International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622
National Level Technical Symposium On Emerging Trends in Engineering & Sciences
(NLTSETE&S- 13th & 14th March 2015)

RESEARCH ARTICLE OPEN ACCESS

Vlsi Modelling Of Side Channel Attacks On Modern Processors

M. Denesh Kumr[1], G. Dilli Rani[2], V. Thrimurthulu[3]
[1] II M.Tech VLSI SD Student, CR Engineering College, Tirupathi, Chittoor (Dist) A.P., India,
[2] Asst. Professor, Dept. of ECE, CR Engineering College, Tirupathi, Chittoor (Dist) A.P., India,
[3] Professor, Head of ECE Dept., CR Engineering College, Tirupathi, Chittoor (Dist) A.P., India,

Abstract
Cryptographic applications like ATM and other commercial electronics are more susceptible to side-channel attacks in which the fault injection attacks are more crucial at today's Deep Sub Micron Technology. Hence the identification and modeling of these attacks from the intruders are critical, playing a vital role and became a challenge for current trend of system designs. Therefore it is necessary to design the systems which are able to identify and to detect these intruder attacks on Cryptographic devices.
The project work has categorized into three levels as described here with, at first level we simulate and synthesis the design procedure using XILINX 14.4 ISE system design tool to estimate and analyse the logic and other constraints for further levels, in the second level we model the level1 abstraction on to VERTEX 5 (XC5VLX50T) FPGA with the help of Digilent ADEPT and test the target results and finally at level3 we analyse the internal results of the core FPGA module by using XILINX COREINSERTER AND CHIPSCOPE pro tool.

I. INTRODUCTION

In an attempt to bridge the performance gap between processor and memory speed, modern high-performance processors have highly sophisticated cache systems. The latter include non-blocking and pipelined caches with hardware support for prefetching blocks of instructions and data. While these are primarily performance enhancing features, they inadvertently thwart clever and sophisticated attacks on software implementations of key cryptographic algorithms.

Apart from exploiting logical weaknesses in any cryptographic algorithm, an attacker can extract sensitive information from its implementation using leaked side channel information. For example, timing information, power consumption and electromagnetic leaks can all provide an extra source of information. The Advanced Encryption Standard (AES), a relatively new algorithm for secret key cryptography, is now widely supported on servers, browsers, etc.The software implementation of AES is memory intensive due to table lookups performed in lieu of time-consuming mathematical field operations. These tables are brought into cache during encryption making AES vulnerable to cache based side channel attacks.

II. RELATED WORK

The vulnerability of lookup tables based implementation of AES was first identified by Bernstein in 2004. This paper reports the extraction of a complete AES key. The target server uses its key to encrypt data using the Open SSL implementation of AES on Pentium III machine. An improved attack using cache collision on Pentium III, Pentium IV Xeon, and Ultra SPARC III appears in. The cache systems of these machines do not support prefetching. Prefetching was supported from Pentium IV series onwards excluding Intel Xeon processors. Consequently, the attack in is not successful on Intel Dual Core, Core 2 Duo and AMD Athlon.

III. AES PRELIMINARIES

A. AES Overview

AES is a symmetric key algorithm standardized by the U.S. National Institute of Standards and Technology (NIST) in 2001. Its popularity is due to simplicity in its implementation yet it is resistant to attacks such as linear and differential cryptanalysis. The full description of AES cipher is provided in. In this paper, we briefly summarize only the relevant aspects of its software implementation.

AES is a substitution-permutation network. It supports a key size of 128,192 or 256 bits and block size = 128 bits. A round function is repeated a fixed number of times (10 for key size of 128 bit) to convert 128 bits of plaintext to 128 bits of ciphertext. The 16 byte input is expressed as a 4x4 array of bytes. Each round involves four steps - Byte Substitution, Row Shift, Column Mixing and a round key operation. The round operations are defined using algebraic operations over the field GF(2^8). In a software implementation, field operations may be

replaced by inexpensive table lookups thereby speeding encryption and decryption.

