once one of the cores becomes untrustworthy.

5. *SecCMP* SUPPORTED CRITICAL INFORMATION ACCESS AND SHARING

An application of the *SecCMP* is to support critical information remote access and sharing. *SecCMP* provides secure channels to generate, store and exchange encryption keys for local host and remote sites to share critical information associated with a specific account (i.e., a bank account, an email account). Each host has a pair of *master public key* (*MUK*) and *master private key* (*MRK*). In addition, each account has a pair of *account public key* (*AUK*) and *account private key* (*ARK*). Based on the identity-based cryptography [1], a user *account public key* can be any arbitrary string. In other words, users may use some well-known information, such as email address, IP address, URL as their *account public key*.

When a local host tries to retrieve critical information from a remote site, it creates a pair of MUK and MRK. The MUK is available to the remote site, and the MRK is distributed and stored in multiple cores of the local host. Such distributed design of the *master private key* is resistant to eavesdrops since at least k cores need to be compromised in the active session to reconstruct *master private key*. Moreover, *k* out of *n* cores need to be contacted in order to create an *account private key* based on the account ID. A buffer overflow attack may expose secret share of a core or interrupt private key generation from a core. Our core authentication service could detect such attacks. Because an attack from network exploits usually cannot be performed in a very short duration, the undergoing attack can be reported and blocked before *k* cores are compromised.

5.1 Identity-based Cryptography

Identity-based systems [1] allow any party to generate a public key from a known identity value such as an ASCII string. The Private Key Generator (PKG) generates the corresponding private keys. To operate, the PKG first publishes a *master public key*, and keeps the corresponding *master private key*. Given the *master public key*, a public key can be generated corresponding to the identity by any party. To obtain a corresponding private key, PKG is contacted to generate the private key using *master private key* based on the identity. As a result, messages may be encrypted without prior key distribution between individual participants. Such solution is helpful when the pre-distribution of the authentication keys is not available. A major challenge of this approach is that the PKG must by highly trusted since it generates any user's private key and thus decrypt messages. In the *SecCMP*, multiple cores work together to provide secure private key generation service when there is no prior distribution of keys.

5.2 Remote Information Access and Sharing

When a local host tries to access or retrieve critical information from a remote site. The local host and remote

site need to authenticate and exchange *master public key* with each other. How two remote hosts authenticate each other and how local host authenticate the current user is valid or not are out of the scope of this paper. The former can be accomplished either by a Certificate Authority (CA) or a trusted third party. The later can be achieved through access control or biometrics. We focus on how remote site and local host generate, store and distribute *master public key*, *master private key*, *account public key* and *account private key*.

Figure 2 shows the general procedure of remote information access and sharing. To initiate the remote critical information access, the local host sends its *master public key* to remote site in the step 1. In step 2, the local host also sends account ID (*AcctID*) whose critical infor-mation the user would like to access. The remote site com-putes an *account public key* based on a *master public key* and an account ID (*AcctID*). After that, the remote host encrypts requested critical information with the *account public key* and transmits requested critical information to public networks in step three. In step 4, the local host computes *account private key* by interacting at least *k* out of *n* cores in the local host processor, and uses the *account private key* to decrypt the received encrypted critical information.

