Improved Cache Trace Attack on AES and CLEFIA by Considering Cache Miss and S-box Misalignment

Xin-jie ZHAO, Tao WANG

Department of Computer Engineering, Ordnance Engineering College, Shijiazhuang 050003, China, zhaoxinjieem@163.com

Abstract. This paper presents an improved Cache trace attack on AES and CLEFIA by considering Cache miss trace information and S-box misalignment. In 2006, O. Acıiçmez et al. present a trace driven Cache attack on AES first two rounds, and point out that if the Cache element number of the Cache block is 16, at most 48-bit of AES key can be obtained in the first round attack. Their attack is based on the ideal case when S-box elements are perfected aligned in the Cache block. However, this paper discovers that, the S-box elements are usually misaligned, and due to this feature and by considering Cache miss trace information, about 200 samples are enough to obtain full 128-bit AES key within seconds. In 2010, Chester Rebeiro et al. present the first trace driven Cache attack on C LEFIA by considering Cache hit information and obtain 128-bit key with 2^{43} CLEFIA encryptions. In this paper, we present a new attack on CLEFIA by considering Cache miss information and S-box misalignment features, finally successfully obtain CLEFIA-128 key for about 220 samples within seconds.

Keywords: Driven; Cache Attack; AES; CLEFIA; Cache Miss; S-box Misalignment.

1   Introduction

Traditionally speaking, the implementation of cipher is a black box and the security of cipher depends on the mathematical function $E_K[P] \rightarrow C$, usually, the adversary tries to crack K from (P,C) by linear or differential methods. With the development of cipher designing, both the key length and algorithm complexity has been greatly improved; it's very difficult to predict K through mathematically analysis. However, actually, the implementation of cryptosystem is a gray box, it may leak information through side channels due to the prosperities and physical requirements of the device, e.g., execution time, power consumption, electromagnetic emanation, acoustic emanation, fault etc. Since the presentation of side channel attack, many research papers have been published on using this cryptanalysis technique to successfully attack various cryptosystems, such as timing attack[1], power attack[2], EM attack[3], acoustic attack[4], fault attack[5]. In this paper, we focus on a type of side channel cryptanalysis utilizing the information leaks through the Cache architecture of a CPU.

There are many papers on Cache based side channel attacks. According to the different information of the measuring stage, it can be classified into timing driven, access driven, and trace driven three types. We present an improved trace driven Cache based attack on AES and CLEFIA in this paper. In trace driven attacks, there are already some attacks on AES[6][7][8][9] and CLEFIA[11]. However, none of the publish papers described to recover AES initial key and CLEFIA round key in the first round attack. Our analysis model is much like [9] by considering Cache miss trace information, the attacks in this paper show that: due to the feature of S-box misalignment in Cache, it's possible to break AES initial key and CLEFIA round key by first round attack within limited samples, and this is much more effective that any previous studies. Table 1 demonstrates the improvements of the attacks in this paper over several previous attacks.

Table 1. Overview of Cache trace attacks on AES-128 and CLEFIA-128.

Attack	Attack Round	Sample needed	Goal
[6]	First round	--	64-bit key recovery
[7]	First round	--	48-bit key recovery
[8]	First two rounds	40	94.5-bit key recovery
[8]	Final round	100	107.6-bit key recovery
[9]	Final round	31	128-bit key recovery
This paper	First round	200	128-bit key recovery
This paper	Final round	30	128-bit key recovery
[11]	First 3 rounds	2^{43}	128-bit key recovery
This paper	First 3 rounds	220	128-bit key recovery

This work is organized as follows. Section 2 presents the background of Cache trace attacks. Section 3 and Section 4 present the improved Cache trace attack on AES and CLEFIA respectively. Finally, the work is concluded in Section 5.

2   Background

2.1   Cache Information Leakage and Cache Analysis model

Modern microprocessors and microcomputers use memory Cache to solve the speed bottleneck between CPU and bus bandwidth. The "Cache hit" and "Cache miss" can brought time and power consumption differences, and most block ciphers usually use S-box lookup to access Cache. Cache access patterns can be measured through timing and power measurement. Cache provides the source of time and power information leakage for cipher process.

