Using a Temporization Technique for Page Encryption on Secure Processors

Osvaldo Espinosa Sosa, Luis Villa Vargas and Oscar Camacho Nieto

Center for Computing Research CIC-IPN, Av. Juan de Dios Batíz, esq. Miguel Othón de Mendizábal, México, D.F., 07738. México
espinosa@cic.ipn.mx , lvilla@cic.ipn.mx, oscarc@cic.ipn.mx
Phone: +(55)57296000 ext. 56519

Abstract. At the present days, there are many people making great efforts to improve security in computer systems with the aim to reduce problems caused by hardware attacks, viruses and intruders as well as piracy of software. Several techniques have been proposed in order to encrypt information allocated in main memory in a way that only the microprocessor can access such information through encryption and decryption processes. This work proposes utilization of a system with temporization to encrypt and decrypt pages of main memory at regular intervals of time, in this way every page encryption permits key change with the main objective of improve security in the system. The main idea is that security system causes the minor possible degradation in microprocessor performance. In this paper is analyzed the effect on performance of an encryption system in a superescalar microprocessor. This work shows that making an adequate selection of period of time and memory page size can be incremented the security level in a high grade without affecting performance beyond 10% compared with the reference architecture. Two types of reference systems are studied: the first one using Direct Encryption Mode and a second one using Counter Mode.

1 Introduction

At the present time is evident that software industry is having billionaire loses due to illegal duplication of application programs, that is commonly called piracy [1]. At the same time, new attacks constantly appears and are produced by malicious software that take advantage of operating systems vulnerabilities and hardware weaknesses in computing systems, especially memory system. In order to reduce these problems, several techniques have appeared at microprocessor level [3]. In these techniques the microprocessor is the only entity authorized to access to information, any other hardware component is considered vulnerable to the attacks due to the fact that anybody can be monitoring the information flowing through the buses [4]. Programs are then stored in memory in an encrypted way and only can be decrypted on chip, taking into account that actual processors contain the first two levels of cache memory in the same silicon die that the microprocessor, this means in the same integrated circuit.

A. Gelbukh, S. Suárez, H. Calvo (Eds.)
Advances in Computer Science and Engineering
Research in Copmuting Science 29, 2007, pp. 304-312

Received 10/07/07
Accepted 19/10/07
Final Version 24/10/07

The intention of this paper is to show the effect in the performance of a superescalar microprocessor due to the inclusion of an encryption and decryption system including the capacity to encrypt pages of main memory at regular intervals of time, changing on every page encryption the key to be utilized with the purpose of increase the system security. It is clear that this encryption system will add more latency to the main memory access and this will affect the overall processor performance. It is important to find a trade off between the period of time and the size of memory pages to be encrypted in such a way that performance will not be affected severely and it can offer an adequate security level. The paper contains the methodology used to evaluate the proposal included in this work which uses an execution driven simulator to obtain detailed statistics of realized experiments and finally we have conclusions and bibliographic references.

2 The Memory Encryption Process

On previous work it has been proposed to encrypt data contained in main memory to offer a good security level against attacks [2]. The encryption and decryption system is usually inserted between level 2 of memory cache and main memory due to it is the place where processor performance is degraded in a minor quantity and because of the fact that levels one and two of cache memory normally reside on chip, it serve as a bus interface with an insecure external world. When the processor performs a read operation to main memory, the data obtained must be decrypted to be used by the processor; likewise, when a write operation to main memory is performed the data must be encrypted before.

There are two approaches to do this: the first one is called *Direct Encryption Mode* where the encryption/decryption engine is placed serially between main memory and the second level cache. It encrypts and decrypts data moving between level 2 of cache and main memory. This encryption mode has the characteristic of exhibit the whole latency of encryption system and then an access to main memory increments its normal latency adding the encryption engine latency resulting in a higher total latency. The second approach is called *Counter Mode.* Unlike Direct Encryption system it does not need to wait until data arrives from memory (it does not work serially), instead it encrypts information known yet at the memory access moment such as the address and/or a counter value and encryption can be done in parallel with memory access. Once data and encrypted information are obtained then these are XOR'ed to produce something called data pad and the encryption engine latency is hidden with the memory access latency. Both cases are shown in fig. 1. We can notice the minor total latency of Counter Encryption Mode.

