ISSN:0975-9646

An Effective Approach of Data Security for Distributed Shared Memory Multiprocessors

Md. Shafakhatullah Khan[1] (Member IAENG), Mourad Mohamed Henchiri[2]
[1, 2] Department of Information Systems,
University of Nizwa, Nizwa, Sultanate of Oman, Oman.

Abstract—**The Concept of Distributed System made life easier to communicate and share resources from any other system with the help of network. Due to the emergence of Distributed system, Data Security has become an increasing concern, and respectively attacks related to hardware mechanism have emerged. However the researchers have implemented so many techniques for hardware encryption and authentication mechanisms as a means of addressing this security issues. Thus, no such techniques have been produced for Distributed Shared Memory (DSM) multiprocessors. Although we have many proposed approaches for uniprocessor and symmetric Multiprocessor (SMP) systems, any of those approaches are not useful for DSMs. In this work we use signcryption technique to provide effective encryption and authentication of the data in DSM systems.**

Keywords—**Distributed Shared Memory, Data Security, Encryption, signcryption, and Authentication.**

I. Introduction

Security is the prominent factor in the design of modern computer systems, and many researchers have recently started to collect information about tamper-resistant execution environments as a means to protect the privacy and integrity of sensitive data in these systems. Providing a secure environment has become more challenging with the increase of hardware attacks, such as power analysis, timing, snooping devices which can be attached to various buses [7, 8]. Because of such attacks, software-based approaches for security are no longer enough since sensitive information used by security software, such as encryption keys themselves, can be compromised because they are kept as unencrypted program variables stored in the main memory and transmitted over the system bus.

Since software-based security mechanisms are themselves vulnerable to hardware attacks, that’s why hardware-based security strategies are required. This hardware support has been proposed in the form of memory encryption to protect the confidentiality of data and memory authentication which leads to secure the integrity of data [5, 6, 12, 13, 16, 17, 18, 21, 23, 24, 25]. With this type of data protection, many important security issues in computing, such as Digital Rights Management (DRM) violations, software piracy, and reverse engineering of code, can be addressed effectively. One important class of systems that will require tamper- resistant designs for data secrecy and integrity are Distributed Shared Memory (DSM) Multiprocessors. [1]

Now-a-days as almost all the companies are using large-scale DSM systems is in the context of utility or on-demand computing so the company owning large systems will host computational and storage resources of the system to customers who want to utilize their IT operations. For example, DSM systems such as the HP Superdome are already being used to offer on-demand computing services [15] to a variety of users. Because large DSMs are powerful but expensive, customers often run critical applications which access and store confidential corporate data (e.g. financial data, product in- formation, client records, etc.) on them. As the utility computing model grows in popularity, a large no of companies will adopt this model, and DSMs will host a wider range of applications using many types of sensitive data. In addition, DSM will be the powerful architecture of these systems since SMP cannot scale to large systems easily. Since these DSM systems are located in a physically remote location, which makes customers often very concerned about the privacy and integrity of their computations, particularly against hardware attacks that may be very hard to detect or trace. For Industries data privacy is one of the major deals, which leads fast adoption of the on-demand computing model [4].

This concern may cause customers to acquire on-demand computing providers to utilize tamper-resistant mechanisms in their DSMs. We note that it is unlikely for the utility computing provider itself to be malicious, as this would create a poor business model. Rather, a large-scale DSM system owned by a corporation will likely be protected with relatively tight physical security that restricts system access to select

F. A. Md. Shafakhatullah Khan is with the Department of Information Systems, University of Nizwa, Nizwa, Sultanate of Oman, Oman. (phone: +968 98118903, e-mail: shafakhat@ unizwa.edu.om.)

S. B. Mourad Mohamed Henchiri is with the Department of Information Systems, University of Nizwa, Nizwa, Sultanate of Oman, Oman. (e-mail: Mourad@ unizwa.edu.om.)

employees. However, lessons from history have taught us that it is unlikely that this single layer of security would be fail-proof. For example, despite the relatively good physical security protection and limited access for Automatic Teller Machines, there have been repeated cases of ATM fraud by some supposedly trusted employees [2]. In one case, an employee inserted a PC into an ATM machine to monitor and steal customer accounts and PINs. DSMs used for on-demand computing are in a similar situation in that the main ingredients that are conducive for physical tampering are there. First, DSMs (like ATMs) store highly valuable information belonging to many customers. For DSMs, this information may include financial data, product information, and client records. Second, the financial motivation to perform an attack can be large because stolen information is valuable to other corporations (corporate espionage) or criminals (identity theft). Finally, there exist some forms of attacks that hardly leave any traces. For example, physically inserting a snooping device in a DSM would be quite easy due to the exposed interconnection at the back of server racks. USB drive-sized devices with multi-GB storage can likely be attached and removed in a matter of seconds without shutting down the system, and without leaving visible traces. Thus, many corporations will likely wish to add another, difficult to break, layer of protection for the security of their data in the form of tamper-resistant DSM systems.