Many block ciphers used S-boxes to improve the implement efficiency and the non-linearity; however, according to the Cache hit and miss timing and power information leakages, these leakages usually have close relationship with the pattern of S-box lookup indices and the secret key. With the development of high-grade precision and sophisticated equipments, the differential behavior in Cache memory with respective to time and power can be measured accurately. Depending on the

attacker abilities, Cache attack can be classified into three types: timing driven attacks[12][13][14][15], access driven attacks[16][17][18], trace driven attacks[6][7][8][9][11].

1. Timing driven Cache attack needs to measure the whole encryption or decryption time of cipher and usually requires millions of samples, its off-line analysis method is more complicated.
2. Access driven Cache attack requires knowledge of the cipher Cache set access pattern by a spy process. Comparing with timing driven Cache attack, its measuring methods are more comparatively complicated, but the analysis methods are much simple.
3. Trace-driven analysis is of high efficiency, but it need to measure every Cache access hit or miss information during the encrypt process, generally speaking, it is usually acquires to gather the Cache hit and miss information by monitoring the power consumption.

2.2   Cache Trace Driven Analysis Model

Trace driven attack is first presented in on AES[6], and researchers make some further studies in [7][8][9]. In 2010, Chester[11] proposes a differential Cache trace attack on CLEFIA, the technique used in this paper is the same as [7] by utilizing Cache hit trace information.

In trace driven Cache attacks, the adversary obtains the traces of Cache hits and misses for some samples of encryptions and recovers the secret key of ciphers using this data. We define a trace as a sequence of Cache hits and misses. For example, *MHHM, MMHH, MHHH, MHMH* are examples of the 16 Cache access traces of AES first round. The letter *M* and *H* denotes a Cache miss Cache hit respectively. In such condition, the adversary has the ability to determine whether a particular Cache S-box lookup to access Cache is a Cache hit or miss.
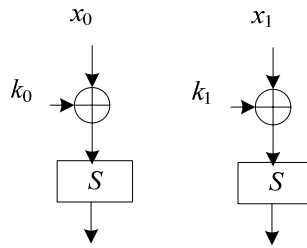
Fig. 1. Trace driven attack model

Fig. 1 shows two access to S-box with indices $(x_0 \oplus k_0)$ and $(x_1 \oplus k_1)$. If $(x_0 \oplus k_0)$ is the first time to access Cache, it can always generate a Cache miss. As to the second time to access Cache, it has two cases:

1. $(x_0 \oplus k_0) = (x_1 \oplus k_1)$

A Cache hit occurs in this case. This results in less execution time and lower power consumption, also reveals some information about the possible candidates for the XOR of the key bits: $(k_0 \oplus k_1) = (x_0 \oplus x_1)$.

2. $(x_0 \oplus k_0) \neq (x_1 \oplus k_1)$

A Cache miss occurs in this case. This result in more execution time and more power consumption, also reveals some impossible candidates for the Xor of the key bits $(k_0 \oplus k_1) \neq (x_0 \oplus x_1)$.

From above, it seems to generate only one guess on the possible $k_0 \oplus k_1$ candidate. However, modern Caches do not store individual bytes, but groups of bytes from consecutive "lines" of main memory. Line size varies, the two common values are 32 bytes for a Pentium III and 64 bytes for Athlon 3000+ CPU. Since the usual size of common block ciphers like AES, Camellia, CLEFIA table entries is 4 bytes; groups of 16 consecutive table entries share a line in the Cache on the Athlon 3000+ CPU. This value we define it as δ. In general, it will hold that $\delta = l/b$ where l is the process's Cache line size and b is the block size of S-box table entries.

The value of δ is critically important to Cache bases attacks because for any address a, b which are equal ($<a> = <b>$) ignoring the lower $\log_2^{\delta}$ bits, looking up address a will cause an ensuing access to b a hit in Cache. Under this precondition, if the second Cache access generates a Cache hit, it will hold $<x_0 \oplus k_0> = <x_1 \oplus k_1>$ (the lower $\log_2^{\delta}$ bits may not identical), so $<k_0 \oplus k_1> = <x_0 \oplus x_1>$, as to 8-bit value k_0, using more samples, the $8 - \log_2^{\delta}$ bits of $k_0 \oplus k_1$ can be obtained, but the $\log_2^{\delta}$ bits of $k_0 \oplus k_1$ is still unknown.

