Journal of Algorithms and Computation

journal homepage: http://jac.ut.ac.ir

Tree Technology for Memory Confidentiality Integrity Protection

HuiMin Meng*[1], HongJin Wang†[2], NianMin Yao‡[3] and ShunYi Cheng§[4]

[1,2] *School of Information Science and Engineering, Dalian Polytechnic University, Dalian, China ,116034.*

[3,4] *Fauclty of Electronic Information Science and Engineering, Dalian University of Technology, Dalian, China, 116024.*

ABSTRACT

PCIP_Tree is a Parallelized memory Confidentiality and Integrity Protection technology (PCIP) method that pre-diffuses and encrypts the data stored in off-chip counter value for confidentiality and integrity protection of data, adding redundant checking data to the protection data block to ensure the integrity and confidentiality of the memory data. Then use the Simple Scalar tool to run 10 processors (SPEC2000) benchmark programs to simulation test. The result shows that the PCIP_Tree method is greatly improved the running efficiency and the efficiency of encryption methods to compare with the Parallelized Encryption and Integrity Checking Engine (PE-ICE) method, at the same time, through the redundant data to check data integrity is better than using complex Hash algorithm to calculate the checking value and reduce system delay, reduce storage overhead chip, and ensure the real-time encryption and checking

Keyword: PCIP_Tree; Storage; Confidentiality; Integrity; Counter Value.

ARTICLE INFO

Article history:
Received 09, February 2018
Received in revised form 04, May 2018
Accepted 26 May 2018
Available online 01, June 2018

*Corresponding author: H. M. Meng. Email: menghm@dlpu.edu.cn
†bala_wanghongjin201@163.com
‡lucos@dlut.edu.cn
§chengsyhlj@sohu.com

AMS subject Classification: 05C79.

1 Introduction

With the rapid growth of digitalized information stored in modern computer system, information security is attracting more and more attention. Business software has been cracked and confidential information has been leaked. The popularity of digital copyright protection, the rise of new computing models, such as cloud computing and large data analysis, makes it more urgent to solve the problem of data security. In most existing computer systems, the data is communicated in plaintext between the processor chip and the other chips. An attacker can get important information or tampering with important hardware and software information through physical hardware attack, such as password, account and other sensitive data. Therefore, protecting the confidentiality and integrity of data and preventing attackers from stealing and modifying key data of the system has become an important direction in the current security field [11].

Complete memory security protection needs to consider the confidentiality and integrity of the data[16].For single-processor architectures,in terms of data confidentiality protection is only to consider the security communication between processors and memory[20].Advisory Experts Group on Inherent Safety(AEGIS)[13] and eXecute-Only Memory(XOM)[21]protection framework were protected memory confidentiality by direct block encryption, the processor can read and write normally after performing the encryption and decryption operation, the whole system of process delay larger. Y Zhang et al. put forward the One-Time Password (OTP) encryption mode in 2003[26]. This method separates the delay of the encryption algorithm from the memory read, reduces the delay on the critical path of the system, and improves the encryption efficiency. Memory integrity protection mainly adopts hash value and Message Authentication Code (MAC) technology, and introduces hash tree protection mechanism in AEGIS framework to prevent replay attack[9]. However, when memory adopts the MAC technology reads and writes data, the two processing operations (encryption operation and digest value calculation) cannot be performed in parallel.

In this paper, PCIP_Tree removes the computational MAC operation by adding redundant verification data to the data and combines the data confidentiality protection and integrity protection through diffusion method and counter encryption method, thereby reducing the delay of compute MAC and improve protection efficiency. Compared with the PE-ICE integrity protection scheme, this method greatly reduces the system delay due to the introduction of protection operation, improves the system efficiency, and has great advantages. PCIP_Tree protection mechanism is used to save Combination of Two Registers (CTR) values off-chip based on the PCIP. Then, we use 10 processors benchmark program (SPEC 2000) to simulation on the SimpleScalar simulator. The results show that PCIP_Tree protection mechanism can reduce the overhead of the system and effectively protect the memory.

2 Memory Protection technology

Memory protection technologies include confidentiality protection technology and integrity protection technology. In order to prevent the leakage of sensitive information and ensure the confidentiality of the memory, the data needs to be encrypted and the data is transmitted or stored off-chip in the form of cipher, so that even if the attackers get the corresponding data cannot understand the specific meaning, and then through the integrity information check the data[15].

2.1 Memory Confidentiality Protection Technology

The main way to encrypt memory data is block encryption and counter encryption. Both XOM and AEGIS protection framework use direct block encryption to encrypt data blocks to protect the confidentiality of the memory. The data is encrypted before being written back to the memory and decrypted when entering the processor. After executing encryption and decryption operation, the processor can perform normal reading and writing operations. The whole system process delay directly affects the operation efficiency of the program. Jun Yang et al. [27, 28]proposed the OTP encryption method, which is called the Counter Mode. The counter encryption method makes the encryption process and memory read parallel, thus reducing the encryption delay and improving the encryption efficiency. However, each data block needs to maintain a corresponding counter value, which increases the maintenance overhead.
Literature[23]uses counter value prediction mechanism to predict several counter values in advance and encrypt them, when the corresponding counter value is returned from the memory and it matches the predicted counter value, we get the plaintext by using the ciphertext corresponding to the counter value with the encrypted data. Literature[24]sets an encrypted prediction engine in the processor to preserve some of data that used frequently and the corresponding encrypted data. When the cache fails, the processor sends a read request to the memory, after obtaining the ciphertext data in memory, compares it with the pre-stored ciphertext in the encryption prediction engine. If the ciphertext matches, the corresponding plaintext is used directly. For these two prediction mechanisms, if the prediction is correct, the delay is reduced, but if the hit rate is low, it will increase the delay.