Architectural support for data secrecy and integrity has been studied extensively by researchers for uniprocessor systems [5, 6, 12, 13, 17, 18, 21, 23, 24], and more recently for Symmetric Multi-Processor (SMP) systems [16, 25].

Unfortunately, such support for DSM systems has not yet been studied in detail. Uniprocessor schemes provide data encryption and authentication only for processor-memory communication and the main memory but do not address data protection for processor-processor communication present in multiprocessor systems. Proposals for secure SMP systems include encryption and authentication mechanisms for processor-processor communication, but these mechanisms rely on the assumption that each processor can observe every coherence transaction in the system, which is satisfied due to the single shared bus in the system.

This assumption cannot be made in DSMs, where communication is point- to-point rather than through broadcast mechanisms. As a result, new techniques for DSMs are needed. The first contribution of this paper is an analysis of the security requirements for protecting DSM systems against hardware attacks. The findings of this analysis are that passive/eavesdropping attacks are more likely to be attempted because they are non-intrusive and leave very few (if any) traces. Active attacks that modify coherence messages and alter the behavior of the DSMs are less likely to be attempted, especially if the system is augmented with the ability to detect them. Therefore, we seek to prevent passive attacks from succeeding, and we simply detect and report active attacks. To achieve this, we find that different coherence protocol messages and different parts of a message need to be protected differently: with both encryption and authentication, with authentication only, or with no protection. One possible approach to create a secure DSM is to provide direct encryption and authentication, in which direct encryption (or decryption) and Message Authentication Code (MAC) generation (or verification) are performed for each coherence message sent (or received). However, this approach would directly add cryptographic latencies to the already problematic communication latencies in DSM systems. Therefore, our second contribution is a new combined counter-mode encryption/authentication scheme that hides much of the cryptographic latencies due to protecting processor-processor communication. Our scheme relies on two essential techniques. First, we observe that if communicating processors share the same communication counter, they can pre-generate one-time pads used for message encryption and decryption. Hence, to hide encryption/decryption latencies, we use per-processor pair communication counters that are incremented asynchronously after each message send/receive. Secondly, we also maintain data integrity through the use of GCM, a MAC-based authentication technique using a combined authenticated-encryption mode [3, 14] to reduce the MAC computation latency to only a few cycles after message cipher-text is available. Finally, we also show how our mechanisms can be seamlessly combined with previously proposed processor-memory data protection mechanisms to provide system-wide data protection for DSMs.

II. LITERATURE REVIEW

Architectural support for data privacy and integrity has been studied extensively by researchers for uniprocessor systems [5, 6, 12, 13, 17, 18, 21, 23, 24]. These studies assume that on-chip storage is secure, while off-chip communication is not secure and needs to be protected against passive and active hardware attacks. They provide encryption and authentication for data in the processor memory communication path through direct encryption [6, 12, 13] or through counter mode encryption [17, 21, 23, 24]. In counter mode, instead of directly encrypting the data, encryption is applied to a seed to generate a pad. A seed typically consists of the memory block address and a counter.

