ISSN: 2278 – 909X
International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE)
Volume 4, Issue 9, September 2015

Verification of Side Channel Attacks on Security Trends and Modern Applications

[1]S.Nireesha
M.tech Student (VLSI), Department of ECE, Shree Institute of Technical Education, Tirupathi.
[2]G.Prasad
Assistant Professor, Department of ECE, Shree Institute of Technical Education, Tirupathi.

***Abstract*— To streamline execution, propelled processors have parts like non-blocking and pipelined stores close by support for data/bearing pre-bringing. In any case, some of these components incidentally destroy absolutely arranged ambushes on cryptographic counts. The strikes fit in with the class of side channel ambushes wherein subtle spillage of information through side channels, for instance, power and timing can be abused to exchange off riddle enters in fragile applications. This paper further expands and changes the present work in the field of store construct side channel ambushes centering with respect to the item execution of Advanced Encryption Standard (AES) - the genuine standard for puzzle key cryptography. The strikes manhandle the way that each AES round makes expansive use of lookup tables in lieu of unrestrained field operations. It is normal that the attacker and loss (the system running AES) offer the same processor store. This is of remarkable essentialness to the cloud environment where two application Rcations may be encouraged on differing virtual machines on the same focus or on particular focuses having the same store. Despite describWg the chaRenges encountered, our exploratory results demonstrate the feasibRity of the changed ambush on the Intel Dual Core, Core 2 Duo and AMD Athlon X2.Index Terms—side channel assaults, stores, pre-bringing, processors, AES, lookup tables, aggress**

1. INTRODUCTION

While trying to connect the execution crevice in the middle of processor and memory speed, current superior processors have very modern store frameworks. The recent incorporate non-blocking and pipelined reserves with equipment support for pre getting pieces of guidelines and information [1].
While these are fundamentally execution upgrading elements, they unintentionally ruin cunning and refined assaults on programming usage of key cryptographic calculations. Aside from abusing intelligent shortcomings in any cryptographic calculation, an assailant can separate touchy data from its execution utilizing released side channel data [2]. Case in point, timing data, power utilization [3] and electromagnetic breaks can all give an additional wellspring of data. The Advanced Encryption Standard (AES) [4], a generally new calculation for mystery key cryptography, is currently broadly upheld on servers, programs, and so on. The product usage of AES is memory serious because of table lookups performed in lieu of drawn out 'numerical field operations [4]. These tables are brought into store amid encryption making AES helpless against reserve based side channel assaults. A run of the mill assault situation is one in which assailant and casualty are executing on two diverse virtual machines on the same center [5]. The casualty is, for instance, an information stockpiling administration supplier who safely stores reports from numerous customers and outfits them on solicitation after due validation. The same key or set of keys is utilized to scramble reports from diverse customers preceding stockpiling. The aggressor is a vindictive client of the administration supplier who more than once demands administration from the supplier. This sort of assault goes under the classification of synchronous assault [6] in which the aggressor can cooperate with the casualty process. The key thought is to have the assailant populate and read a vast cluster (that basically gets to each line of store). At that point, the casualty performs encryption after which the cluster is read once more. The casualty procedure will oust a few lines of the aggressors cluster to make space for that some piece of the AES lookup tables it employments. By measuring the entrance times, both previously, then after the fact the encryption, the assailant reasons which pieces of the AES tables were not utilized by the casualty. This methodology, called Prime+Probe [6], helps the assailant to diminish the hunt space of the key from approximately 2128 to around 236 which bumpers an animal power assault achievable. Store based side channel assaults were executed on past era processors like Pentium IH [7, 8], Pentium 4E [9] and Athlon 64 [6, 10]. One of the objectives of this paper is to develop and change prior store based assaults and show that they could even be effectively keep running on later processors, for example, the Intel Dual Core and Intel Core 2 Duo.

This paper is sorted out as takes after: Section II condenses the business related to this field. section IH contains a brief prologue to AES usage utilizing lookup tables and the cryptanalytic assault on AES.