2.2 Memory Integrity Protection Technology

Memory integrity protection technology is a kind of detection method of active attack, the main protection strategies are cryptographic hash function, Message Authentication Code (MAC) function and block-level Added Redundancy Explicit Authentication(AREA). The method of checking the hash value of the data block of the hash function requires a large storage cost, which is not very practical. In order to reduce the storage overhead, the information needs to be stored off-chip, but the off-chip is not safe and easy to be attacked by the Replay. To ensure that information is not attacked by Replay, the MAC

function is used to calculate the MAC value of a data block. The only value need to store in the on-chip secure area, however, this requires a large storage overhead. AREA technology is writing the block data of added redundant information by block encryption method into memory. If the verification data obtained after decryption will be consistent with the added redundant information, then the data is not stolen, otherwise, the data is tampered with[4, 7].
Massachusetts Institute of Technology put forward a kind of "Lazy" integrity checking technology. On the basis of Log Hash Integrity Checking (LHash), Hierarchical Scheme of Log Hash Integrity Checking(H-LHash) uses partial principle to use L2-Cache to buffer some nodes[22]. Compared with LHash, the performance can be further improved and reduce the time between two checks, but the attack behavior cannot be guaranteed in real time.

3 PE-ICE

PE-ICE was proposed by Reouven Elbaz et al.[5]. The integrity check of this method depends on the completion of the encryption operation, using the AREA method[8], the diffusion feature of the packet encryption algorithm is used to integrate the integrity checking operation into the encryption operation. If one of the ciphertexts is modified, all the bits in the decrypted plain text are affected.
The check flag T of PE-ICE is generated by the counter in CPU. Before the data is encrypted, the flag T for check is added into the plaintext data. After encrypting, the ciphertext C of the indistinguishable $P_L \mid T$ is written back to the memory. When the read operation is performed, the ciphertext C is decrypted, then the corresponding T is extracted from the decrypted data, and comparing with T, if the same, the check is successful, otherwise the data is tampered with. PE-ICE encryption is a direct block encryption method, the integrity check operation is integrated into the encryption operation by using the full diffusion characteristic of packet encryption algorithm, and compared with the Galois Counter Mode(GCM)[10, 17, 18, 25, 29], the system delay is greatly increased.

4 PCIP

Due to the lack of PE-ICE, this paper proposes PCIP to reduce the delay caused by encryption. The counter encryption method is used to replace the direct block encryption method in PE-ICE. At the same time, AREA technology is used to add redundant data to the data and using the data for integrity checking after decryption. In order to ensure the effect of diffusion, the data block ($P_L + Tag$) needs to be diffused before encryption. If an attacker modifies any of these bits, it will affect the data information added to check, so that all kinds of attacks can be detected when checking integrity.

4.1 Generating Check Data

PCIP uses different ways to construct the check values of read-only data and read-write data. For read-only data, it is sensitive to Spoofing attacks and Splicing attacks. There is no problem of Replay attacks. So its corresponding address value can be used as its check data. For read-write data, the content can be modified when the program runs, so that it is also sensitive to replay attacks. If only the address is used as the check value, the processor cannot prove that the data stored at the given address is the latest. In order to ensure that the check value Tag is different at each write operation, it can be implemented in two ways[19]. For the first method, the check value Tag is a one-time only number that can be generated by the counter. In other words, Every time the operation is written, the counter adds 1. Therefore, it is possible to ensure that two ciphertext blocks are used in different Tags. When the counter is overrun, the encryption key must be replaced and all the related memory areas are re-encrypted. However, when the number of bits in the counter is small, the system needs to be re-encrypted frequently. Another method is to avoid re-encryption process, the check value Tag can be used to form a random number. From an attacker's point of view, the check value Tag is random, so it is not possible to predict when the two blocks will have the same Tag. However, the random number cannot be guaranteed to be used only once, which means that there is a possibility of Spoofing attacks and Replay attacks. In order to integrate the security of the system and the storage overhead of the chip, the size of the check value needs to be controlled. The larger the check value, the higher the system resistance to attack, the higher the security, and the greater the storage cost.

4.2 Diffusion Operation

In order to ensure that the ciphertext achieves the effect of full diffusion after being counter-encrypted, the data needs to be diffused before being encrypted. Take a 128-bit data block as an example, consider the data block as a 4×4 two-dimensional array of 8-bit bytes, each byte was treated as an element in GF $(2^8)$and the column vector of 4 bytes were viewed as the number of less than 4 of the polynomial in GF (2^8). The diffusion operation can be achieved by two state columns mixing and one byte transposition.
First, giving three 3rd polynomials:

$$\begin{cases} a(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 \\ b(x) = b_0 + b_1 x + b_2 x^2 + b_3 x^3 \\ c(x) = c_0 + c_1 x + c_2 x^2 + c_3 x^3 \end{cases} \tag{1}$$

Let $a(x)$ and $b(x)$ be two variable polynomials with coefficients a_i and b_i respectively, and $c(x)$ be the given polynomial and multiply the matrix and $c(x)$ in the $mod x^4 + 1$:

$$\begin{aligned} b(x) &= a(x) \times c(x) \\ &= (a_0 + a_1 x + a_2 x^2 + a_3 x^3) \times (c_0 + c_1 x + c_2 x^2 + c_3 x^3) \\ &= (b_0 + b_1 x + b_2 x^2 + b_3 x^3) \times (mod x^4 + 1) \end{aligned} \tag{2}$$