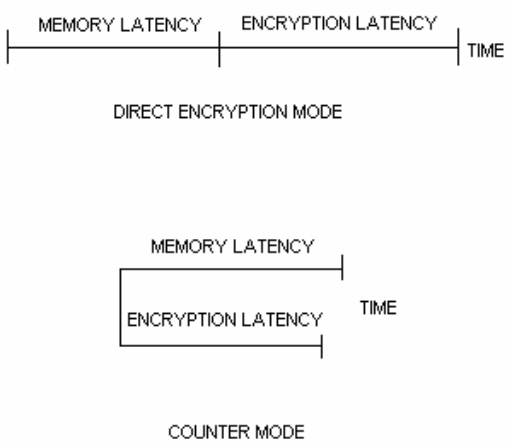

Fig 1. Two approaches for encryption/decryption

For encryption/decryption is usually utilized the AES algorithm (Advanced Encryption Standard) which is an algorithm of fixed parameters. The AES requires as input a data block of 16 bytes in order to encrypt a cache memory line of 64 Bytes, four AES blocks are required as shown in figure 2. The encryption or decryption process can use always the same key, which represent a vulnerability due to the possibility that algorithm can be broken by an expert intruder. Other option is to change the key after certain number of encryptions and decryptions, however when the key is replaced the main memory must be re-encrypted causing that system could be stopped a large interval of time (in the order of seconds) in a system working at frequencies in the order of Ghz. To improve security in a computer system without important performance degradation we propose a system where periodically keys are replaced using a pull of keys, even more we can use different keys for each memory page, as we are going to explain in the next section.

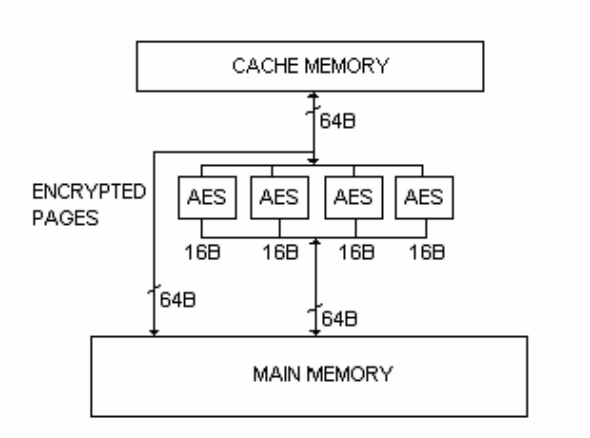

Fig. 2. Encryption and decryption circuit detail.

3 Proposed Architecture

The encryption system is shown in figure 3. The main memory is divided in pages of fixed size. There is a *timer* that activates the key replacing mechanism periodically at regular intervals of time. Keys are generated in a random form. Every time this mechanism is activated a memory page is decrypted using the old key and re-encrypted using the new key. Afterwards, the encrypted information is sending back data to main memory.

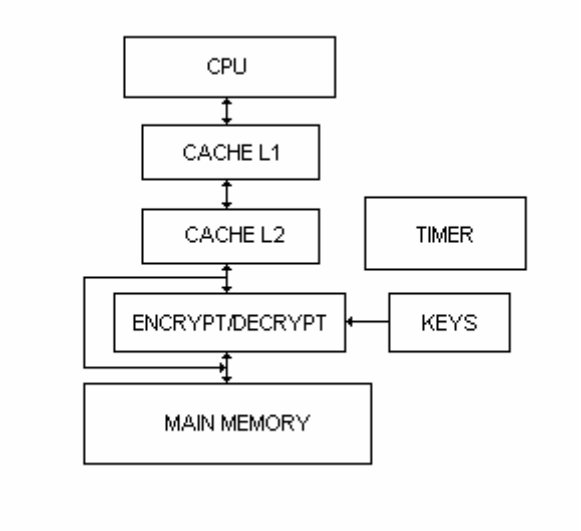

Fig. 3. Proposed architecture.

The next time (after a period of time) a new page will be encrypted using the same process and following a Round Robin scheme to select the page in main memory using a different key and in this way offer a higher level of security in comparison with proposals that use only one key. Using this new page encryption model, the security level increases without an important performance degradation. In order to know what memory pages are active on memory, the processor has a special group of page registers working together with operating system which is the responsible of resource management (memory in this case). The processor includes a set of special registers to store old keys because of the fact that these are used to decrypt memory pages before it use the new one to encrypt data and send it back to memory.