Segment IV clarifies why the pre-getting streamlining yields uncertain results and indicating fruitful distinguishing proof of lookup table areas. The usage and aftereffects of the second stride of the assault utilizing Evict+Time strategy is as a part of segment V. Area VI finishes up the paper.**II.**

EXISTING SYSTEM

The defenselessness of lookup tables based usage of AES was initially recognized by Bernstein in 2004 [8]. This paper reports the extraction of a complete AES key. The objective server utilizes its key to encode information utilizing the Open SSL usage of AES on Pentium HI machine. The reserve frameworks of these machines don't bolster pre-bringing. Pre-getting was bolstered from Pentium IV arrangement onwards barring Intel Xeon processors [15]. Hence, the assault in [7,8] is not fruitful on Intel Dual Core, Core 2 Duo and AMD Athlon. Recuperation of full 128 bit AES enter initially showed up in [10] on AMD 64 utilizing just first round assault with 350 examples. They accepted that the area of the casualties AES lookup tables is known while we expressly address this issue in our work. Likewise, they report few subtle elements of the engineering viewpoints identified with the assault. Definite business related to this assault was accounted for by Osvik and Shamir in 2006 [9]. They actualized the assault on Pentium 4E (single center framework) for removing a complete AES key. Later, they reconsidered their assault [9] in 2009 on AMD64 framework [6]. They additionally accept information of the area of every lookup table and henceforth of the reserve sets to which it is mapped. They utilized pointer pursuing procedure to handle pre-getting. Be that as it may, the principle center of their paper was on cryptographic parts of the assault though our emphasis is on structural issues - primarily the disposal of the impact of pre-getting and other reserve 'related issues.

Background

The attainability of the store based side-channel assaults, contracted to "reserve assaults" from here on, was initially said by Kocher and after that Kelsey et al. in [13, 14]. D. Page portrayed and recreated a hypothetical store assault on DES [23]. Real reserve based timing assaults were actualized by Tsunoo et al. [27, 28]. The first assault onMISTY1 proposed in [28] has as of late been enhanced in [29]. The subject of reserve based side-channel assaults has been extremely prominent since mid 2005. In spite of the fact that, store side-channel danger had been known for a few years, the first productive and sensible assaults were not created until 2005. Bernstein demonstrated the weakness of AES programming usage on different stages [5]. There was a typical conviction that Bernstein's assault is a practical remote assault and it can recoup a whole AES key. In any case, Neve et al. indicated in [17] that this is just an error. They portrayed the circumstances in which the assault may work furthermore the confinements of the Bernstein assault. The subtle elements of this examination can likewise be found in [19]. At the same time, however freely of Bernstein's endeavors, an examination group that comprises of Acıı¸cmez, Schindler, and Ko¸c added to a reasonable remote assault on the AES. Albeit there is not any openly accessible report of their work, they exhibited the rudiments of the assault in a few events [2]. Osvik et al. depicted different nearby reserve assault variations first in [21] in 2005, then they exhibited their outcomes at CT-RSA in mid 2006 [22].

They made utilization of a neighborhood exhibit and abused the crashes between the table lookups and the entrance operations to this cluster. Neve et al. enhanced the assaults in [22] by taking the last AES round into thought [18]. The same thought of abusing crashes between two unique procedures was likewise utilized by Colin Percival as a part of [26]. He made utilization of synchronous multithreading component of the current processors and built up a reserve assault on RSA. Like outside crashes between distinctive procedures, the interior impacts inside a figure can likewise be exploited. Inside store crashes were initially utilized as a part of [27] and [28]. The remote assault of Acıı¸cmez et al. what's more, Lauradoux's assault are likewise in view of interior crashes [2, 16]. A late composition that compresses store crash assaults on AES will be introduced at CHES'06 [6]. A few equipment and programming based countermeasures were proposed to anticipate store assaults. There are three distinct sorts of store assaults, in particular time-driven, follow driven, and access-driven. Time-driven and follow driven assaults were initially portrayed by Page in [23]. Access-driven assaults are generally new and first seen in [21, 22]. The distinction between these assault sorts are the capacities of the foe. The enemy is thought to have the capacity to catch the profile of the store action amid an encryption in follow driven assaults. This profile incorporates the results of each memory get to the figure issues as far as reserve hits and misses. Along these lines, the foe can watch if a specific access to a lookup table yields a hit and can surmise data about the lookup files, which are key ward. This capacity gives a foe the chance to make surmisings about the mystery key. Time-driven assaults, then again, are less prohibitive in light of the fact that they don't