Figure 1: The effect of offset

Calculated the product and distinguished according to the power of x, corresponding to the matrix representation:

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} c_0 & c_3 & c_2 & c_1 \\ c_1 & c_0 & c_3 & c_2 \\ c_2 & c_1 & c_0 & c_3 \\ c_3 & c_2 & c_1 & c_0 \end{bmatrix} \times \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} \tag{3}$$

In order to ensure the data can be restored, it is necessary to ensure the polynomial is invertible, and then obtain the multiplicative inverse. According to Rijndael algorithm [6], the polynomial $c(x)$ is defined as follows:

$$c(x) = 02 + 01x + 01x^2 + 03x^3$$

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} c_0 & c_3 & c_2 & c_1 \\ c_1 & c_0 & c_3 & c_2 \\ c_2 & c_1 & c_0 & c_3 \\ c_3 & c_2 & c_1 & c_0 \end{bmatrix} \times \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} \tag{4}$$

The corresponding inverse of column mixture is transformed by multiplying each column with a fixed polynomial $d(x)$ as follows:

$$\begin{cases} d(x) = 0Bx^3 + 0Dx^2 + 09x + 0E \\ (03x^3 + 01x^2 + 01x + 02 \equiv 1(modx^4 + 1) \end{cases} \tag{5}$$

$$\Rightarrow \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \times \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} \tag{6}$$

After the columns are mixed, the number of each column in the matrix $b(x)$ is related to each number of the corresponding columns in the original matrix to achieve the purpose of the column mixing. Then the matrix is processed by byte transposition, and using different offsets to circular shift the rows in the matrix. To get the best diffusion, we used 4 different offsets for each row. The effect of offset on state is shown in Figure1.
After the offset is completed, the column state is mixed again, and each element in the obtained matrix is related to all the elements in the original matrix to achieve the effect of diffusion.

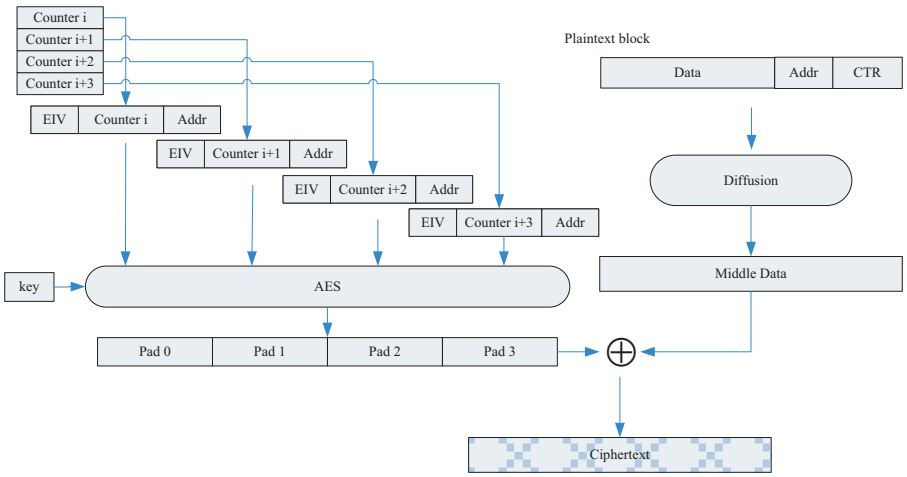


Figure 2: Encryption process

4.3 Data Encryption

In this paper, the counter encryption method is used to encrypt the corresponding intermediate data. The encryption process is shown in Figure 2. First of all, the system generates the counter value of the corresponding data block, after forming the seed, using the packet encryption algorithm to encrypt the seed and obtain the encrypted stream data. At the same time, connecting the data with the check value, we get the intermediate data after the diffusion of the data, and then the intermediate data exclusive or the key stream to obtain the ciphertext. The decryption operation is similar.

4.4 Read/Write Operations

When the processor performs a "read" operation, if the data is in the CPU cache, the processor directly reads the block of data and run the corresponding operation. Otherwise, the processor needs to read the ciphertext data from memory and calculate the corresponding decryption key stream based on the corresponding counter. When the ciphertext is read into the CPU from memory, it is decrypted and restored. Then the check value is extracted and compared with the check value stored in the CPU. If it is matched, the check is passed and the processor performs the corresponding operation on the data. If the check fails, the data is illegally tampered and the processor is thrown out of the exception.

When the processor performs a "write" operation, it needs to change the data block of an address. If the data is in the CPU cache, the data is modified directly, and the new check values are regenerated to replace the old check values. When the system uses a write-through strategy, the data and check values are diffused and encrypted to write to memory. When write-back strategy is used, the data is only written to the memory after the system calls the flush operation and the data block is replaced and cached by the cache replacement algorithm. If the data is not in the CPU cache, the data is