Four tables are employed (each of size 1KB). Each round involves only 4 table lookups and 4 EX-ORs per column per round. Each table, T_i is accessed by using an 8 bit index and outputs a 32-bit string. There is a separate key setup phase where a given 16-byte secret key $k = (k_o, k_{15})$ is expanded into 10 round keys, $K^{(t)}$ for r = 1, . . . , 10. Every round key is divided into 4 words of 4 bytes each: $K^{(r)} = (K_o^{(r)}, K_1^{(r)}, K_2^{(r)}, K_3^{(r)})$. The 0^{th} round key is just the raw key: $K_i^{(0)} = (k_{4i}, k_{4i+1}, k<_{4i+2}, k_{4i+3})$ for j = 0,1,2,3.

$$(X_0^{(r+1)}, X_1^{(r+1)}, X_2^{(r+1)}, X_3^{(r+1)}) \leftarrow T_0[x_0^{(r)}] \oplus T1[x_5^{(r)}] \oplus T_2[x_{10}^{(r)}] \oplus T_3[x_{15}^{(r)}] \oplus K_0^{(r+1)}$$

$$(X_4^{(r+1)}, X_5^{(r+1)}, X_6^{(r+1)}, X_7^{(r+1)}) \leftarrow T_0[x_4^{(r)}] \oplus T_1[x_9^{(r)}] \oplus T_2[x_{14}^{(r)}] \oplus T_3[x_3^{(r)}] \oplus K_1^{(r+1)}$$

$$(X_8^{(r+1)}, X_9^{(r+1)}, X_{10}^{(r+1)}, X_{11}^{(r+1)}) \leftarrow T_0[x_8^{(r)}] \oplus T_1[x_{13}^{(r)}] \oplus T_2[x_2^{(r)}] \oplus T_3[x_7^{(r)}] \oplus K_2^{(r+1)}$$

$$(X_{12}^{(r+1)}, X_{13}^{(r+1)}, X_{14}^{(r+1)}, X_{15}^{(r+1)}) \leftarrow T_0[x_{12}^{(r)}] \oplus T_1[x_1^{(r)}] \oplus T_2[x_6^{(r)}] \oplus T_3[x_{11}^{(r)}] \oplus K_3^{(r+1)}$$

Finally to compute the last round, the above equation is repeated with r = 9, except that $T_0, \ldots T_3$ is replaced by $T_0^{(10)}, \ldots T_3^{(10)}$. The resulting $X^{(10)}$ is the ciphertext. The change of lookup tables in the last round (for r = 10) is due to the absence of the Column Mixing step.

B. Attack Overview

The access driven cache timing attack is described in [10]. It exploits the fact that in the first round of AES, the nonaccessed table indices are simply $x_i^{(0)} \neq p_i + k_i$ for all i=0,...,15, where p_i and k_i are the i^{th} byte of the plaintext and encryption key. Therefore, *if* $x_i^{(0)}$ is the non-accessed index of the four lookup tables (T_0, T_1, T_2, T_3), then we have $k_i^{(0)} \neq p_i + x_i^{(0)}$. So, as long as we identify the nonaccessed table index x_i,

we can eliminate all impossible key candidate values for k_i from the initial 256 possible values. For finding whether a block is accessed or not during AES, explicit cache considered as an "accessed block". Hence, precise measurement of timing is very important in such attacks.

$k_i \neq p_i \oplus x_i \neq$ 0x66 $\oplus$ *0x60, 0x61, 0x62, 0x63, 0x64, 0x65, 0x66, 0x67, 0x68, 0x69, 0x6a, 0x6b, 0x6c, 0x6d, 0x6e, 0x6f*

The heart of the attack is thus precise identification of nonaccessed cache blocks of AES lookup tables which, in turn, depends on accurate timing information.

IV. Identifying the location of AES Tables

+Table I summarizes relevant features of the machines used in our experiments. The OpenSSL version 0.9.8(a) of AES implementation was targeted for the attack.