depend on the capacity of catching the results of individual memory gets to. Enemy is thought to have the capacity to watch the total profile, i.e., aggregate quantities of store hits and misses or possibly an esteem that can be utilized to surmised these numbers. Case in point, the aggregate execution time of the figure can be measured and used to make surmising about the quantity of reserve misses in a period driven store assault. In access-driven assaults, the foe can focus the reserve sets that the figure procedure changes. Accordingly, she can comprehend which components of the lookup tables or S-boxes are gotten to by the figure. At that point, the wrong key suspicions that would bring about an entrance to un-got to parts of the tables can be dispensed

III PROPOSED SYSTEM

A. AES Overview

AES is a symmetric key calculation institutionalized by the U.S. National Institute of Standards and Technology (NIST) in 2001. Its prevalence is because of effortlessness in its execution yet it is impervious to assaults, for example, straight and differential cryptanalysis. The full portrayal of AES figure is given in [4]. In this paper, we quickly outline just the pertinent parts of its product usage. AES is a substitution-change system. It bolsters a key size of 128, 192 or 256 bits and piece size = 128 bits. A round capacity is rehashed an altered number of times (10 for key size of 128 bit) to change over 128 bits of plaintext to 128 bits of ciphertext. The 16 byte info is communicated as a 4x4 variety of bytes. Every round includes four stages - Byte ubstitution, Row Shift, Column Mixing and a round key operation. The round operations are characterized utilizing mathematical operations over the field GF(28). In a product execution, field operations may be supplanted by modest table lookups consequently speeding encryption and unscrambling [4]. Four tables are utilized (each of size IKB).

Each round where a given 16-byte secret key k = (ko, . . , kls) is expanded into 10 round keys [6], K(r) for r = 1, ... , 10. Every round key is divided into 4 words of 4 bytes each: K(r) = (Ko(r), K1 (r), K2(r) , K3(r)). The Oth round key is just the raw key: Kj(o) = (k4j, k4j+1, k4j+2, k4j+3) for j = 0, 1, 2, 3. Given a 16-byte plaintext p = (po, , pls), encryption proceeds by computing a 16-byte intermediate state X(r) = (xo, . , Xls) at each round r. The initial state X(o) is computed by Xi(o) = pi @ ki for (i = O, , 15). The first 9 rounds are computed by updating the intermediate state using the following equation [4], for r = 0. . . 8.

$(X_0^{(r+l)}, X_1^{(r+1)}, X_2^{(r+1)}, X_3^{(r+1)},) \leftarrow T_0\ [x_0^{(r)}]\ T_l[X_5^{(r)}]\ T_2[X_{10}^{(r)}]\ T_3[X_{15}^{(r)}]\ K_0^{(r+1)}$

$(X_4^{(r+l)}, X_5^{(r+1)}, X_6^{(r+1)}, X_7^{(r+1)},) \leftarrow T_0\ [x_4^{(r)}]\ T_l[X_9^{(r)}]\ T_2[X_{14}^{(r)}]\ T_3[X_3^{(r)}]\ K_1^{(r+1)}$

$(X_8^{(r+l)}, X_9^{(r+1)}, X_{10}^{(r+1)}, X_{11}^{(r+1)},) \leftarrow T_0\ [x_8^{(r)}]\ T_l[X_{13}^{(r)}]\ T_2[X_2^{(r)}]\ T_3[X_7^{(r)}]\ K_2^{(r+1)}$

$(X_{12}^{(r+l)}, X_{13}^{(r+1)}, X_{14}^{(r+1)}, X_{15}^{(r+1)},) \leftarrow T_0\ [x_{12}^{(r)}]\ T_l[X_1^{(r)}]\ T_2[X_6^{(r)}]\ T_3[X_{11}^{(r)}]\ K_3^{(r+1)}$