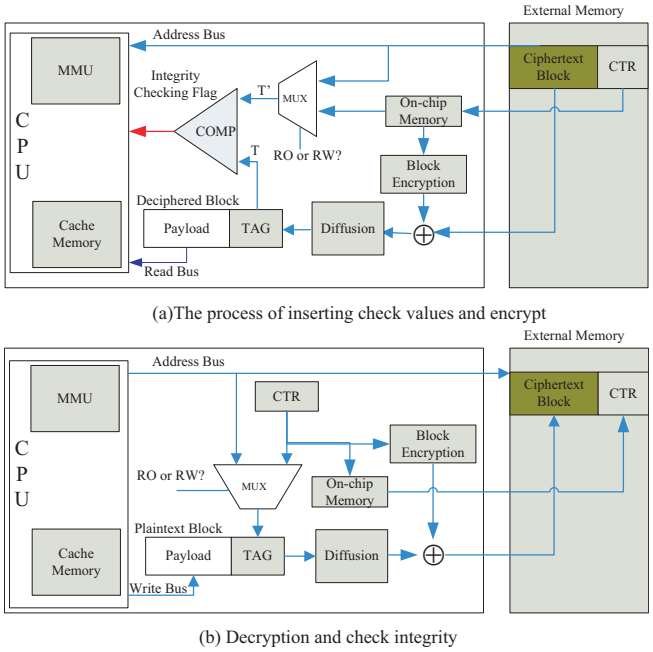


Figure 3: Memory protection system structure

written to the cache, and if the cache is full, then the corresponding block is replaced by the corresponding cache replacement algorithm, and the new check values are generated. Then the two checks are connected and written back to memory after being diffused and encrypted.

5 The Design of the memory protection system

For the security system, the delay in reading and writing memory has the most direct impact on system performance. According to the memory protection idea, the operation of reading and writing memory can be done after the encryption and decryption operations and the integrity check are completed. Therefore, the system should minimize redundant operations and reduce the delay on the path. PCIP needs to use generated check value counter and encryption counter, the combination of the two is represented by CTR, which is used to form the counter encryption seeds, and also to generate the check values of the data blocks, the design of the memory protection system structure shown in Figure 3.
When writing the data block, firstly, according to the data block CTR and address generate the corresponding check value Tag. Then the check value and the effective data block are combined to form a plaintext block, after diffused the data and exclusive or the encrypted data stream, the ciphertext block was generated. Finally, written back to the memory.
When reading a block of memory data, the ciphertext block and its corresponding CTR are read in first. The CTR is used to generate a decrypted stream of data by the encryption

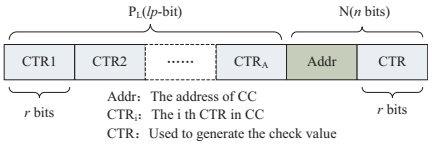


Figure 4: Composition of unencrypted CC blocks

algorithm and exclusive or the ciphertext to obtain the intermediate data, and then the intermediate data is restored to plaintext data block by the corresponding inverse diffusion algorithm. Finally, the check values in the plaintext block are extracted for integrity checking. After the same, the corresponding valid data are read into the CPU, otherwise the integrity check fails.

Since each data block has a corresponding CTR, so a large number of CTR need to be saved. However, the space in the chip is limited, so some of the CTRs need to be stored off-chip. But off-chip storage space is unsafe, and the attacker can tamper with the CTRs to achieve its purpose, so it is necessary to protect the CTRs stored off-chip. In this paper, we design a data block composed of multiple CTRs into a new data block, and use iterative PCIP to protect these CTRs to obtain a tree structure for preventing replay attacks.

5.1 PCIP-Tree STRUCTURE

First of all, the CTR used to check a plurality of consecutive data blocks is merged into a payload data, it recorded as P_L, and then a new check value is generated for the payload data, it is labeled as N, connected them to form a plaintext block, it recorded as CTR Chunk (CC).After the plaintext block P is diffused and encrypted, the ciphertext C is obtained and stored in the memory. The structure of CC is shown in Figure 4.

The size of P_L is l_p, the size of each counter value is r bits. And the number of counter values CTR that can be stored in a CC is:

$$A = \left\lfloor \frac{l_p}{r} \right\rfloor \tag{7}$$

Therefore, storing the CTR of the data block off-chip will reduce the on-chip memory overhead to $1/A$, even so the storage overhead required by the new CTR of saved data is too large for on-chip storage overhead. In order to further reduce the on-chip memory overhead, the newly allocated CTR in the CC is formed into a data block by an iterative method and the corresponding CTR is allocated until only one CTR is stored in the chip, it is called Root_CTR. So we got an A forked tree, and arbitrarily tampering the nodes in the tree will cause the check value extracted from the root node mismatched with the Root_CTR. Therefore, the tree can effectively protect these CTRs, and greatly reducing the storage overhead. Figure 5 shows a 3-layer 4-tree[15].

In the figure, the data block node is taken as a leaf node in the tree, the CC node is an internal node, and each node in the tree has the size of b-bit. The data that needs