To encrypt or decrypt a data block, it is XORed with the pad. When a block needs to be fetched from memory, if its counter is available on chip, pad generation can be overlapped with DRAM access latency. Counter mode encryption's security relies on the uniqueness of the pad/counter each time it is used for encryption (through incrementing the counter on each write back), hence it is often referred to as a one-time pad scheme. To provide data integrity, an authentication mechanism based on a Merkle Tree was proposed [5]. The Merkle Tree maintains an authentication tree whose leaf nodes are data blocks, and the root node is always stored securely on chip. Merkle Trees were proposed as a way to prevent replay attacks in which an attacker replays a previously observed data value and corresponding MAC. Because uniprocessor protection mechanisms only apply to processor-memory communication, researchers have proposed protection schemes for processor-processor communication in bus-based Symmetric Multi-Processor (SMP) systems [16, 25]. The fundamental assumption used for such protection is that each processor can observe every coherence transaction in the system provided naturally through snooping the bus. In this system, each processor maintains a global encryption counter or global pad used for processor- processor communication. On each bus transaction, each processor updates its counter [16], or uses the snooped data to generate a new Cipher Block Chaining (CBC) encryption pad [25]. The pad is used for both encrypting and authenticating processor-processor communication. Unfortunately, neither uniprocessor nor SMP protection schemes can be extended directly to protect DSM systems. Extending direct encryption/authentication for processor-to-processor communication would incur a very high performance overhead due to the added latencies at the sender side for encrypting data and generating MACs, and at the receiver side for decrypting data and verifying the MACs. With a recent hardware implementation showing an AES latency of 37ns and MD5 or SHA-1 over 300ns [11], this approach is either too costly or not feasible.

Alternatively, one may imagine an approach in which uniprocessor counter-mode encryption is directly extended to protect processor-processor communication by treating processor-to-processor data transfer similarly to a processor-to-memory write back. However, this approach is problematic to support due to the need to keep the counters in both the sending and receiving processor coherent. For example, in response to an intervention to a dirty line, a processor flushes the line to the requester, and the flushed line would be encrypted by XORing it with a pad obtained by incrementing the current counter for the block. This increment would trigger invalidation of other cached copies of the same counter. In order for the receiving processor to decrypt the flushed line, it needs to obtain the new counter value for the block. It would do so by sending a read request for the cache block that contains the counter, which eventually appears as an intervention to the sender processor. Hence, the latency for processor-processor communication is effectively doubled (obtain data, then its counter).

In addition, the counter communication needs to be protected against tampering as well, so it requires high-latency authentication. Similar difficulties exist with maintaining coherency among nodes in the Merkle tree. It is also clear that SMP protection cannot be extended easily for protecting DSM systems. The requirement that each processor observes all coherence transactions would be costly to support in terms of ensuring a global ordering of all transactions as well as the large bandwidth requirement needed for broadcasting each transaction to all processors. Our work in this paper differs from previous approaches in that it proposes architectural support for data secrecy and integrity in DSM multiprocessors. It does not rely on maintaining coherence for counters or the authentication tree, and does not require broadcasting of coherence transactions. Finally, while the use of Galois/Counter Mode (GCM) for processor-memory protection in uniprocessor system has been proposed in [23], this paper applies GCM in the different context of processor- processor communication protection. Hence, the input to GCM is very different than that for uniprocessor systems.

III. Security Models

A. Uniprocessor Security Model

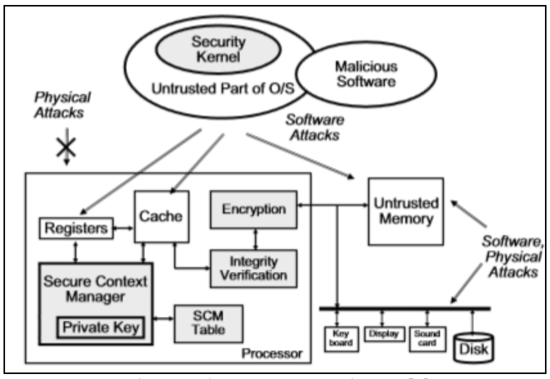

Fig 1: Uniprocessor Security Model

The processor, implemented on a monolithic integrated circuit (IC), is assumed to be trusted and protected from physical attacks; its internal state cannot be tampered with or observed directly by physical means. The processor can contain secret information that identifies it and allows it to communicate securely with the outside world. This information could be a Physical Random Function [27], or the secret part of a public key pair protected by a tamper-sensing environment [28]. The trusted computing base (TCB) consists of the processor chip and optionally1 some core parts of the operating system that plays the part of the Nexus in Palladium [26] or the security kernel in AEGIS [29]. The processor is used in a multitasking environment, which uses virtual memory, and runs mutually mistrusting processes. External memory and peripherals are assumed to be untrusted; they may be observed and tampered with at will by an adversary. The system provides programs with two secure execution environments: tamper evident (TE) and private tamper resistant (PTR). In the TE environment, the integrity of a program's execution is guaranteed. The PTR environment ensures the privacy of instructions and data in addition to integrity. Once a program has entered a secure execution environment using a special instruction, the TCB protects it and provides it with an additional instruction to sign messages with the processor's private key. The resulting signature is used to prove to a user that he is seeing the results of a correct execution of his program. Since the adversary can attack off-chip memory, the processor needs to check that it behaves like valid memory. Memory behaves like valid memory if the value the processor loads from a particular address is the most recent value that it has stored to that address. We therefore require memory integrity verification. The TCB needs to ensure the integrity of memory accesses before it performs a signing operation or stores data into non-private memory space. For PTR environments, we additionally have to encrypt data values stored in off-chip memory. The encryption and decryption of data values can be done by a hardware engine placed between the integrity checker and the off-chip memory bus, as in AEGIS. We assume that programs are well-written and do not leak secrets via memory access patterns. We do not handle security issues caused by bugs in an application program.