Finally to compute the last round, the above equation is repeated with r = 9, except that To, ..., T3 is replaced by To(lo), ..., T3(lo). The resulting X(lo) is the cipher text. 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 entrance driven store timing assault is depicted in [10]. It misuses the way that in the first round of AES, the nonaccessed table lists are essentially Xi(o) pt @ kt for all i=0,...,15, where pi and ki are the ith byte of the plaintext and encryption key. Consequently, if X{o) is the nonaccessed record of the four lookup tables (To, Tl , T2, T3), then we have kt(o) pt @ X,.(o). In this way, the length of we recognize the nonaccessed table record xi, we can dispose of all incomprehensible key hopeful qualities for kt

from the starting 256 conceivable qualities. For discovering whether a piece is gotten to or not amid AES, unequivocal store square expulsion is finished by getting to aggressors hinders that guide to the same set as that of the AES lookup table square. The same should be possible by utilizing clflush ( ) [18]. To begin with the time for encryption with all lookup table squares in reserve is measured. A particular store piece is then expressly expelled and time is measured for encryption once more.

On the off chance that the piece is utilized by the encryption process, the encryption time will be all the more when contrasted with the encryption time measured with the square in reserve and subsequently the piece would be considered as a "got to piece". Subsequently, exact estimation of timing is vital in such assaults. The granularity of store access is a square which is 64 bytes for our situation. Subsequently, every square of reserve compares to 16 table files (relating to every record a 32 bit worth is put away). By method for instance, let the/th

plaintext bytept be 0x66 and accept To involves sets 0 through 15 of store.

$$ki \neq pi \bigoplus xi$$

≠ 0x66, 0x60, 0x61, 0x62, 0x63, 0x64, 0x65, 0x66, 0x67, 0x68, 0x69, Ox6a, Ox6b, Ox6c, Ox6d, Ox6e, Ox6f

From the AES mathematical statement, four gets to are made in cycle 1. Thus, for any piece in the table that is not got to, we can apply the above mathematical statement for four key bytes to take out competitor keys. On the off chance that this procedure is rehashed for some plaintexts, more competitor keys are disposed of until for every key byte stand out piece is cleared out. In this manner, the competitor key space can be decreased to 16 conceivable bytes from the starting 256 conceivable qualities for every byte. The heart of the assault is therefore exact recognizable proof of non-gotten to reserve pieces of AES lookup tables which, thus, relies on upon precise timing data.

C. Identifying the Location of AES Tables

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

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

D. Original Prime Probe method

The first step of our attack is to determine the location of 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 associatively and s is number of sets in cache.

1. Prime Phase: Every b^{th} byte in array, A is accessed by the attacker..
2. Encryption is performed by the victim on plaintext p.
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 for each square in An is gotten to so that each piece of reserve is filled. Step 2 (encryption) includes bringing obliged squares of the AES tables. This expels particular pieces containing exhibit A. At the point when components of An in those squares are perused in the Probe stage, their entrance times, by and large, will be higher those for different pieces. To focus the area of the AES tables in physical memory, we register, for each of the s sets in reserve, the most extreme of the entrance times of the pieces in that set. In a plot of these circumstances versus set number, we would hope to see a level where the AES tables live.

The measure of the level is an element of b and the extent of the AES tables. For b = 64 bytes and the aggregate size of the AES tables = 4KB (4 tables, each of size IKB), the AES tablesspan 64 lines. These lines happen in bordering sets. Our preparatory analyses indicated outright no such level on Intel Dual center (Fig. 1). This perception is clarified next.

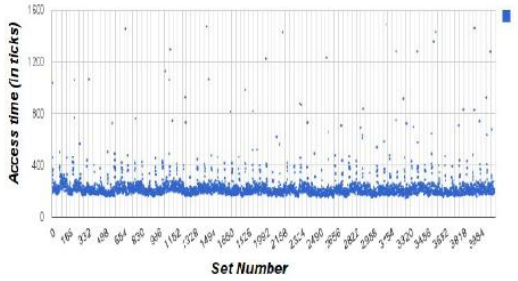


Fig.1. Original Prime+Probe results on L2 cache of I .73 GHz Intel Dual Core. Horizontal axis: cache set number; Vertical axis: maximum access time among the blocks of a set (ticks).