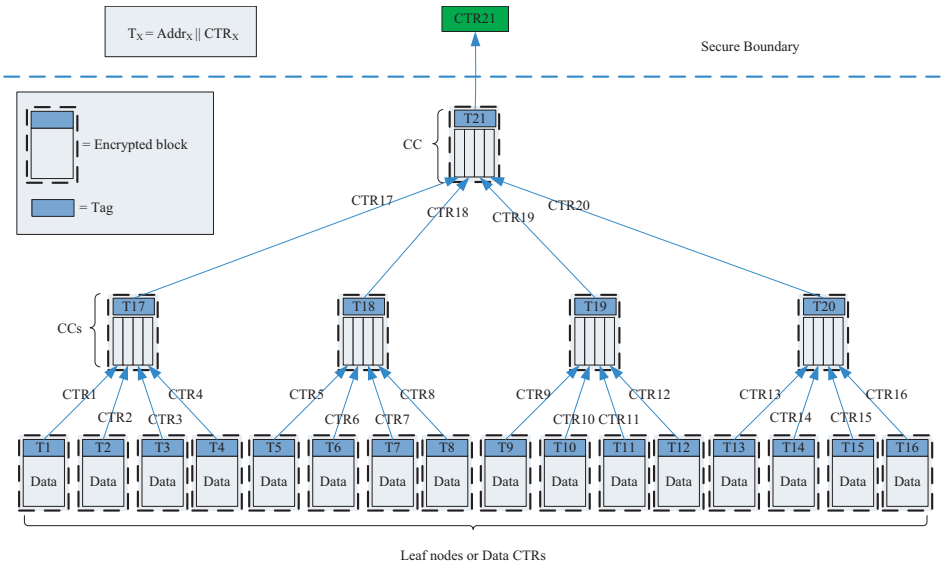


Figure 5: Structure of 3-layer 4-fork tree

to be protected is in the Data CTR (DC) node, and each CC node is used to check the leaf nodes below it, and the Root_CTR that checks the root node is stored in the chip. The arrow in the figure is directed by a node to the CTR that protects the corresponding check value of the node.

5.2 Read / Write Operation

When the processor performs the "read" operation, it needs to read some address data block. If the cache hit, the processor reads the block of data directly from the CPU cache to perform the corresponding operation. If the cache misses, the processor needs to read the corresponding ciphertext data from the memory. At that time, the encrypted data block of address is read into the chip and waiting for decryption and diffusion operation. The parent node is also read in and waiting for the decryption and diffusion operation to get the corresponding CTR of the address data. Until the root node is read, the decryption process is started, and the decryption started from the root node down to the leaf node. When the data is decrypted and diffused, the corresponding check value is extracted and compared with the check value generated by itself. If all nodes pass the integrity check, the processor performs corresponding operations on the data.Otherwise, the data is illegally tampered with and the processor throws an exception. When the cache miss and need to read a data from memory, the number of nodes that need to be checked is:

$$H = log_A(S) + 1 \tag{8}$$

H is also the height of the tree, and S is the total number of data nodes. The whole process of decryption verification proceeds from the root node. Only after the parent node decrypts, can the CTR of the leaf node be got, so that the leaf nodes can be

decrypted.
When the processor executes "write" operations, it is necessary to change the data of an address. If the data is in the CPU cache, CPU directly modifies and generates a new counter value for the data block, updating the corresponding CC node when the block of data is written back to the memory. If the data is not in the CPU cache, the data is read in first and then the write operation is executed. When data blocks are written back to memory, a new CTR is generated. After diffusion and encryption, the data is written to memory and its corresponding CC node is updated until all nodes of the data block node to the root node branch are updated. The data can be updated after all the data of the nodes loaded in the chip pass the integrity check.

5.3 Storage Overhead and Performance

The storage overhead of the PCIP protection mechanism is divided into two parts, one is the cost for saving the CC node, and the other is the overhead for saving the check value Tag in the data block [15]. The cost of building an A fork tree on leaf nodes is $1/(A-1)$,and the cost ratio of inserting the check value in the data is n/l_p(n is the bit of Tag,l_p is the number of bits to load).So the total overhead of O is:

$$O = \frac{1}{A-1}\left(1+\frac{n}{l_p}\right)+\frac{n}{l_p} = \frac{l_p+nA}{l_p(A-1)} \tag{9}$$

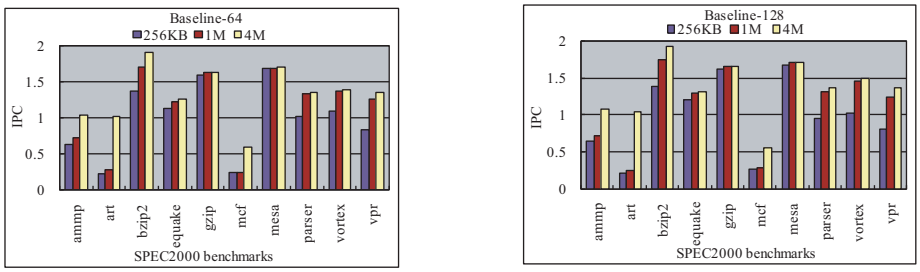
When the processor performs the read operation, the node decrypts from the root node down to the leaf node in turn, and the process cannot be carried out in parallel. While updating the data, the parent can immediately know the new CTR corresponding to the node when it is updated to generate a new CTR. Therefore, when the data node is updated, the CC node can update the new CTR using the leaf node accordingly. All nodes from the node to the root node can be updated at the same time when the data node is updated. So the method has the characteristics of being updated in parallel, it is suitable for the program with more frequent updates.

6 SimpleScalar simulation experiment and result analysis

In this paper, we use Simple Scalar [1, 2, 3, 12] to simulate the proposed scheme. The simulation program uses 10 SPEC2000 benchmark programs (ammp, art, bzip2, equake, gzip, mcf, mesa, parser, vortex, vpr, etc.) to evaluate the performance of the proposed scheme.