It is important to mention that when a memory page is re-encrypted, information pass through the encryption circuit and then data is given back to main memory without affecting cache memory. This process is mainly affected by memory page reads because data must be read before it can be used unlike memory page write accesses which are more flexible in the time of execution of an operation.

4 Methodology

We have evaluated our model using the simplescalar 3.0 simulator tool set which performs a very detailed simulation of a superescalar processor with out of order execution [5]. The simulator was configured as an Alpha 21264 because this architecture has been considered the best superscalar processor at its time of appearance. This processor contains two level 1 cache memories (Instructions and data) of 64 KB 2-way associative with 64 byte blocks. The second level of cache is unified with size of 1 MB being 8-way associative with 64 byte block. The simulator has as input a set of application programs called benchmarks; we used especially the SPEC CPU 2000 suite which is composed of twelve applications of fixed point and fourteen floating point programs. In this work the performance is monitored whenever we change the page size or the period of time for page re-encryptions. Simulations was made executing 5 X 10^8 instructions skipping the first 1 X 10^8 instructions with the aim of eliminate initialization effects on statistics. Results are shown in terms of IPC average of the 26 SPEC CPU programs.

5 Evaluation

We can take a look first at the case with Direct Encryption Mode. As we can see on fig. 4 the average performance for all applications decrease when we add the page encryption scheme in relation to the system that only uses encryption for normal read and write accesses (this latter case corresponds to 100% on performance, this is our baseline model). The performance reduction is higher if we reduce the interval of time for page re-encryption (number of cycles between page re-encryptions).

If the page size is constant (4 KB) and considering that microprocessor frequency is 1 Ghz, the figure 4 shows that best results are obtained for configurations of 1 X 10^6 cycles (1 ms) and 1 X 10^5 cycles (0.1 ms) between re-encryptions, in both cases performance is degraded only 0.7% and 6.95% respectively. In the previous experiments we consider acceptable a result if performance is not degraded beyond 10%. Using these two best configurations, we vary the page size to see the effect on the average performance.

Figure 5 shows the average results for configurations of pages of 4 KB, 8 KB, 16 KB y 32 KB with an interval of 100,000 cycles between page re-encryptions. We can notice that pages of 4 KB are suitable because performance is diminished only 6.95% and increasing page size the performance is degraded in a major form, in this case re-encryption of pages is performing 10,000 times in a second with the implication of an important improve on security and an acceptable reduction in performance.

It is shown on fig. 6 the result of use a period of time equivalent to 1,000,000 cycles where we can see an improvement on performance respect to the previous case in which it is possible to increase the page size with a minor reduction on performance.

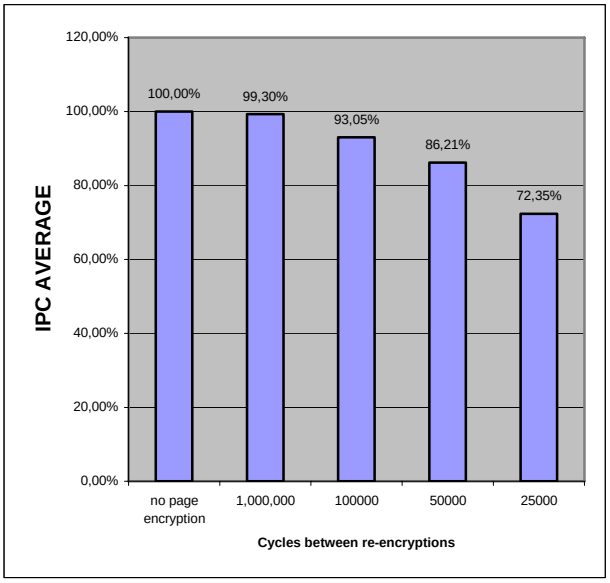

Fig. 4. Page encryption at different periods

It is shown that pages could be up to 32 KB without degrading performance beyond 6% of the average performance.