E. Effect of pre-fetching

The rule of spatial area is utilized in numerous routes in reserve frameworks. A square - the granularity of exchange between the reserve and the following level of the memory chain of importance is generally 64 bytes so that few words may be moved in a solitary access. Further, to stay away from extravagant memory slows down, modem

reserve frameworks pre-bring one or more extra, back to back pieces of information/directions in foresight of their future utilization. Pre-getting is frequently processor-particular and makers don't by and large discharge points of interest of pre-bringing conduct. Pre-getting should likewise be possible by the compiler embeddings pre-get directions when it recognizes general access examples to information. Case in point, while repeating over a substantial exhibit in a circle, as much as a page of information may be pre-gotten.

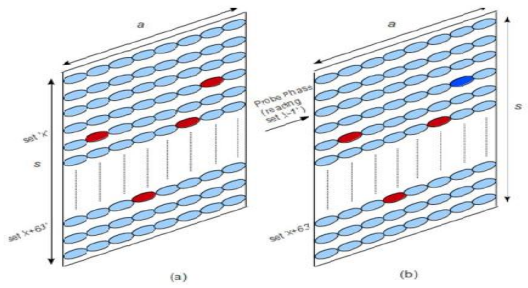


Fig. 2. Schematics of cache states during Prime+Probe (a) Cache is filled with attackers data (light blue color) and lookup tables occupy 64 blocks from set _x to _x+63 (red color). (b) In probe phase, while reading attackers blocks that map to set _x-1, blocks that map to set _x are pre-fetched evicting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

Fig. 2 demonstrates the format of a commonplace a-way set acquainted store wherein every cell speaks to a piece of reserve. All cells in a solitary line guide to the same set. The AES clusters are highlighted in red. Note that, since the gets to the aggressors exhibit happen with settled step, its pieces are pre-brought if not as of now in reserve. Specifically, pieces of the assailants cluster that converge with the AES exhibits are pre-gotten. Our exploratory estimations on distinctive processors substantiate the hypothesis and recommend that information was without a doubt pre-gotten amid the Probe period of the assault accordingly overcoming endeavors to discover the area of the AES exhibits.

F. Modified Prime+Probe method for handling pre-fetching

Equipment pre-bringing may be debilitated through the CPU/FSB setup in BIOS yet this requires the aggressor to have consent to change the BIOS settings. To invalidate the impact of pre-bringing, we propose the accompanying altered adaptation of the Prime+Probe calculation where the complete procedure is rehashed for each situated of reserve (beginning from 0 to s-1). The aggressor first designates a substantial cluster BigArr of b*a*s bytes square with in ability to the reserve. For every square of BigArr , the set to which it is mapped, is computed (utilizing virtual addresses specifically for LI store since LI is by and large practically ordered and utilizing physical locations for L2 reserve subsequent to L2 is physically filed [11]). A preprocessing step is done to change over virtual to physical location for L2 reserve utilizing two framework documents/proc/pid/maps and/proc/pid/pagemap (where pid is the id of assailants procedure) on Linux working framework. From these virtual address and set number matches, a two dimensional exhibit b/ock_addrc[s][a] is readied, where column I speaks to set i. The a sections in column i are addresses, each of which guide to set i. The Modified Prime+Probe strategy uses addresses from this cluster to populate hinders inside of a given set. Note that, since these locations are arbitrary, there is no degree for compiler-helped pre-getting.

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 i then time- taken to read the byte from each This greatly helped to unambiguously Identify the location block is measured.

2 .Encryption is performed by the victim on plaintext p.

3 .Probe phase: The time taken to read the byte from In the software implementation of AES, the four look-up each block of set i is measured. The maximum these times are recorded.

Time measured is in clock ticksl utilizing rdtsc() direction [13]. While sets display higher access times, these are for the most part disconnected making it difficult to uncover the area of the AES tables. To invalidate the impact of the commotion brought on bv the simultaneous execution of different procedures, the same investigation was rehashed various times (35 times on the Dual Core svstem) and the normal of the greatest access times were plotted in Fig. 4.