Also, if the second Cache access generates a Cache miss, it will hold that $<x_0 \oplus k_0> \neq <x_1 \oplus k_1>$, so $<k_0 \oplus k_1> \neq <x_0 \oplus x_1>$, if x_0 and x_1 is known, as to one of 256 candidates k_0, the second Cache miss will eliminate δ candidates for k_1, ideally speaking, if k_0 is guessed correctly, about $256/\delta$ Cache miss samples are enough to obtain δ un-eliminated k_1 candidates, the correct key byte is always among them, but if k_0 is guessed wrongly, using more samples, it will always eliminate all 256 k_1 candidates and have a empty candidate set. So the adversary can at most recover $8 - \log_2^{\delta}$ bits value of k_0.

2.3 S-box alignment in Cache

Most of the attacks proposed assume that S-box element are perfected aligned in Cache block, as to one AES S-box, the table size is 1Kb, 256 table elements, one S-box line usually related with 16 elements. If the Cache block size of the processor is 64 bytes, 4 bytes for each Cache element, one S-box line 16 S-box element are perfectly aligned in the same Cache block, as is shown in Fig. 2(a). However, during the experiment on Windows platform, we find out that one S-box line 16 elements are not mapping to the same Cache block, as is shown in Fig. 2(b). 4 S-box elements from 16th to19th are stored in the 1st Cache block, and other 12 S-box element from 20th to 31 are stored in the 2rd Cache block.
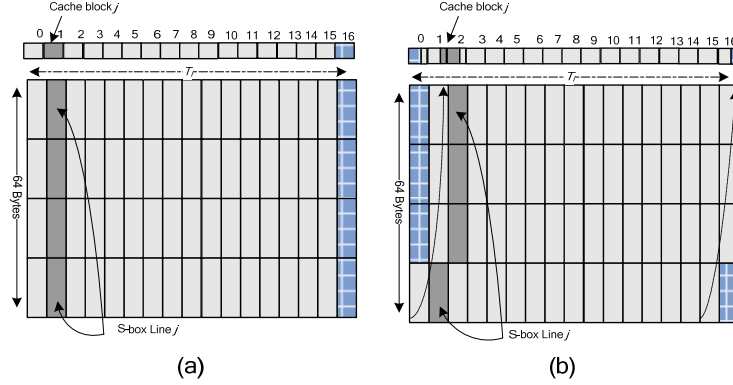
Fig. 2. Alignment of S-box in Cache

2.4 Assumptions

(1) The Cache trace data is gathered through software simulations. We have
simulated this process by augmenting the encryption code to keep a record of every S-
box lookup access Cache hits and misses during the encryption operation.

(2) Unlike the cache trace attacks described by [7], [8] and [10], we do not assume
the cache is completely clean prior to encryption.

(3) We do assume that any cache line loaded during encryption will remain there
for the duration of the encryption.

3   Improved Attack on AES

3.1   AES Algorithm

A full description of the AES cipher is provided in [19], but below is a brief
description of the cipher's properties that are utilized in this study. This paper will
focus exclusively on AES with 128-bit key.

AES is an iterated cipher: Each round r takes a 16-byte block of input X^r and a 16-
byte block of key K^r, producing a 16-byte block of output X^{r+1}. Each round is carried
out by performing the algebraic operations SubBytes, ShiftRows, and MixColumns
on X^r, then taking the exclusive or with the round key K^r. Performance-oriented
software implementation of AES combines all three operations and pre-compute the
values. The values are stored in large lookup tables, T_0, T_1, T_2, T_3, each mapping one
byte of input to four bytes of output. The whole encryption can be performed very
efficient in software this way, using just 160 table lookups and 176 word-length Xors.
But as the table lookup results are stored in the cache, and the table lookup index has

close relationship with the encryption key, as long as we gather enough information of AES table lookup indices, one round expanded key can be calculated. Also, AES key expansion structure is explicitly chosen to be invertible given any 16 consecutive bytes of the expanded key. This is useful to an attacker in that recovery of any 16 bytes expand key is equivalent to the recovery of original key. So Cache, AES table lookup index, invertible key expansion structure provides good points for attacker.

3.2 Related Work

Bertoni et. al. [6] showed that power profiles of the cache reveal secret information about the cryptographic algorithm being executed. Their attack was the first trace driven Cache attack simulated on a power analysis tool and targeted on a single table AES implementation. A first round cache trace attack[7] on the standard software implementation of AES was presented by O. Acııçmez and extended to the second round[8] by exploring Cache hit trace information, Joseph Bonneau[9] extends this work to a final round attack by exploring Cache miss trace information, and recovers AES-128 bit key for about 31 samples. However, none of the published papers except [20] and [21] present the Cache timing attacks by utilizing Cache misalignment feature, which great improves the attack efficiency, but [20] and [21] are belonged to access driven type attack, in this Section, we present an improved trace driven Cache attack using this misalignment feature and Cache miss trace information.