B. Multiprocessor Security Model

For multiprocessor shared-memory protection, it is possible to apply uniprocessor security schemes, but cache-to-cache communications need a different protection scheme. Unlike uniprocessor secure computing models, encryption and generation of MAC in multiprocessor systems become time-critical because a receiving processor stalls to wait for a reply. As for authentication of cache-to-cache communications, Shi, et al. proposed an authentication speculation execution to remove MAC latency from the critical path [16]. In this scheme, while the receiver verifies using MAC; it speculatively continues to execute using un-authenticated data. Those executions are committed only after all operands become authenticated. This scheme reduces performance overhead by overlapping authentication and CPU execution, but each processor needs a complex speculation circuit and this scheme is still vulnerable to replay attacks. Zhang, et al. used Cipher Block Chaining (CBC) mode in which the previous MAC is used to make the next MAC, preventing replay attacks [25]. Rogers, et al. pointed out the limitation of above schemes on DSM systems and proposed an efficient data protection design [1]. By focusing on point-to-point communications of the directory-based cache coherence protocol, they were able to utilize DSM systems' temporal locality of communications, which means a processor communicates with a relatively small number of neighboring processors in a short period of time. Such locality makes it possible for each processor to have a small table to hold counters, resulting in good scalability.

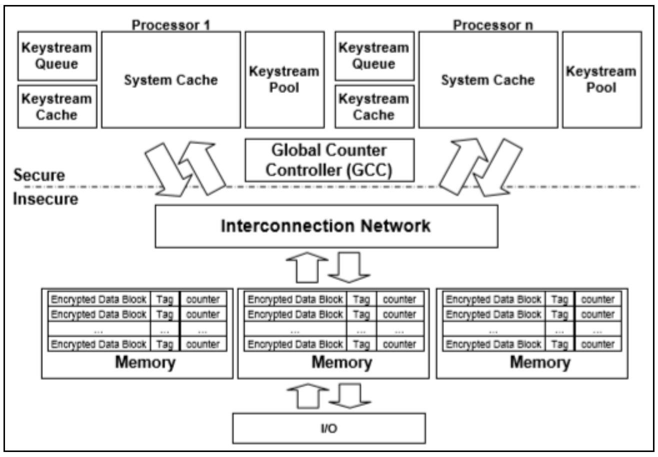

Fig 2: Multiprocessor Security Model

Please note that in multiprocessor shared memory protection, all processors and related components like the memory controller are assumed to share the same secret key. This can be done through the fabrication from factory or runtime distribution as described in [30]. Therefore, even if an ASIC or FPGA is hooked up to the system and pretends to be a peer processor in the multiprocessor systems, it cannot break the privacy and integrity of the system since it is practically impossible for an illegal device to have the same secret key.

C. Signcryption

Signcryption proposed by Zheng [31] at Crypto'97 is a public key or asymmetric cryptographic method that provides simultaneously both message confidentiality and unforgeability at a lower computational and communication overhead than doing signature and public key encryption separately. Recent progress in the security analysis of signcryption indicates that the specific instantiations of signcryption demonstrated in [31] are indeed secure in a very strong sense. More specifically, it has been proven in [32, 33] that these schemes are secure against adaptive chosen ciphertext attacks and existentially unforgeable against adaptive chosen message attacks, both in the random oracle model, relative to Gap Diffie-Hellman and Strong Discrete Logarithm problems respectively.

It should be emphasized that the signcryption schemes could be proven secure without any significant changes of the schemes. However to simplify analysis, [32, 33] modified the original schemes slightly by introducing an extra one-way hashing into the signcryption and unsigncryption operations.