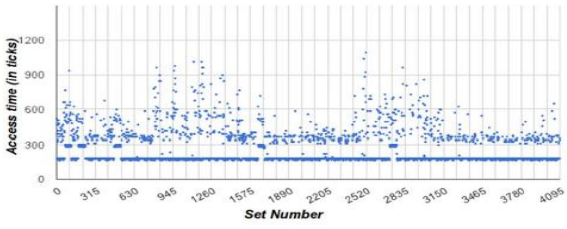


Fig. 3. Modified Prime+Probe results on L2 cache of 1.73 GHz Intel Dual Core for 1 run. Horizontal axis: cache set number; Vertical axis: maximum access time among the blocks of a set (ticks)

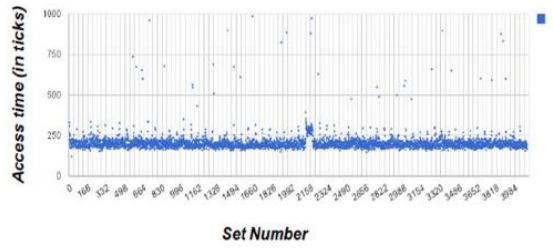


Fig. 4. Modified Prime+Probe results on L2 cache of 1.73 GHz Intel Dual s Core for 35 runs. Horizontal axis: cache set number; Vertical axis: maximum for each cache set i do access time among the blocks of a set (ticks)

To the prompt left of the piece counterbalance field is the list field or set number in store that a specific location maps to. These are the 12 = 1og2 (4 MB/(64 * 8)) bits, i.e. address bits a17 . . . . a6. Since the page size is 4KB, two components of the aggressor exhibit An in agreement guide to two sets with indistinguishable bits, all ..... a6, in the record field. The taking after enlightens are helpful deciding the area of the AES tables.

1. The last byte of the first chunk is the last byte of a page. So, its address must have all 1s in its twelve least significant positions. So, the least significant 6 bits of the set to which this byte is mapped must be all 1s.
2. The first byte of the second chunk is the first byte of a page. So, its address must have all 0s in its twelve least significant positions. So, the least significant 6 bits of the set to which this byte is mapped must be all 0s.
3. The most significant 6 bits of the set number that the blocks of the same chunk map to are identical.

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= 0001001 100111 through
639= 001001 111111
And
896= 001110 000000
934= 001110 100110

We likewise performed the Modified Prime+Probe investigate the AMD Athlon X2. This machine has two centers with non-shared reserves [12]. In this way, we exchanged off one of the centers [19] to guarantee that the aggressor and casualty procedures kept running on the same center. While we utilized the L2 reserve on the Intel Dual Core and Core 2 Duo machines, we utilized LI store here. The recent has 512 sets while the LI reserve of the Intel machines has just 64 sets. Fig. 6 demonstrates the outcomes acquired with the AMD Athlon. The area of the AES tables on this stage is exceedingly prominent. Additionally, the LI store is for all intents and purposes recorded along these lines deterring the requirement for the preprocessing stride with L2 reserve.

G. Effect of other cache properties

From our trials, we have additionally watched that the sort of store influences the commotion saw amid timing estimations. At the point when the trial to discover lookup table area, was performed on L2 store (bound together) in Intel frameworks, it took 35-40 cycles of Modified Prime+ Probe way to deal with diminish clamor while for doing likewise errand on LI reserve (information) in AMD, 5-10 emphasess were adequate. This is on the grounds that the watched "clamor" in a brought together reserve is more contrasted with that in a store utilized solely for information.