3.3 First Round Attack

The AES we analyze is implemented in OpenSSL v. 0.9.8.(a). It employs 4 different lookup tables in the first 9 rounds, and a different one in the final round.

The AES first 9 rounds encryption principle is shown in equation (1). It's clear to see that in each round, there are 16 S-box lookups to access Cache, and 4 S-box lookups for T_0, T_1, T_2, T_3 S-boxes.

$$
\begin{cases}
(x_0^{r+1}, x_1^{r+1}, x_2^{r+1}, x_3^{r+1}) = T_0[x_0^r] \oplus T_1[x_5^r] \oplus T_2[x_{10}^r] \oplus T_3[x_{15}^r] \oplus (K_0^r, K_1^r, K_2^r, K_3^r) \\
(x_4^{r+1}, x_5^{r+1}, x_6^{r+1}, x_7^{r+1}) = T_0[x_4^r] \oplus T_1[x_9^r] \oplus T_2[x_{14}^r] \oplus T_3[x_3^r] \oplus (K_4^r, K_5^r, K_6^r, K_7^r) \\
(x_8^{r+1}, x_9^{r+1}, x_{10}^{r+1}, x_{11}^{r+1}) = T_0[x_8^r] \oplus T_1[x_{13}^r] \oplus T_2[x_2^r] \oplus T_3[x_7^r] \oplus (K_8^r, K_9^r, K_{10}^r, K_{11}^r) \\
(x_{12}^{r+1}, x_{13}^{r+1}, x_{14}^{r+1}, x_{15}^{r+1}) = T_0[x_{12}^r] \oplus T_1[x_1^r] \oplus T_2[x_6^r] \oplus T_3[x_{11}^r] \oplus (K_{12}^r, K_{13}^r, K_{14}^r, K_{15}^r)
\end{cases}
\tag{1}
$$

The first round S-box lookup can be expressed by equation (2),

$$
x_i^0 = P_i \oplus K_i \Rightarrow K_i = P_i \oplus x_i^0
\tag{2}
$$

P_i and K_i are the ith byte of the plaintext and the initial key respectively, and $i \in [0,15]$. The indices of the first 4 references to the first table T_0 are:

$$
x_0^0 = P_0 \oplus K_0; \quad x_4^0 = P_4 \oplus K_4; \quad x_8^0 = P_8 \oplus K_8; \quad x_{12}^0 = P_{12} \oplus K_{12}
\tag{3}
$$

The Cache hit and miss result of the second T_0 lookup, i.e., the one with the index $P_4 \oplus K_4$, gives information about K_0 and K_4. This paper just explores the Cache miss information. If the second T_0 lookup generates a Cache miss, it will hold:

$$< P_0 \oplus K_0 > \neq < P_4 \oplus K_4 > \Rightarrow K_4 \neq P_4 \oplus \{< P_0 \oplus K_0 >\} \qquad \textbf{(4)}$$

From Section 2,we know that a Cache miss means that the high 8-$\log_2^{\delta}$ bit of the second S-box lookup index is not equal to the high 8-$\log_2^{\delta}$ bit of the first S-box lookup index, so K_4 is impossible to be equal to P_4 Xored with δ candidates has the same high 8-$\log_2^{\delta}$ bit value. As to Athlon 3000+ CPU, the Cache line size is 64 bytes, the S-box element size is 4 bytes, δ=16.

If the Cache S-box is perfected aligned in the Cache, each S-box row 16 bytes can perfectly be mapped into one Cache line, which means that, every Cache miss block is related with 16 high 4-bit identical, and low 4-bit distinct bytes. If P_0=0x61, P_4=0x73, suppose K_0=0x00, the second T_0 Cache miss will eliminate 16 candidates whose high 4-bit identical, and low 4-bit distinct bytes as follows:

$$K_4 \neq P_4 \oplus \{< P_0 \oplus K_0 >\} \neq 0x73 \oplus \{< 0x61 > \oplus x00\} \neq 0x73 \oplus \{< 0x61 >\}$$

$$\neq 0x73 \oplus \{0x60, 0x61, 0x62, 0x63,...0x6f\}$$

$$\neq \{0x13, 0x12, 0x11, 0x10,...0x1c\}$$

$$\neq \{0x10, 0x11, 0x12, 0x13,...0x1f\}$$

Using more Cache miss samples, if K_0 is guessed right, as the correct K_4 candidate and the other 15 candidates has the same high 4-bit value as K_0 are impossible to be eliminated, so at most 16 K_4 candidates can be obtained, but if K_0 is guessed wrongly, all the K_4 candidates can be eliminated.