TABLE I. CACHE DETAILS OF EXPERIMENTAL SYSTEMS

System	Inclusive/ Exclusive Cache	L1 cache	L2 cache
Intel Dual Core (1.73 GHz) and Intel Core 2 Duo (1.83 GHz)	Inclusive	32KB, nonsharcd, 8-way set associative	2MB, shared, 8-way set associative

A. Original Prime+Probe method

The first step of our attack is to determine the location cf the AES lookup tables.

TABLE I. CACHE DETAILS OF EXPERIMENTAL SYSTEMS

System	Inclusive/ Exclusive Cache	L1 cache	L2 cache
Intel Dual Core (1.73 GHz) and Intel Core 2 Duo (1.83 GHz)	Inclusive	32KB, nonsharcd, 8-way set associative	2MB, shared, 8-way set associative
AMD Athlon, 2.0 GHz	Exclusive	64KB, nonshared, 2-way set associative	512KB, nonshared, 16-way set associative

A. Original Prime+Probe method

The first step of our attack is to determine the location cf the AES lookup tables. The second step is to precisely identify which lines of the tables have not been accessed in a complete encryption of a block of plaintext. In the first step, we use Modified Prime+Probe method. The basic method [6] is summarized below after which the modified approach is presented.

The attacker allocates a byte array, *A* of size *b*a*s* where *b* is the block size, *a* is the associativity and *s* is number of sets in cache.

1. Prime Phase: Every b"[1] byte in array, *A* is accessed by the attacker.
2. Encryption is performed by the victim on plaintext.
3. Probe Phase: The attacker reads the same bytes of array, *A*, accessed in Step 1. The time to access each byte is carefully measured.

In Step 1, one byte per block in A is accessed so that every block of cache is filled. Step 2 (encryption) involves fetching required blocks of the AES tables. This evicts specific blocks containing array *A*. When elements of *A* in those blocks are read in the Probe phase, their access times, on average, will be higher those for other blocks. To determine the location of the AES tables in physical memory, we compute, for each of the *s* sets in cache, the maximum of the access times of the blocks in that set. In a plot of these times versus set number, we would expect to see a plateau where the AES tables reside.

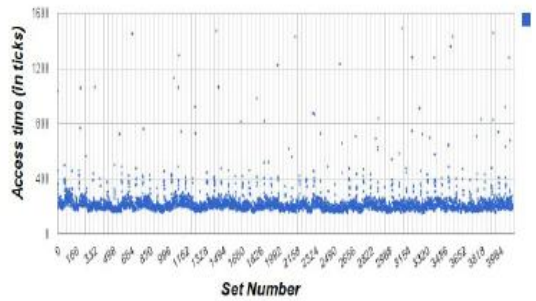

Fig. 1. Original Prime+Probe results on L2 cache of 1.73 GHz Intel Dual Core. Horizontal axis: cache set number; Vertical axis: maximum access time among the blocks of a set (ticks).

B. Effect of prefetching

The principle of spatial locality is leveraged in multiple ways in cache systems. A block - the granularity of transfer between the cache and the next level of the memory hierarchy - is usually 64 bytes so thai several words may be transferred in a single access. Further, to avoid expensive memory stalls, modern cache systems prefetch one or more additional, consecutive blocks of data/instructions in anticipation of their future use. Prefetching is often processor-specific and manufacturers do not generally release details of prefetching behavior. Prefetching can also be done by the compiler - inserting prefetch instructions when it detects regular access patterns to data. For example, while iterating over a large array in a loop, as much as a page of data may be prefetched.