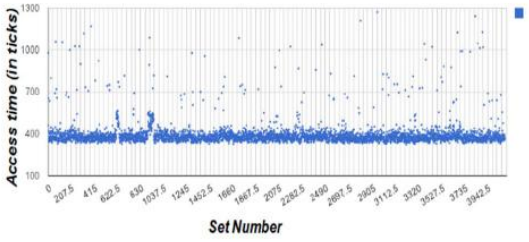


Fig. 5. Cache sets (noncontiguous) of AES lookup tables on L2 cache of 1 .73 GHz Intel Dual Core. Horizontal axis: cache set number; Vertical axis: maximum access time among the blocks of a set (ticks)

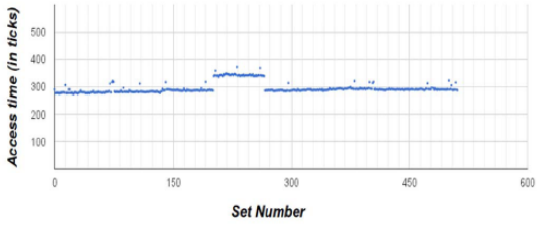


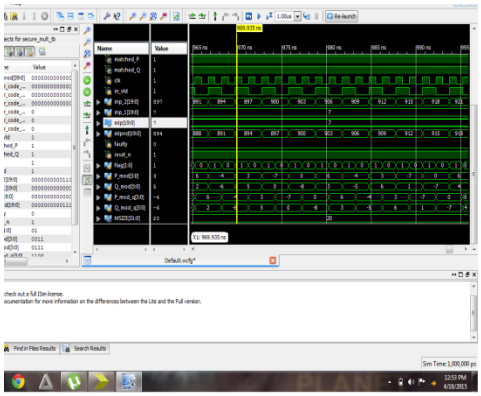
Fig. 6. Sets of AES lookup tables on L I cache of 2 GHz AMD Athlon x2. Horizontal axis: cache set number; Vertical axis: maximum access time among the blocks of set (ticks).

The inclusive and exclusive property of caches also affects the attack. As mentioned in Table I, AMD Athlon x2 systems have exclusive caches

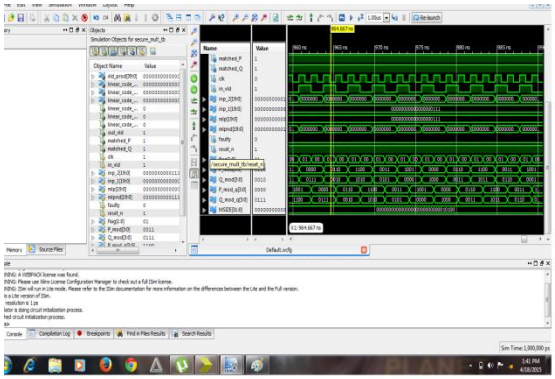
[16]. Such frameworks better use store memory since information may be available in either LI or L2 reserve however not in both. This property makes actualizing side channel assaults more troublesome. Our trials on this machine demonstrated that aggressors information from L2 reserve sets were not removed by the AES process. This is on account of the AES lookup table pieces, when gotten to, are conveyed specifically to LI reserve without getting put away in L2 store. If there should be an occurrence of different machines, AES lookup table squares live in both LI and L2 reserve. In this manner, the adjusted assault was fruitful on L2 store in Intel Dual Core and Core 2 Duo however not in AMD Athlon.

IV SIMULATION RESULTS

A Existing System



B Proposed System



V CONCLUSION

Store construct side divert assaults in light of programming usage of AES were beforehand executed on a before era of processors. Our preparatory perceptions recommend that those assaults are unrealistic on later machines. Specifically, prefetching obstructed those assaults. Subsequently, we changed and developed those assaults and tentatively showed that our assaults are effective on Intel double center, Core 2 Duo and AMD Athlon x2. The assault has two sections. To get the area of AES lookup tables, we utilized the Modified Prime +Probe approach. To distinguish which squares of lookup tables are not got to, we utilized Evict +Time approach. We additionally examined the impact of different elements of store frameworks on the assault, for example, separate direction/information reserves versus brought together reserves and select versus comprehensive reserves. We are right now trying different things with porting changed adaptation of the assault on Intel Core i3, i5 and i7 processors.

REFERENCESS

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

[2] N. Lawson, "Side-Channel Attacks on Cryptographic Software", */EEE Securi & Priva*, 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 (ITCCO5), 2005.

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

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

[6] E. Tromer, D. A. Osvik, and A. Shamir, " Efficient cache attacks on AES, and countermeasures," J. Cryptology, 23(1):37-71, Jan 2009.