However, if the Cache S-box is dis-aligned in the Cache, just like Fig .1, it depicts that the offset is 12. The second T_0 Cache miss will eliminate 16 candidates as follows:

$$< P_0 \oplus K_0 > \neq < P_4 \oplus K_4 > \Rightarrow K_4 \neq P_4 \oplus \{< P_0 \oplus K_0 >\}$$

$$K_4 \neq P_4 \oplus \{< P_0 \oplus K_0 >\} \neq 0x73 \oplus \{< 0x61 > \oplus x00\} \neq 0x73 \oplus \{< 0x61 >\}$$

$$\neq 0x73 \oplus \{0x54, 0x55, 0x56, 0x57,...0x5f, 0x60, 0x61, 0x62, 0x63\}$$

$$\neq \{0x27, 0x26, 0x25, 0x24,...0x2c, 0x13, 0x12, 0x11, 0x10\}$$

$$\neq \{0x10, 0x11, 0x12, 0x13, 0x24, 0x25, 0x26,...0x2f\}$$

It's clear to see that we can eliminate 16 candidates, among of which both high 4-bit and low 4-bit are not identical, also 12 high 4-bit identical and 4 high 4-bit identical. Using more samples, if K_0 is guessed correct we can eliminate all wrong 255 candidates, and obtain the unique K_4, if K_0 is guessed wrongly, we can eliminate all 256 wrong candidates and obtain an empty candidate set for K_4.

Apply this technique to other 2 T_0 lookups to recover K_8,K_{12}, other 4 T_1 lookups to recover K_5,K_1,K_{13},K_1, other 4 T_2 lookups to recover K_{10},K_{14},K_2,K_6, other 4 T_3 lookups to recover K_{15},K_3,K_7,K_{11}. Note that the jth (j>1) T_i table Cache access can eliminate 16*(j-1) wrong key candidates at most, the Cache eliminate efficiency is increased with the latter S-box lookups.

3.4 Final Round Attack

In OpenSSL v. 0.9.8.(a) AES implementation, only T_4 table is used in the final round. There are 16 S-box lookups T_4 lookups to access Cache. The AES final round encryption principle is shown in equation (5):

$$\begin{cases} (C_0,C_1,C_2,C_3) = (T_4[x_0^9] \oplus K_0^{10}, T_4[x_5^9] \oplus K_1^{10}, T_4[x_{10}^9] \oplus K_2^{10}, T_4[x_{15}^9] \oplus K_3^{10}) \\ (C_4,C_5,C_6,C_7) = (T_4[x_4^9] \oplus K_4^{10}, T_4[x_9^9] \oplus K_5^{10}, T_4[x_{14}^9] \oplus K_6^{10}, T_4[x_3^9] \oplus K_7^{10}) \\ (C_8,C_9,C_{10},C_{11}) = (T_4[x_8^9] \oplus K_8^{10}, T_4[x_{13}^9] \oplus K_9^{10}, T_4[x_2^9] \oplus K_{10}^{10}, T_4[x_7^9] \oplus K_{11}^{10}) \\ (C_{12},C_{13},C_{14},C_{15}) = (T_4[x_{12}^9] \oplus K_{12}^{10}, T_4[x_1^9] \oplus K_{13}^{10}, T_4[x_6^9] \oplus K_{14}^{10}, T_4[x_{11}^9] \oplus K_{15}^{10}) \end{cases} \quad (5)$$

If the second T_4 access generates a Cache miss, it will have:

$$< T_4^{-1}[C_0 \oplus K_0^{10}] > \neq < T_4^{-1}[C_1 \oplus K_5^{10}] > \Rightarrow K_5^{10} \neq C_1 \oplus \{T_4[< T_4^{-1}[C_0 \oplus K_0^{10}] >]\} \quad (6)$$

No matter $\{< T_4^{-1}[C_0 \oplus K_0^{10}] >\}$ is 16 consecutive high 4-bit identical, or 16 consecutive high 4-bit may not identical, after the T_4 function, they can be transferred to 16 non-consecutive and divergent candidates which both high 4-bit and low 4-bit are not equal. As there are 16 table lookups to the same table T_4, so the eliminate efficiency are much better than first round's 4 lookup of each $T_i (i \in [0,3])$. As to the 16th T_4 table lookup, it can at most eliminate 240 wrong candidates ideally speaking, but as the precondition is to predict other 15 K_{10} bytes correctly.