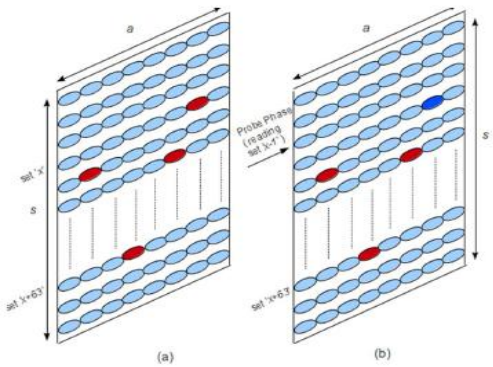

Fig. 2. Schematics of cache states during Prime+Probe (a) Cache is filled with attacker's data (light blue color) and lookup tables occupy 64 blocks from set 'x' to 'x+63' (red color), (b) In probe phase, while reading attacker's blocks that map to set 'x-1', blocks that map to set 'x' are prefetched evicting the block of AES lookup table (dark blue color) that maps to set 'x*.

Fig. 2 shows the layout of a typical a-way set associative cache wherein each cell represents a block of cache. All cells in a single row map to the same set. The

C. Modified Prime+Probe method for handling prefetching

Hardware prefetching may be disabled through the CPU/FSB configuration in BIOS but this requires the attacker to have permission to change the BIOS settings.

The steps to find the location of the AES tables are:

For each cache set i do

1. Prime phase: A byte is written into each block of set. Then, the time taken to read the byte from each block is measured.
2. Encryption is performed by the victim on plaintext.
3. Probe phase: The time taken to read the byte from each block of set *i* is measured. The maximum of these times is recorded.

The maximum access times were plotted in Fig.

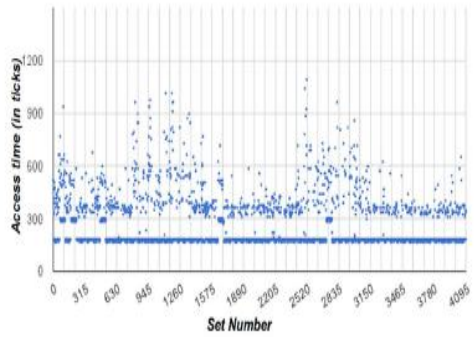

Fig. 3. Modified Prime+Probe results on L2 cache Core for 1 run. Horizontal axis: cache set number; access time among the blocks of a set (ticks)

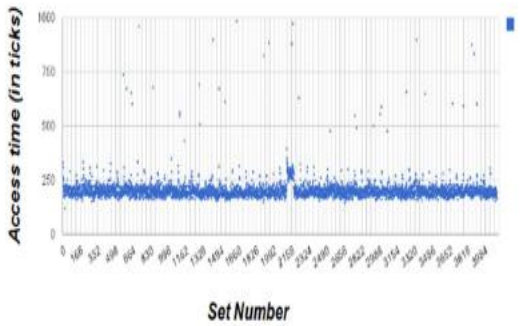

Fig. 4. Modified Prime+Probe results on L2 cache of 1.73 GHz Intel Dual Core for 35 runs. Horizontal axis: cache set number; Vertical axis: maximum access time among the blocks of a set (ticks)

This greatly helped to unambiguously identify the location of the AES tables. Identical experiments performed on the Intel Core 2 Duo were similarly successful.

Fig. 5 shows two chunks over which the AES tables were split during one complete run of the attack. The first chunk comprises 25 sets while the second chunk comprises 39 sets. The ranges of set numbers for the two chunks are respectively:

615 = 001001 100111 through
639 = 001001 111111 and
896 = 001110 000000 through
934 = 0 0 1 1 1 0 100110

We also performed the Modified Prime+Probe experiment on the AMD Athlon X2. This machine has two cores with nonshared caches [12], So, we switched off one of the cores [19] to ensure that the attacker and victim processes ran on the same core. While we used the L2 cache on the Intel Dual Core and Core 2 Duo machines, we used LI cache here. The latter has 512 sets while the LI cache of the Intel machines has only 64 sets. Fig. 6 shows the results obtained with the AMD Athlon. The location of the AES tables on this platform is highly conspicuous. Also, the LI cache is virtually indexed thus obviating the need for the preprocessing step with L2 cache.