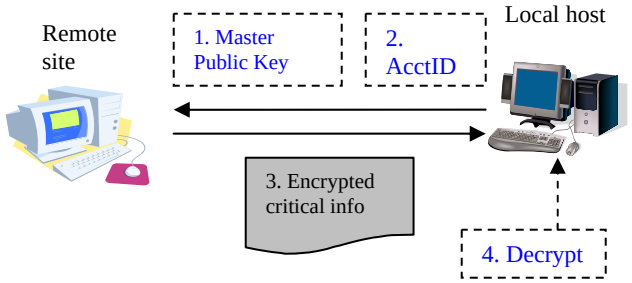

Figure 2: Remote Critical Information Access

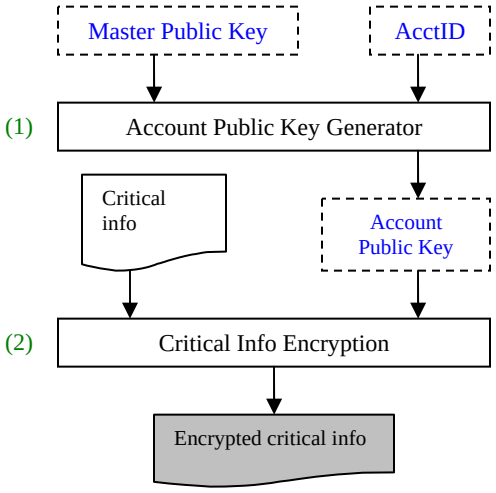

Figure 3: *Account Public Key* Generation and Encryption in the Remote Site

The remote site needs to generate an *account public key* to encrypt requested critical information and send it to public networks. The local host will obtain *account private key* to decrypt the critical information encrypted using its corresponding public key. Figure 3 shows the *account public key* generation and critical information encryption in remote site.

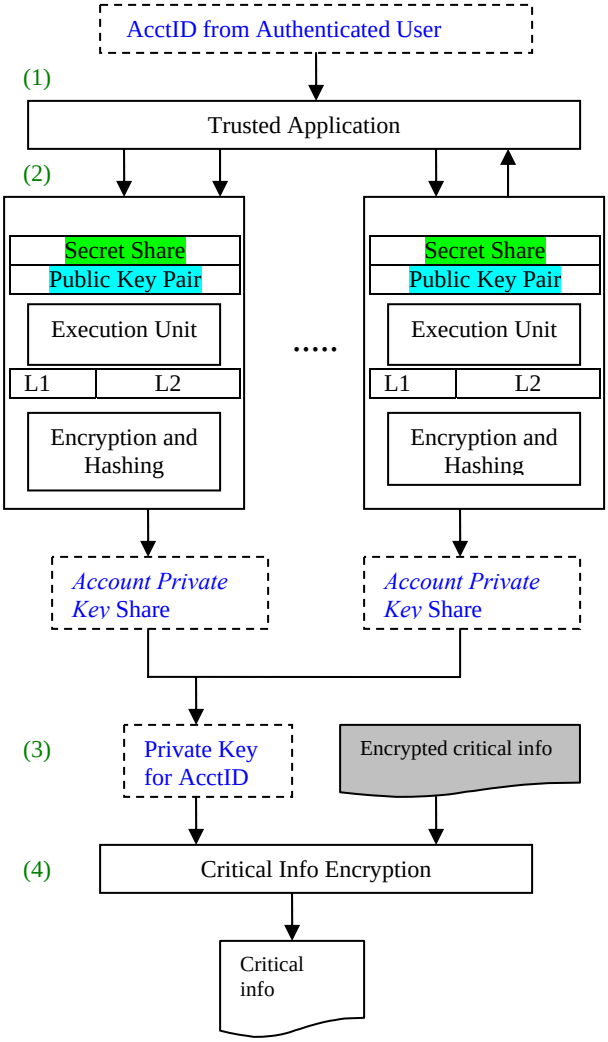

Figure 4: *Account Private Key* Generation and Decryption in the Local Host

The local host generates an *account private key* by which the encrypted critical information is decrypted, as shown in Figure 4. The method to obtain the *account private key* is to contact at least *k* cores, present the account identity and request private key generation service.

The trusted application is the only application that can access the secret share in registers of a core. When a trusted application sends the *MUK* to remote site, the *MRK* was divided into n secret shares and kept in n cores respectively.

The general procedure of the account private key generation and decryption in the local host is listed as follows:

(1) Authenticated users input to the trusted application and *AcctID* whose related critical information will be retrieved through public network by an authenticated user.
(2) Each core generates an *account private key* share.
(3) *K* shares of the *account private key* construct the corresponding *ARK* for *AUK* of the *AcctID*.
(4) The *ARK* is used to decrypt the cipher text encrypted by the corresponding *AUK*. By doing so, a user can securely access and share critical information remotely.