3.5 Experiment Results

As to Athlon 3000+ CPU, the Cache block size is 64 bytes, and each Cache element size is 4 bytes, so $\delta=16$. During the first round attack on AES-128, the relations between sample size and AES key searching space for different offset ℓ is shown in Fig.3. It's clear to see that if the AES S-box is aligned perfectly ($\ell=0$), after the first round attack, the adversary can reduce the key searching space down to 2^{80} at best, but if the S-box is misaligned, after the first round attack, it's possible to reduce the key searching space down to 2^{16} for about 200 samples.
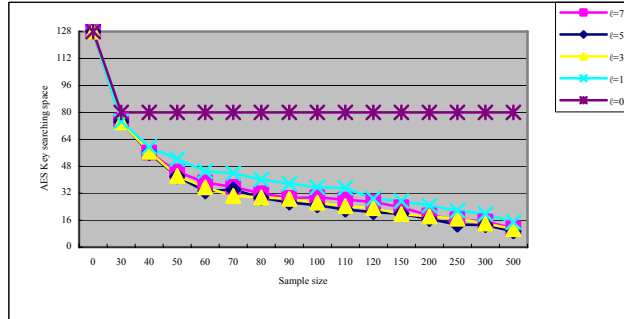
Fig. 3. Sample size and AES key searching space for different offset ℓ
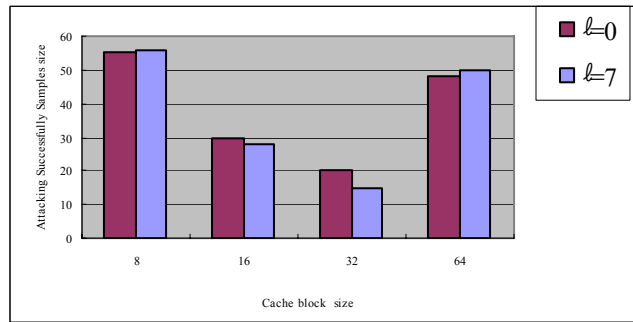

Fig. 4. Cache block size and successfully attacking sample size

During the final round attack on AES, the relations between Cache block element size and attacking successfully sample size for different offset ℓ is shown in Fig.4. It's clear to see that no matter the AES S-box is perfectly aligned or misaligned, it's possible to recover AES-128 key within very limited samples. And due to the perfect avalanche feature of AES S-box, the final round elimination effect is much better than the first round attack.

Compared with previous works of trace driven Cache attacks against AES [6][7][8][9], all of the attacks above all considered the case that the S-box elements are perfected aligned in the Cache block, and it's impossible to obtain full AES-128 key in the first round attack. In fact, at many cases, the S-box elements are usually misaligned in the Cache block, and in these cases, all of the attacks above may not work. The attacks proposed in this section are effective in all the cases, and also are much more effective than previous papers, 200 samples are enough to obtain 128-bit AES key very efficiently within seconds.

4 Improved Attack on CLAFIA

4.1 CLAFIA Algorithm

CLEFIA[22] is a block cipher that has a block length of 128 bits and key lengths of 128, 192, and 256 bits. The data processing part is a four-branch generalized Feistel structure with two parallel F functions (F_0, F_1) per round. The number of respective rounds r for 128-bit, 192-bit and 256-bit keys are 18, 22 and 26. The encryption function ENC_r generates 128-bit ciphertext from 128-bit plaintext, $2r$ 32-bit round keys ($RK_{0(32)}, \ldots, RK_{2r-1(32)}$), and four 32-bit whitening keys ($WK_0, \ldots, WK_3$). The structure of ENC_r is shown in Fig. 3. ENC_r is defined as follows.