D.Signcryption Algorithm

The signcryption algorithm SC (.) is run by the sender S. The common parameter cp, the sender S's secret key xS, and the receiver R's public key yR and bind info containing the sender and receiver's public keys (yS; yR) are provided as input to this algorithm. We remark that including bind info in the input to the signcryption algorithm was first suggested in [31] and it was shown in [32, 33] that bind info is necessary for the signcryption to be proven secure. As pointed out in [31], bind info could contain strings that uniquely identify the sender S and the receiver R or hash values of the public key of each party. However we assume in this report that bind info contains the concatenation of yS with yR.

A detailed description of SC (cp; m; xS; yR; bind info) is as follows.

Signcryption Algorithm SC (cp; m; xS; yR; bind info)

```
1. Pick x uniformly at random from [1; …. ; q -1]
2. w = yx R mod p
3. K = G(w)
4. r = H(m; bind info;w) where bind info =(yS; yR)
5. s = x=(r + xS) mod q if 'type1' is used,
                    or
   s = x=(1 + xS . r) mod q if 'type2' is used
6. c = EK(m) 7. Return C = (c; r; s)
```

E. Unsigncryption

Now we describe the unsigncryption operation of the signcryptext C = (c; r; s) by the receiver R. Note that the common parameter cp, the receiver R's secret key xS, and the sender S's public key yS and bind info containing the sender and receiver's public keys (yS; yR) are provided as input to the unsigncryption algorithm USC(.). [34]

Unsigncryption Algorithm USC (cp;C; xR; yS; bind info)

```
1. Parse C as (c; r; s)
2. w = (yS . gr)s.xR mod p if 'type1' is used,
                    or
   w = (g . yrS )s.xR mod p if 'type2' is used
3. K = G(w)
4. m = DK(c)
5. If r =2 [0; …; q ¡ 1]
                    or
   s =2 [1; ….; q ¡ 1] then return `Rej (reject)'
6. If r = H(m; bind info;w) then return m. Else output `Rej '.
```

IV. DSM Security Issues

As mentioned earlier, our goal is to protect DSM systems against hardware attacks in the context of on-demand computing. We assume that the system has relatively strong physical security, but is not immune to attacks by a select few employees or other parties who have physical access to the machine. Since it is likely that only a few people have physical access to the machine, any attacks that leave traces may easily provide sufficient information that can lead to the attacker. We define a trace as a detectable anomaly of the system behavior; Hence, the fundamental assumption is that the goal of an attacker is to perform traceless attacks in order to steal sensitive data that belongs to the application. We broadly categorize hardware attacks into three categories. [1]

The first category is sabotage attacks in which the attacker's goal is to crash the application or even damage the system. Our scheme does not seek to protect against sabotage attacks, including application or system crashes, since it is extremely difficult to protect the system against such sabotage when the attacker has physical access to the machine. On the other hand, the attacker lacks the incentive to do so because the attack can be easily traced back to him/her, and there is probably little financial reward for sabotage attacks. Another category is passive attacks in which the attacker's goal is to eavesdrop on processor-processor or processor- memory communication, as illustrated in Fig 3.

An example of this attack is the physical insertion of a snooping device onto the exposed interconnects at the back of server racks. A small USB drive-sized device with multi-GB storage can likely be attached and removed in a matter of seconds without shutting down the system if the system can recover from temporary link failures. Cable clutter may also hide the device from cursory visual inspections.

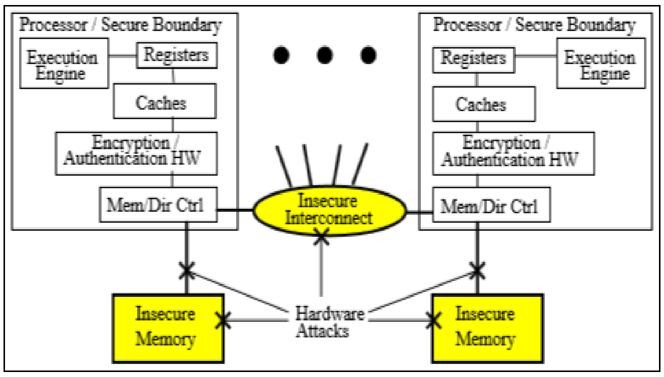

Fig 3: Hardware attacks

Finally, in active attacks, the goal of the attacker is to steal sensitive information by modifying coherence messages communicated between processors, or data in a node's local memory or on the memory bus. Although active attacks are certainly more difficult to perform than passive attacks, we cannot rule out the possibility of an attacker attempting them, especially if passive attacks are no longer fruitful due to the system encrypting all off-chip communication, and if the attack does not result in any traces. A coherence message typically contains message type, memory block address, routing information (source and destination processors), and, for data messages, user data. [1]