6.1 Benchmarks

To evaluate the performance of the 10 benchmarks without using any protection mechanism, when the cache line sizes are 64 bytes and 128 bytes respectively and the L2 cache



(a) Cache Line=64B (b) Cache Line=128B

Figure 6: Benchmark test

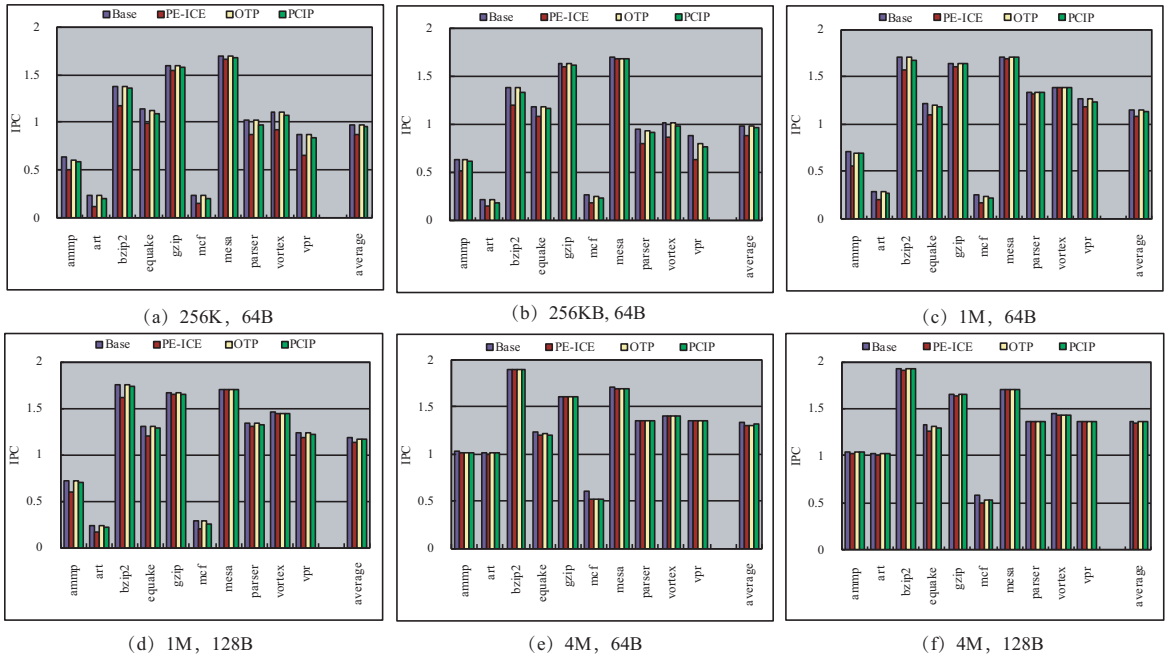
capacities are 256KB, 1MB and 4MB respectively, the Instruction-Per-Cycle(IPC) result of each test procedure is shown in Figure 6.
Figure 6 shows that different cache capacity and different blocks have different effects on different test processes. This is mainly due to the inconsistent read-write features and local characteristics of different programs. The larger the L2 Cache capacity, the higher the Cache hit rate and the higher the corresponding IPC.

6.2 The influence of encryption on program performance

The experiment uses the 64KB-sized on-chip security CTR_Cache to store the corresponding counter, and compares the PCIP diffusion plus counter encryption method with the direct block encryption mode adopted by PE-ICE. The result is shown in Figure 7.The four schemes in the figure include Base–unprotected benchmark, PE-ICE–direct block encryption, OTP–counter encryption and PCIP–diffusion plus counter encryption scheme. The performance of the proposed scheme is between PE-ICE and OTP, especially when the L2 Cache capacity is small, which has a greater advantage than PE-ICE. As the cache capacity increases, the cache hit rate continues to increase, so that the impact of encryption on program performance is gradually reduced. When the L2 cache is increased to 4MB, the impact of each encryption scheme on most program performance is the same.

6.3 The influence of tree mechanism on the program

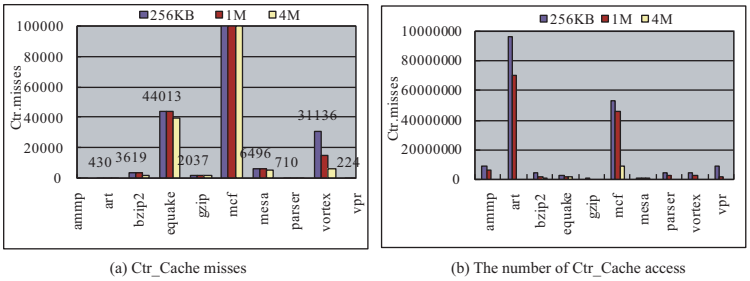
Since the Ctr_Cache used in PCIP is not enough to save the CTR of all data blocks, PCIP_Tree can be used to protect the CTR stored off-chip. The PCIP_Tree protection scheme proposed in this paper reads the CTR only when the data block is not cached. Whether the CTR is hit directly relates to the delay caused by the check. If the cache hit, there is no need to read the CTR from the memory and do not need to check the CTR. Otherwise, read the CTR from the memory and check according to the tree protection scheme. In the different L2 Cache capacity, the access to Ctr_Cache due to the data block cache miss in the PCIP scheme is shown in Figure 8.
The PCIP_Tree scheme is simulated under different L2 cache capacities and compared with the PCIP without the tree protection mechanism. The experimental results are