D. Effect of other cache properties

From our experiments, we have also observed that the type of cache affects the noise observed during timing measurements. When the experiment to find lookup table location, was performed on L2 cache (unified) in Intel systems, it took 35-40 iterations of Modified Prime+Probe approach to reduce noise whereas for doing the same task on LI cache (data) in AMD, 5-10 iterations were sufficient. This is because the observed "noise" in a unified cache is more compared to that in a cache used exclusively for data.

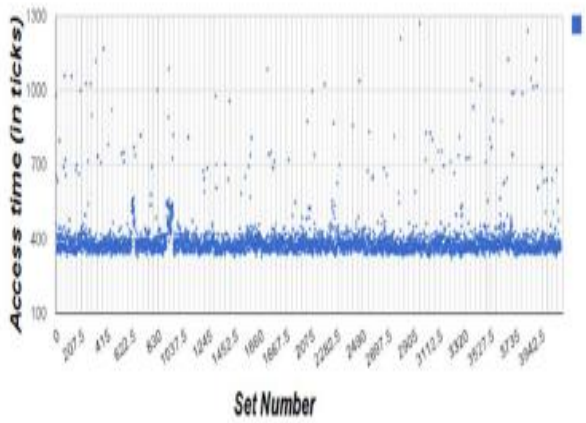

Fig. 5. Cache sets (noncontiguous) of AES lookup tables on L2 cache of 1.73 GHz Intel Dual Core. Horizontal axis: cache set number; Vertical axis: maximum access time among the blocks of a set (ticks)

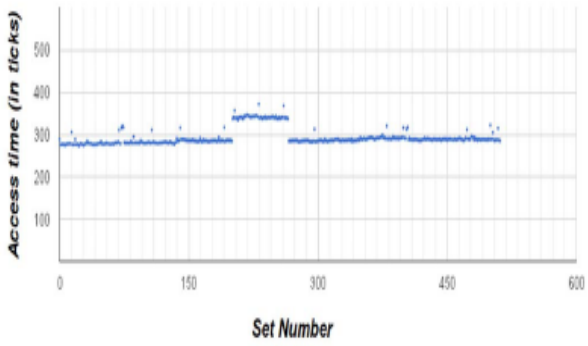

Fig. 6. Sets of AES lookup tables on LI cache of *2* GHz AMD Athlon x2. Horizontal axis: cache set number; Vertical axis: maximum access time among the blocks of set (ticks).

V. Identification of Non Accessed Blocks

Let the set of cache sets to which AES lookup tables map is *S'*. Using 5", nonaccessed blocks are determined by Evict+Time method as follows:

1. For each set *i* in *S'* and a plaintext;?
   - Run AES to get the blocks into the cache.
   - Run AES again and record the time taken for encryption, ***cachedTime.***
   - Evict all the blocks of set *i* (access *a* blocks which map to set *i)*
   - Run AES again and record the time taken for encryption after eviction, ***evictedTime.***
   - Record the ***evicted Time – cached Time* in *excess Time [i]***
2. Calculate the average excess time, ***avgexcessTime*** for all sets in *S*
3. **If** the ***excessTime[i]*** is greater than ***avgexcessTime*** then ***frequency[i]*** is incremented.
4. Repeat steps 1 to 3 for 20 times for the same plaintext.
5. If value of ***frequency[i]*** is 0, then set i has not been accessed.

For each plaintext, we measure the impact of absence of each block on the time taken for encryption multiple times to reduce unexpected timing deviations. This is to make sure that the conclusion that a block of AES table is not accessed by AES is correct. Even a single wrong conclusion would mean the elimination of possible candidate key bytes from the search space. As shown in Fig. 7, the nonaccessed blocks are highlighted by red circle for a single random plaintext. The lookup table blocks resides in sets 2112 to 2175 as located by using Modified Prime+Probe method (Fig.