We do not make any assumptions as to the specific abilities of attackers to modify signals, so we assume the worst case in which the attacker is able to modify any parts of the message. We distinguish between attacks that modify application data as data spoofing versus ones that modify other information as non-data spoofing. The attacker may also be able to replay an old message. Finally, the attacker may also modify the coherence protocol directory information stored at each node.

V. Research Approach

A. Problem Definition

The emergence of the technology made easy to communicate and share resources with the remote devices in a secure way. Apart from this simultaneously intruders also become strong to break the secure environment to get what they want. There are so many techniques have been proposed by many researchers to secure the DSM systems. Shi, et al, Zhang, et al, Rogers, et al [23, 33, 21] have proposed so many techniques even though they are not strong enough to secure the data. The primary purpose of this research is to provide protection against hardware attacks on data messages in DSM systems.

To achieve this purpose it is necessary to have a technique to encrypt and authenticate data during processor-processor data sharing in a network. Here we are not going to show how to develop the whole process as it is already explained by Shi, et al, Zhang, et al, Rogers, et al [23, 33, 21]. We are going to use the technique of signcryption to encrypt and authenticate data during processor-processor data transfer over network. The signcryption technique is naturally equivalent to both encryption and restrictive authentication of data signatures. The following are the various mechanisms used to encrypt and authenticate data for process-process communication.

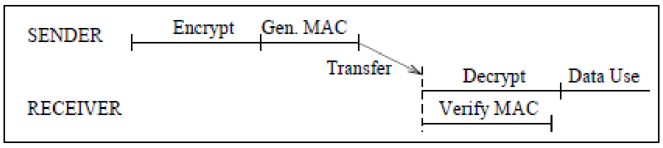

Fig 4: Direct Encryption and Ciphertext-based MAC

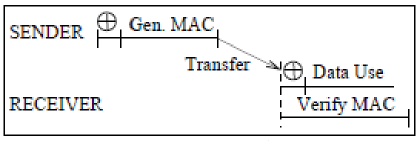

Fig 5: Pre Generated pads for Encryption

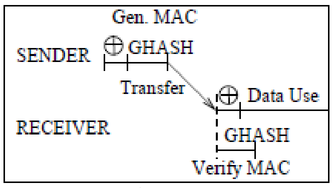

Fig 6: Pre Generated pads for Encryption and Authentication

B. Methodology

In this paper we are going to prove that the technique of signcryption used in DSM systems data protection is the best. The following illustration shows how the protection of DSM systems data is done.

The flow of the techniques is as follows: Initially the sender system a value from the large range of numbers and it will be consider as SK. Then the senders public key PK and the value SK both will be compute hash of it. This will generate a 128-bit string KEY. Then the sender system splits the KEY two equal halves KEY1 and KEY2 as mentioned in the Fig 7. Following that using KEY1 the plain text will be encrypted this will generates ciphertext (C) then the system uses KEY2 for one-way keyed hash function to generate hash message (r), system then computes the value of s. System does this using the value of SK, her private key SK_a, the large prime number q and the value of r. $s = SK / (r + SK_a) \bmod q$. Sender system now has three different values, c, r and s. sender then has to send these three values to Receiver in order to complete the transaction. Sender can do this in a couple of ways. Sender can send them all at one time. Sender can also send them at separately using secure transmission channels, which would increase security. Thus on sender system part, Signcryption of the message is done.

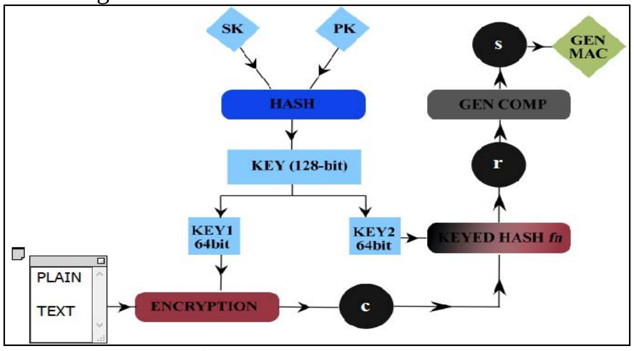

Fig 7: Process of Encrypting and authentication data during processor-processor data transfer over network

Once the receiver receives the values sent by the sender, then receiver uses the values of r and s, his/her private key SK_b, sender’s public key PK and P and G to compute a hash which would give receiver a 128-bit result.