Step 1. $T_0 \mid T_1 \mid T_2 \mid T_3 \leftarrow x^{\{0,0\}} \mid (x^{\{0,1\}} \oplus WK_0) \mid x^{\{0,2\}} \mid (x^{\{0,3\}} \oplus WK_1)$

Step 2. For $i=0$ to $r-1$ do the following:

Step 2.1. $T_1 \leftarrow T_1 \oplus F_0(RK_{2i}, T_0)$, $T_3 \leftarrow T_3 \oplus F_1(RK_{2i+1}, T_2)$

Step 2.2. $T_0 \mid T_1 \mid T_2 \mid T_3 \leftarrow T_1 \mid T_2 \mid T_3 \mid T_0$

Step 3. $C^{\{r,0\}} \mid C^{\{r,1\}} \mid C^{\{r,2\}} \mid C^{\{r,3\}} \leftarrow T_3 \mid (T_0 \oplus WK_2) \mid T_1 \mid (T_2 \oplus WK_3)$

The two F functions, F_0 and F_1, have 32-bit data x and 32-bit key RK as input; they output the 32-bit data y. F_0 is defined as follows.

F0:

Step 1. $T \leftarrow RK \oplus x$

Step 2. Let $T = T_{0(8)} \mid T_{1(8)} \mid T_{2(8)} \mid T_{3(8)}$

$T_0 \leftarrow S_0(T_0)$, $T_1 \leftarrow S_1(T_1)$, $T_2 \leftarrow S_0(T_2)$, $T_3 \leftarrow S_1(T_3)$

Step 3. Let $y = y_{0(8)} \mid y_{1(8)} \mid y_{2(8)} \mid y_{3(8)}$

${}^t[y_0, y_1, y_2, y_3] = M_0 \, {}^t[T_0, T_1, T_2, T_3]$

F_1 is defined by replacing the terms in F_0 as follows: S_0 is replaced with S_1, S_1 with S_0, and M_0 with M_1. The structures of F_0 and F_1 are shown in Fig. 6. S_0 and S_1 are non-linear 8-bit S-boxes. The two matrices M_0 and M_1 are defined as

$$M_0 = \begin{pmatrix} 01 & 02 & 04 & 06 \\ 02 & 01 & 06 & 04 \\ 04 & 06 & 01 & 02 \\ 06 & 04 & 02 & 01 \end{pmatrix} \qquad M_1 = \begin{pmatrix} 01 & 08 & 02 & 0a \\ 08 & 01 & 0a & 02 \\ 02 & 0a & 01 & 08 \\ 0a & 02 & 08 & 01 \end{pmatrix}$$
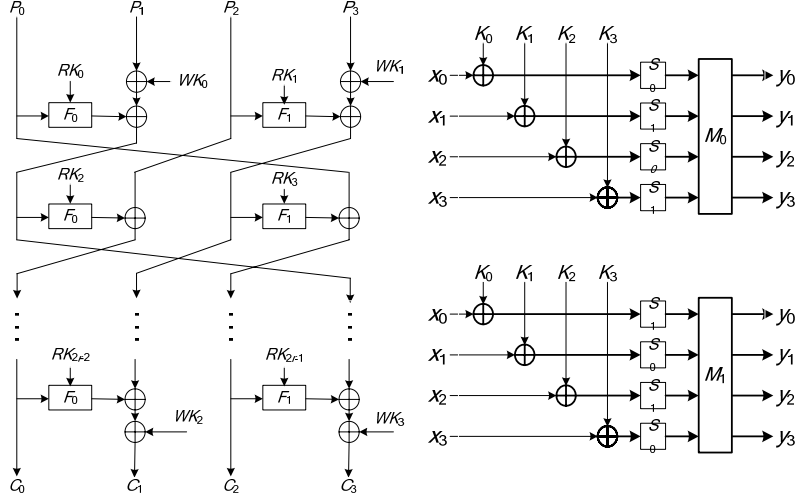
Fig. 5. CLEFIA encryption procedure

The key scheduling generates whitening keys $WK_i (0 \leq i < 4)$, and round keys $RK_j (0 \leq j < 2r)$. Let K be a k-bit key, where k is 128, 192 or 256. The key scheduling is divided into the following two sub-parts.

(1) Generating an intermediate key L from K.

(2) Expanding K and L to generate WK_i and RK_j.

The key scheduling is explained according to the sub-parts.

For the 128-bit key scheduling, the 128-bit intermediate key L is generated by applying $GFN_{4,12}$ which takes twenty-four 32-bit constant values CON_i^{128}, $0 \leq i < 24$ as round keys and $K = K_0|K_1|K_2|K_3$ as an input. Then K and L are used to generate $WK_i (0 \leq i < 4)$ and $RK_j (0 \leq j < 36)$ in the following steps.