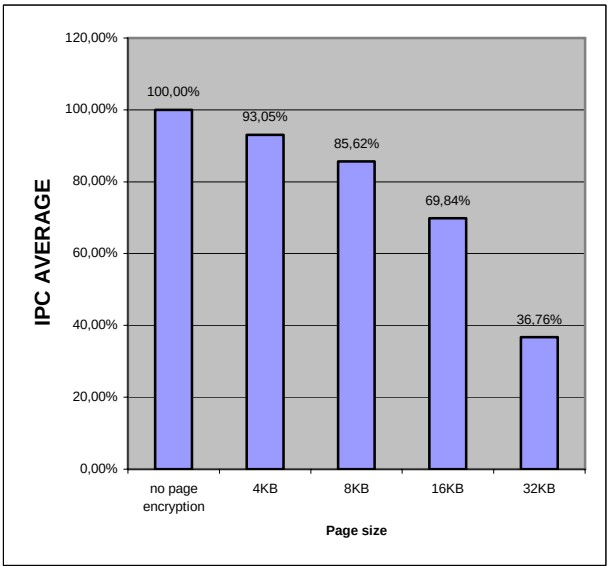

Fig. 5. Effect of page size (100,000 cycles of period)

It must be considered that in the previous case only 1000 pages will be re-encrypted but in contrast this scheme permits to reduce the total quantity of keys to be

used by the system 8 times (if we use pages of 32KB). Now, we can analyze the case for Counter Mode Encryption. Results obtained for a constant 4 KB pages and different periods of time are shown on fig. 7.

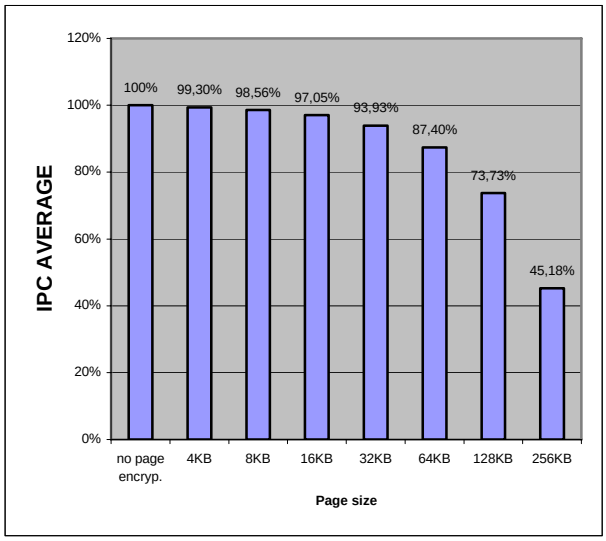

Fig. 6. Effect of page size (1000000 cycles of period)

Results can show that with 4 KB pages in Counter Mode Encryption inclusive with periods of 50,000 cycles, results are tolerable, lower than 10% on average in performance degradation.

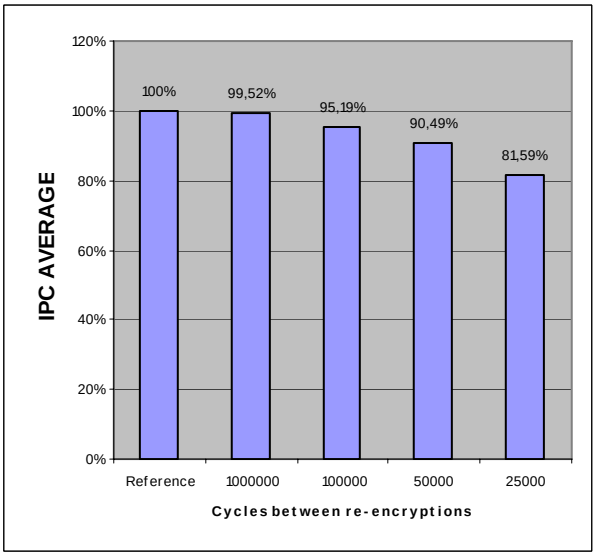

Fig. 7. Page encryption at different periods

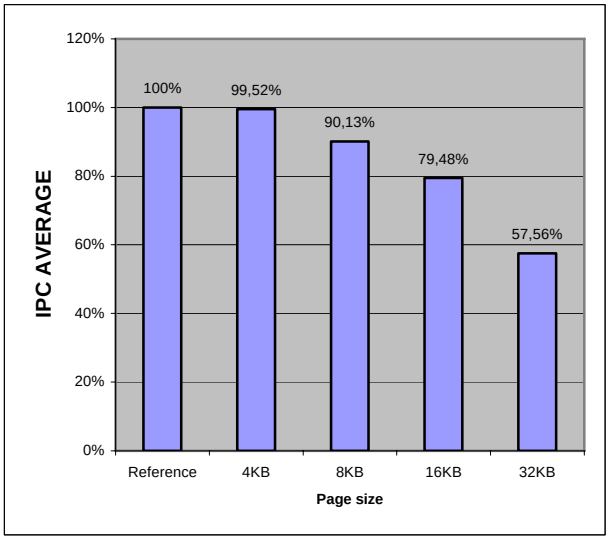

Fig. 8. Effect of page size (100,000 cycles of period)