6. SECURITY AND COMPLEXITY ANALYSIS

Confidentiality and *Integrity* are taken care of by the encryption and hash function performed on secret share. Hash function guarantees that a share being transferred is never corrupted due to non-benign failure. *Availability* ensures the survivability of a processor chip despite denial of service attack. In our schema we take care of this problem by making use of *(k, n*) threshold secret sharing algorithm, as any *k* out of *n* cores work together for critical master key reconstruction. Thus our security solution is tolerant to *k-1* compromised cores. *Authentication* is taken care of by digital signature that enables a core to ensure the identity of the peer core it is communicating with. No adversary can masquerade a core, thus gaining unauthorized secret share. Main computations in our approach come from secret share reconstruction and encryption. The reconstruction *computational complexity* depends on the number of thresholds. The encryption *computational complexity* is same as the traditional schemes and depends on the size of shares. The shorter length of a share results in less resource consumption. The computations will be accelerated by involvement of multi-cores.

7. SUMMARY AND FUTURE WORK

In this paper, we briefly introduce a secure architecture design for CMPs. The proposed architecture employs a threshold *Secret Sharing* scheme to protect critical secrets and support core authentication for a CMP system. Moving forward, we will implement the proposed scheme and evaluate its performance and security by software simulations.

8. ACKNOWLEDGMENTS

This work is supported in part by the Louisiana Board of Regents grants NSF (2006)-Pfund-80 and LEQSF (2006-09)-RD-A-10 and the LSU Faculty Research Grant program. This work is also supported in part by Tennessee Higher Education Commission's Center of Excellence in Applied Computational Science and Engineering under grants R04-1302-005. Anonymous referees provide helpful comments.

9. REFERENCES

[1] D. Bonh and M. Franklin, "Identity-Based Encryption from Weil Pairing," Advances in Cryptology, CRYPTO 2001, Lecture Notes in Computer Science, Vol. 2139, pp. 213-229, Springer Verlag, 2001.

[2] B. Gassend, G. Suh, D. Clarke, M. Dijk, and S. Devadas, Caches and Hash Trees for Efficient Memory Integrity Verification, *In Proceedings of the 9th Intl. Symposium on High-Performance Computer Architecture* (HPCA), 2003.

[3] R. B. Lee, P. C. S. Kwan, J. P. McGregor, J. Dwoskin, Z. Wang, Architecture for Protecting Critical Secrets in Microprocessors, *In Proceedings of the 32nd International Symposium on Computer Architecture*, 2005.

[4] H.-Y. Lin, L. Harn, A Generalized Secret Sharing Scheme with Cheater Detection, Advances in Cryptology-ASIACRYPT '91 Proceedings, Springer-Verlag, 1993, pp. 149-158.

[5] K.M.Martin, *Untrustworthy Participants in Perfect Secret Sharing Schemes, Cryptography and Coding III*, M.J. Ganley, ed., Oxford: Clarendon Press, 1993, pp. 255-264.

[6] T. P. Pedersen, A Threshold Cryptosystem without A Trusted Party, *In Proceedings of EUROCRYPT*, 1991.

[7] T. P. Pederson, Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing, *Lecture Notes in Computer Science*, pp. 129-140, 1992.

[8] W. Shi, H.-H. Lee, M. Ghosh, C. Lu, and A. Boldyreva, High Efficiency Counter Mode Security Architecture via Prediction and Precomputation, *In Proceedings of the 32nd International Symposium on Computer Architecture*, 2005.

[9] W. Shi, H-H Lee, L.Falk and M.Ghosh, An Integrated Framework for Dependable and Revivable Architectures Using Multicore Processors, *In Proceedings of the 33rd International Symposium On Computer Architecture,* 2006.

[10] C. Yan, B. Rogers, D. Englender, Y. Solihin, M. Prvulovic, Improving Cost, Performance, and Security of Memory Encryption and Authentication, *In Proceedings of the 33rd International Symposium On Computer Architecture*, 2006.

[11] J. Yang, Y. Zhang, and L. Gao, Fast Secure Processor for Inhibiting Software Piracy and Tampering, *In Proceedings of the 36th International Symposium on Microarchitecture*, 2003.