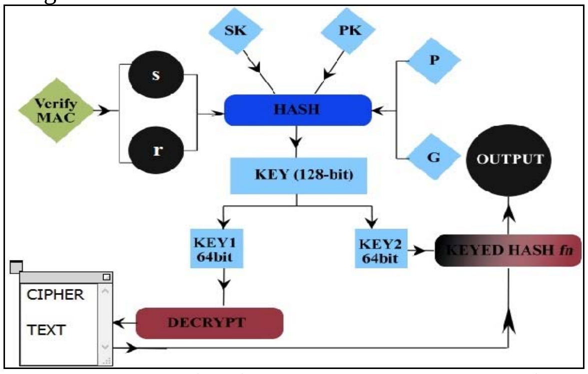

Fig 8: Process of verify authentication and decryption of Data once the data is received by the receiver

This 128-bit hash result is then split into two 64-bit halves which would give him/her a key pair (KEY1, KEY2). This key pair would be identical to the key pair that was generated while signcrypting the message. Now receiver system does a one-way keyed hash function on ciphertext using the key KEY2 and compares the output with the value r receiver received from sender. If they match, it means that the data was indeed signed and sent by sender, if not receiver will know that the message was either not signed by sender or was intercepted and modified by an intruder.

VI. Conclusion

This proposed system is to provide a good, efficient method for providing security for Distributed Shared Memory data from hackers and sent to the destination in a safe manner. Signcryption and Unsigncryption techniques have been used to make the security system more sophisticated and robust.

References

[1] B. Rogers, M. Prvulovic, and Y. Solihin. Efficient data protection for distributed shared memory multiprocessors. In 15th International Conference on Parallel Architecture and Compi- lation Techniques (PACT 2006). ACM, 2006.

[2] R. Anderson. Why cryptosystems fail. In Proceedings of the 1st Conf. Computer and Communications Security (CCS '93), pages 215–227, 1993.

[3] R. K. B. Yang, S. Mishra. A high speed architecture for galois/counter mode of operation (gcm). In Cryptology ePrint Archive: Report 2005/146, 2005.

[4] D. Bartholomew. On Demand Computing–IT On Tap? http://www.industryweek.com/ReadArticle.aspx?ArticleID=10303&SectionID=4,June2005

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

[6] T. Gilmont, J.-D. Legat, and J.-J. Quisquater. Enhanc- ing the Security in the Memory Management Unit. In Proc. of the 25th EuroMicro Conference, 1999.

[7] A. Huang. Hacking the Xbox: An Introduction to Re- verse Engineering. No Starch Press, San Francisco, CA, 2003.

[8] A. B. Huang. The Trusted PC: Skin-Deep Security. IEEE Computer, 35(10):103–105, 2002.

[9] IBM. IBM Power4 System Architecture White Paper. http://www-1.ibm.com/servers/eserver/pseries/hardware/whitepapers/power4.html, 2002.

[10] J. Renau, et al. SESC. http://sesc.sourceforge.net, 2004.

[11] T. Kgil, L. Falk, and T. Mudge. ChipLock: Support for Secure Microarchitectures. In Proceedings of the Work- shop on Architectural Support for Security and Anti- Virus (WASSA), Oct. 2004.

[12] D. Lie, J. Mitchell, C. Thekkath, and M. Horowitz. Specifying and Verifying Hardware for Tamper- Re- sistant Software. In IEEE Symposium on Security and Privacy, 2003.

[13] D. Lie, C. Thekkath, M. Mitchell, P. Lincoln, D. Boneh, J. MItchell, and M. Horowitz. Architectural Support for Copy and Tamper Resistant Software. In Proc. of the 9th International Conference on Architectural Sup- port for Programming Languages and Operating Sys- tems, 2000.

[14] D. A. McGrew and J. Viega. The Galois/Counter Mode of Operation (GCM). http://csrc.nist.gov/ Cryp- toToolkit/modes/proposedmodes/gcm/, 2004.