The fact is due to a minor memory latency that experiments the system with this encryption mode and then it supports more page re-encryptions. Although the average in performance losses is acceptable for three cases: 1000000, 100000 and 50000 cycles between re-encryptions, we can take into account that for 50000 cycles there are some particular programs which experiments losses of 15% on performance. Compared with Direct Encryption Mode this system could tolerate memory latency better.

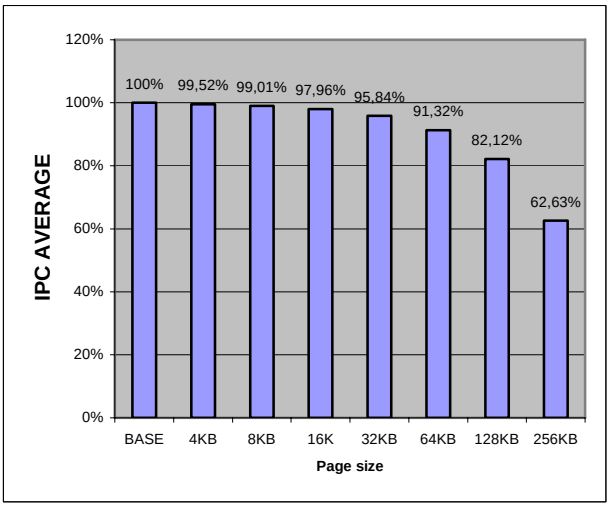

Fig. 9. Effect of page size (1,000,000 cycles of period)

If we take the period of 100,000 cycles as a constant and we vary the page size, it is clear that we can grow up the page size up to 8 KB with acceptable results and it is important to notice the fact that in that case are necessary a minor number of keys (two times) with page re-encryption every 0.1 ms. Fig. 8 contains the exposed information in a graphic form. Again, data obtained with periods of 1000000 cycles and pages of 4 KB indicate a better system performance as it is shown in fig. 9. We can notice that page size could be increased up to 64 KB having a great performance and reducing the number of keys sixteen times (pages of 64 KB). In the sane manner, it is important to see that the greater the number of keys the greater is the security but more resources are needed and performance is affected. In contrast, the greater the page size the lower is the security.

6 Conclusions

In this work it was studied the effect of the inclusion of page encryption using a temporized scheme. It was shown that it is possible to increase the security level of processors when they include capacities of change encryption keys with periodicity and even better using one key for every page instead of one unique key for the whole memory. All without affect in an important manner the processor performance. The small loss of performance is well compensated with the level of security gained. As we can see with the results of experiments, it is important to consider the period of time used by page re-encryption as well as the page size in order to obtain an adequate trade-off between security and performance. Results obtained show us that Counter Mode Encryption has a lower effect on the system but maybe it could be more easily to be attacked that Direct Encryption Mode. The reason of this is that one part of encrypted data (such as memory address can be observed on buses) could be known by an intruder. As a counterpart Direct Encryption Mode offers better levels of security with a more performance impact due to higher memory latency.

References

1. Yang,Zhang,Gao. Fast secure processorfor inhibiting software piracy and tampering. Proceedings of the 36th International Symposium on Microarchitecture MICRO 36-2003.
2. Ruby B. Lee, Peter C. S. Kwan. Architecture for protecting critical secrets in microprocessors. Proceedings of the 32nd Annual International Symposium on Computer Architecture 2005.
3. T.Kgil, L.Falk and T. Mudge. ChipLock: support for secure microarchitecures. Workshop on architectural support for security and anti-virus, 2004.
4. Chenyu Yan, Brian Rogers et. Al. Improving cost, performance and security of memory encryption and authentication. International Symposium on computer architecture ISCA 2006.
5. Burger,Dough. The simplescalar toolset, version 3.0 *Computer Architecture News*, **25** (3), pp. 13-25, June, 1997.