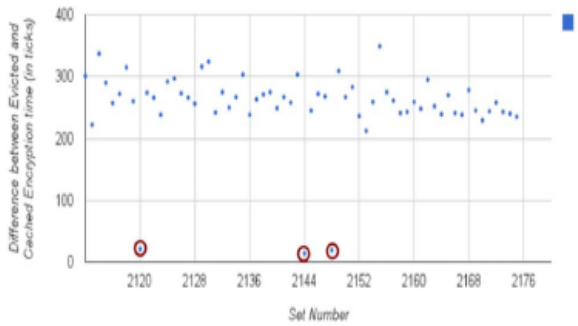

Fig. 7. Nonaccessed blocks obtained using Evict+Time method on L2 cache of 1.73 GHz Intel Dual Core. Horizontal axis: Cache set number, Vertical axis: difference between Evicted and Cached encryption time (ticks).

The information of nonaccessed blocks is used to perform the attack as explained in section III(B).

VI. Conclusion

Cache based side channel attacks on software implementation of AES were previously implemented on an earlier generation of processors. Our preliminary observations suggest that those attacks are not possible on more recent machines. In particular, prefetching thwarted those attacks. Hence, we modified and extended those attacks and experimentally demonstrated that our attacks are successful on Intel dual core, Core 2 Duo and AMD Athlon x2.

We are currently experimenting with porting modified version of the attack on Intel Core i3, i5 and i7 processors.

REFERENCES

[1] John L. Hennessy and David A. Patterson, "*Memory Hierarchy Design*," in *Computer Architecture, A Quantitative Approach*, 5" ed. Morgan Kaufmann, 2011, ch. 2, sec. 2, pp. 78-95

[2] N. Lawson, "*Side-Channel Attacks on Cryptographic Software*", *IEEE Security & Privacy*, vol. 7, no. *6*, pp.65 -68 2009

[3] G. Bertoni, V. Zaccaria, L. Breveglieri, M. Monchiero and G. alermo, "*AES Power Attack Based on Induced Cache Miss and Countermeasure*", Proc. of the International Conference on Information Technology: Coding and Computing (ITCC05), 2005

[4] J. Daemen and V. Rijmen, "*The Design of Rijndael: AES - The Advanced Encryption Standard*," Springer, 2002. ISBN 3-540-42580-2. (238pp.)

[5] Yinqian Zhang, A. Juels, A. Oprea, and M.K. Reiter. "*Home Alone: Coresidency Detection in the Cloud via Side-Channel Analysis*," In security and Privacy (SP) ,2011 IEEE Symposium on, pages 313 -328, May 2011

AUTHORS

Name: DENESH KUMAR MUKKU DENESH KUMAR MUKKU received his B.Tech Degree in Electronics & Communication Engineering from Chadalawada Ramanamma Engineering College Tirupati (A.P),India, in the year2011. Currently pursuing his M.Tech Degree in VLSI SYSTEM DESIGN at Chadalawada Ramanamma Engineering College, Tirupati (A.P),India ..His area of research includes in

VLSI MODELLING OF SIDE CHANNEL ATTACKS ON MODERN PROCESSORS.

Name : G.DILLIRANI

Chadalawada Ramanamma Engineering College Tirupati .G.Dillirani is currently working as Assistant Professor in the Department of Electronics & Communication Engineering at Chadalawada Ramanamma Engineering College Tirupati, India .She has 6 years of teaching experience. Her's extensive education includes B.Tech from SIET Puttur,from JNTU Hyderabad University, plus M.Tech in SIET Puttur, A.P, India. She is making research in the field of Biomedical Signal Processing.

Name: Dr.V.THRIMURTHULU

Dr.V.Thrimurthulu M.E., Ph.D., MIETE., MISTE Professor & Head of ECE Dept. He received his Graduation in Electronics & Communication Engineering AMIETE in 1994 from Institute of Electronics & Telecommunication Engineering, New Delhi, Post Graduation in Engineering M.E specialization in Microwaves and Radar Engineering in the year Feb, 2003, from University College of Engineering, Osmania University, Hyderabad., and his Doctorate in philosophy Ph.D from central University, in the year 2012. He has done his research work on Ad-Hoc Networks.