[15] T. Olavsrud. HP Issues Battle Cry in High-End Unix Server Market. ServerWatch, http://www.serverwatch. com/news/article.php/1399451, 2000.

[16] W. Shi, H.-H. Lee, M. Ghosh, and C. Lu. Architec- tural Support for High Speed Protection of Memory Integrity and Confidentiality in Multiprocessor Systems. In Proceedings of the International Conference on Parallel Architectures and Compilation Techniques, pages 123–134, September 2004.

[17] 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 Archi- tecture, June 2005.

[18] W. Shi, H.-H. Lee, C. Lu, and M. Ghosh. Towards the Issues in Architectural Support for Protection of Software Execution. In Proceedings of the Workshop on Architectureal Support for Security and Anti-virus, pages 1–10, October 2004.

[19] P. Shivakumar and N. P. Jouppi. Cacti 3.0: An inte- grated cache timing, power, and area model. In Tech- nical Report WRL Technical Report 2001/2. Compaq Western Research Laboratory, Aug 2001.

[20] Silicon Graphics, Inc. SGI Altix 3000 Data Sheet. http://www.sgi.com/products/servers/altix, 2004.

[21] G. Suh, D. Clarke, B. Gassend, M. van Dijk, and S. Devadas. Efficient Memory Integrity Verification and Encryption for Secure Processor. In Proc. of the 36th Annual International Symposium on Microarchitecture, 2003.

[22] S. Woo, M. Ohara, E. Torrie, J. Singh, and A. Gupta. The splash-2 programs: characterization and method- ological considerations. In Proceedings of the 22nd International Symposium on Computer Architecture, pages 24–36, 1995.

[23] C. Yan, B. Rogers, D. Englender, Y. Solihin, and M. Prvulovic. Improving cost, performance, and secu- rity of memory encryption and authentication. In Proc. of the International Symposium on Computer Architec- ture, 2006.

[24] J. Yang, Y. Zhang, and L. Gao. Fast Secure Proces- sor for Inhibiting Software Piracy and Tampering. In Proc. of the 36th Annual International Symposium on Microarchitecture, 2003.

[25] Y. Zhang, L. Gao, J. Yang, X. Zhang, and R. Gupta. SENSS: Security Enhancement to Symmetric Shared Memory Multiprocessors. In International Symposium on High-Performance Computer Architecture, February 2005.

[26] A. Carroll, M. Juarez, J. Polk, and T. Leininger. Microsoft "Palladium": A Business Overview. In Microsoft Content Security Business Unit, August 2002.

[27] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas. Silicon Physical Random Functions . In Proceedings of the Computer and Communication Security Conference, May 2002.

[28] S. W. Smith and S. H. Weingart. Building a High- Performance, Programmable Secure Coprocessor. In Com- puter Networks (Special Issue on Computer Network Secu- rity), volume 31, pages 831–860, April 1999.

[29] G. E. Suh, D. Clarke, B. Gassend, M. van Dijk, and S. Devadas. AEGIS: Architecture for tamper-evident and tamper-resistant processing. In Proceedings of the 17th Int'l Conference on Supercomputing, June 2003.

[30] R. B. Lee, P. C. S. Kwan, J. P. McGregor, J. Dwoskin, and Z. Wang. Architecture for protecting critical secrets in micro- processors. In 32nd Annual International Symposium on Computer Architecture (ISCA'05), pages 2–13, 2005.

[31] Yuliang Zheng, "Digital Signcryption or How to Achieve Cost(Signature & Encryption) << Cost(Signature) + Cost(Encryption)", CRYPTO '97 Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology Springer-Verlag London, UK 1997, Page no: 165-179

[32] J. Baek, R. Steinfeld and Y. Zheng: Formal Proofs for the Security of Signcryption, Proceedings of Public Key Cryptography 2002 (PKC 2002), Vol. 2274 of LNCS, Springer- Verlag 2002, pages 80-98.

[33] J. Baek, R. Steinfeld and Y. Zheng: Formal Proofs for the Security of Signcryption, A full version, Submitted to Jornal of Cryptology. A draft is available upon request to the authors.

[34] Joonsang Baek and Yuliang Zheng, "Description of Provably Secure Signcryption Schemes", http://www.signcryption.org/publications/pdffiles/yz-baek-sc-description-02.pdf, Aug 2002.