(a) 256K, 64B (b) 256KB, 64B (c) 1M, 64B

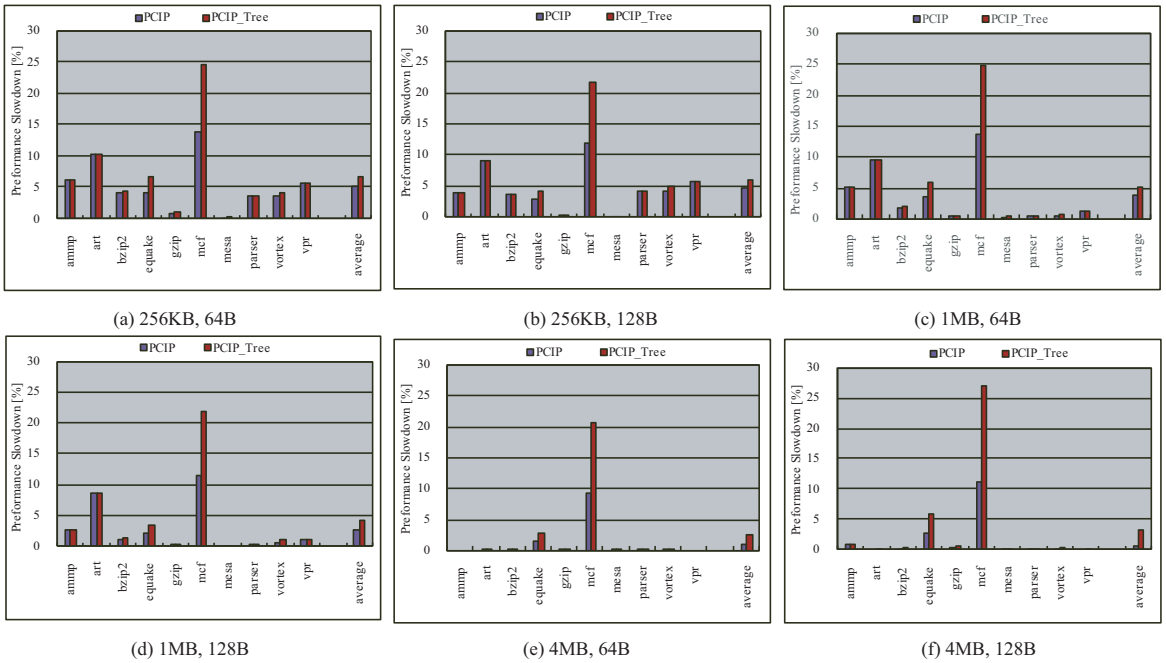
(d) 1M, 128B (e) 4M, 64B (f) 4M, 128B

Figure 7: IPC of different L2 Cache sizes at different encryption schemes



(a) Ctr_Cache misses (b) The number of Ctr_Cache access

Figure 8: Accessing Ctr_Cache with different L2 Cache capacities



(a) 256KB, 64B (b) 256KB, 128B (c) 1MB, 64B

(d) 1MB, 128B (e) 4MB, 64B (f) 4MB, 128B

Figure 9: The impact of a complete protection scheme on the performance of the program

shown in Figure 9. Figure 9 shows that The PCIP_Tree protection mechanism has a lower impact on program performance than the absence of a tree mechanism, which is mainly due to the higher CTR hit rate in Ctr_Cache, and generating it less delay due to the tree protection mechanism when CTR is not in the cache. Some programs have greatly reduced performance after adopting the tree protection mechanism, such as mcf, equake, etc., because their CTR hit rate is low, the number of Ctr.misses is large. At the same time, the overhead of integrity check sharply increased when CTR cache misses.

7 Conclusion

Because of the characteristics of the existing computer system, the memory is easily attacked by various physical attacks. For different physical attacks, the protection of memory can be divided into confidentiality protection and integrity protection. This paper summarizes the existing memory technologies of confidentiality protection and integrity protection firstly, and then combined memory confidentiality and integrity protection schemes for analysis. On the basis of PE-ICE, using AREA technology proposed PCIP to encrypt first and then encrypt to protect the confidentiality and integrity of the data at the same time. After diffusing the data and the check value, counter encryption is used to encrypt. On the one hand, compared to the PE-ICE method, the scheme has a great

improvement in the performance of encryption. As an example of the size of 256KB L2 Cache, the impact of the protection mechanism on program performance decreased from 17.52% of PE-ICE to 5.25%. On the other hand, compared to the simple OTP approach, PCIP provides confidentiality and integrity protection for memory with minimal overhead. Finally, based on this method, tree protection mechanism is used to save CTR to the off-chip, greatly reducing the on-chip overhead and effectively protecting the memory. The advantage of PCIP_Tree is that the process of updating the tree can be carried out in parallel, and the verification process needs to start from the root node and down to the corresponding leaf node in turn. The method mainly considers memory protection under the single processor,for the protection of multi-processor memory it has a big difference and more complex even though under the similar technology, such as SMP, DSM, etc. The next step is to consider how to extend the proposed scheme to the multiprocessor environment.

References

[1] Austin, T., Ernst, D., Larson, E. and Weaver, C., SimpleScalar Tutorial (for release 4.0), The 34th Annual International Symposium on Microarchitecture, In Proceedings(2001), pp.1-3

[2] Austin, T., Larson, E.,and Dan, E., Simplescalar: an infrastructure for computer system modeling, Computer Society,International Conference on (2002), IEEE, 2002, 35(2), pp. 59-67.

[3] Burger, D. and Austin T. M.,The SimpleScalar Tool Set,Version 2.0,Acm Sigarch Computer Architecture News,1997, 25(3), pp.13-25.

[4] Elbaz, R., Hardware Mechanisms for Secured Processor Memory Transactions in Embedded Systems.PhD Thesis, University of Montpellier, 12(2006), pp.13-27.

[5] Elbaz, R., Torres, L., Sassatelli, G., Guillemin, P., Bardouillet, M. and Martinez, A., A Parallelized Way to Provide Data Encryption and Integrity Checking on a Processor-Memory Bus.Proceedings of the 43rd Design Automation Conference(DAC),(2006), ACM, pp.506-509.

[6] Elbaz, R., Champagne, D., Lee, R. B., Torres, L., Sassatelli, G. and Guillemin, P. TEC-Tree:A Low-Cost,Parallelizable Tree for Efficient Defense Against Memory Replay Attacks, Cryptographic Hardware and embedded systems(CHES),International Workshop on (2007), IEEE, pp.289-302.

[7] Elbaz, R., Champagne, D., Gebotys, C., Lee, R.B., Potlapally, N. and Torres, L., Hardware Mechanisms for Memory Authentication:A Survey of Existing Techniques and Engines, Transactions on Computational Science IV, Springer Berlin Heidelberg, 2009, pp.1-22.

[8] Elbaz, R., Torres, L., Sassatelli, G., Guillemin, P., Bardouillet, M. and Martinez, A., Block-level added redundancy explicit authentication for parallelized encryption and integrity checking of processor-memory transactions, Transactions on Computational Science X,10(2011),231-260.

[9] Gassend, B., Suh, G. E., Clarke, D., Dijk, M. V. and Devadas, S., Caches and hash trees for efficient memory integrity verification.High Performance Computer Architecture, International Symposium on (2003), IEEE, pp. 295-295.

[10] Gaborit, P., Ruatta, O., Schrek, J. and Zmor, G., New Results for Rank-Based Cryptography, Progress in Cryptology-AFRICACRYPT 2014.Springer International Publishing, 2016, pp.1-12.

[11] Huang, A. B., Hacking the Xbox: An Introduction to Reverse Engineering, 2nd ed. No Starch Press, 2003.

[12] Kalaitzidis, K., Dimitriou, G., Stamoulis, G. and Dossis, M., Performance and power simulation of a functional-unit-network processor with simplescalar and wattch.Panhellenic Conference on Informatics (2015), ACM, pp. 71-76.

[13] Lie, D., Thekkath, C., Mitchell, M., Lincoln, P., Dan, B and Mitchell, J., Architectural support for copy and tamper resistant software, Acm Sigplan Notices 35, 11(2000), 168-177.

[14] Lee R. B., Kwan P. C. S., Mcgregor J. P., Dwoskin, J. and Wang, Z. Architecture for Protecting Critical Secrets in Microprocessors.International Symposium on Computer Architecture,ISCA '05. Proceedings(2005), IEEE, pp. 2-13.

[15] Lehman, T. S., Hilton, A. D., Lee, B. C., PoisonIvy: Safe speculation for secure memory. Computer Society, International Symposium on Microarchitecture, (2016), IEEE, pp. 1-13.

[16] Maude, T and Maude, D., Hardware protection against software piracy, Communications of the ACM, Volume 27, No 9, (1984), pp. 950-959.

[17] Mcgrew, D. A., The Galois/Counter Mode of Operation (GCM),3rd ed. Submission to NIST Modes of Operation Process, 2005.

[18] Mcgrew, D., Efficient authentication of large,dynamic data sets using Galois/counter mode (GCM).Computer Society, International Security in Storage Workshop(2005), IEEE, pp.89-94.

[19] Ma, H., Yao, N., Cai, S. and Han, Q. Memory Confidentiality and Integrity Protection Method Based on Variable Length Counter.Computer Society,International Symposium on Distributed Computing and Applications To Business,Engineering and Science(2012),IEEE,pp.290-294.

[20] Rogers, B., Yan, C., Chhabra, S., Prvulovic, M., Single-level integrity and confidentiality protection for distributed shared memory multiprocessors, High PERFORMANCE Computer Architecture, International Symposium on (2008), IEEE, pp.161-172.

[21] Suh, G. E., Clarke, D., Gassend, B., Dijk, M. V.and Devadas, S., AEGIS:architecture for tamper-evident and tamper-resistant processing, In Proc,International Conference on Supercomputing(2003), IEEE, pp.160-171.

[22] Suh, G. E., Clarke, D., Gassend, B., Dijk, M. V.and Devadas, S., Efficient Memory Integrity Verification and Encryption for Secure Processors.International Symposium on Microarchitecture,2003.Micro-36.Proceedings(2003),IEEE,pp.339-350.

[23] Shi, W., Lee, H. H. S., Ghosh, M., and Lu, C., High efficiency counter mode security architecture via prediction and precomputation.Computer Society, International Symposium on Computer Architecture (2005), IEEE, pp.14-24.

[24] Shi, W. and Lee, H. H. S., Accelerating memory decryption and authentication with frequent value prediction.Computing Frontiers,International Conference on(2008),IEEE,pp.35-46.

[25] Tjuawinata, I., Huang, T. and Wu, H., Cryptanalysis of the Authenticated Encryption Algorithm COFFE.International Conference on Selected Areas in Cryptography(2015), IEEE, pp.510-526.

[26] Yang, J., Zhang, Y. and Gao, L., Fast Secure Processor for Inhibiting Software Piracy and Tampering. In Proc Micro-36 Proceedings, International Symposium on Microarchitecture, (2003), IEEE/ACM, pp.351.

[27] Yang,J.,Gao,L.,and Zhang,Y.Improving memory encryption performance in secure processors.Transactions on Computers,International Conference on(2005), IEEE, pp. 630-640.

[28] Yan, C., Rogers, B., Englender, D. and Solihin, D., Improving Cost, Performance, and Security of Memory Encryption and Authentication.Computer Society, International Symposium on Computer Architecture (2006), IEEE, pp. 179-190.

[29] Zhu, B., Tan, Y. and Gong, G., Revisiting MAC Forgeries,Weak Keys and Provable Security of Galois/Counter Mode of Operation. Cryptology and Network Security.Springer International Publishing,2013.