Step 1. $L \leftarrow GFN_{4,12}(CON_0^{(128)}, ..., CON_{23}^{(128)}, K_0, ..., K_3)$

Step 2. $WK_0|WK_1|WK_2|WK_3 \leftarrow K$

Step 3. For $i = 0$ to 8 do the following:

$$T \leftarrow L \oplus (CON_{24+4i}^{(128)} \mid CON_{24+4i+1}^{(128)} \mid CON_{24+4i+2}^{(128)} \mid CON_{24+4i+3}^{(128)}); L = \sum (L);$$

if i is odd. $T = T \oplus K; RK_{4i} \mid RK_{4i+1} \mid RK_{4i+2} \mid RK_{4i+3} \leftarrow T$

The DoubleSwap function $\sum: \{0, 1\}^{128} \rightarrow \{0, 1\}^{128}$ $(X_{128} \rightarrow Y_{128})$ is defined as follows:

$$Y = X[7 - 63] \mid X[121 - 127] \mid X[0 - 6] \mid X[64 - 120]$$

4.2 Related Work

In 2009, Chester et al. propose the first Cache timing attack[10] on CLEFIA using the same technique as Bernstein, the attack is belonged to time driven type, and recover the 121 bits of the 128 bit key with $2^{26.64}$ CLEFIA encryptions on an Intel Core 2 Duo

machine. In earlier 2010, Chester et al. present an trace driven Cache timing attack[11] on CLEFIA using the same technique as the basic trace driven attack of Bertoni et. al.[6] by exploring the Cache hit trace information, successfully obtain the entire 128-bit key with 2^{43} encryptions and requires around 1 hour and 30 minutes for the complete attack. In this Section, we present a new trace driven Cache timing attack on CLEFIA by exploring both the Cache miss trace information and S-box misalignment feature, our attack are much more efficient than[10] and [11], 1000 encryptions are enough to obtain 128-bit CLEFIA key within 3 seconds.

4.3 Attack Procedure

In order to obtain CLEFIA, the attacker needs to obtain RK_0 and RK_1 through first round attack, $RK_2 \oplus WK_0$ and $RK_3 \oplus WK_1$ through the second round attack, RK_4 and RK_5 through the third round attack, and using the key expansion algorithm to reduce the CLEFIA key search space from 2^{128} to 2^7.

4.3.1 Determining RK_0 and RK_1

Each CLEFIA round has 4 times S_0 table lookups and 4 S_1 table lookups, the 8 table lookup index is given by equation (7).

$$
\begin{aligned}
&I_{S0,0}^{1,0} = P_0 \oplus RK_{0,0} && I_{S1,0}^{1,1} = P_1 \oplus RK_{0,1} \\
&I_{S0,1}^{1,2} = P_2 \oplus RK_{0,2} && I_{S1,1}^{1,3} = P_3 \oplus RK_{0,3} \\
&I_{S1,2}^{1,4} = P_8 \oplus RK_{1,0} && I_{S0,2}^{1,5} = P_9 \oplus RK_{1,1} \\
&I_{S1,3}^{1,6} = P_{10} \oplus RK_{1,2} && I_{S0,3}^{1,7} = P_{11} \oplus RK_{1,3}
\end{aligned}
\tag{7}
$$

Note that $I_{Sj,m}^{r,i}$ denotes the index to the ith $(0 \leq i)$ access to CLEFIA table lookups, and specifically the mth access to S_j table in round r.

When the mth $(1 \leq m < 4)$ access to S_j table generates a Cache miss, at most $16m$ key candidates can be eliminated. Using the attack models in Section 2.2, $RK_{0,0}, RK_{0,2}, RK_{1,0}, RK_{1,2}$ can be guessed in sequence together, $RK_{0,1}, RK_{0,3}, RK_{1,1}, RK_{1,3}$ can be guessed in sequence together.

4.3.2 Determining $RK_2 \oplus WK_0$ and $RK_3 \oplus WK_1$

As RK_0 and RK_1 can be recovered in Section 4.3.1, the 8 table lookup index of the second CLEFIA round is given by equation (8):

$$I_{S0,4}^{2,8}=P_4\oplus(WK_{0,0}\oplus RK_{2,0})\oplus F_0(RK_0,P_{(0-3)})_0 \qquad I_{S1,4}^{2,9}=P_5\oplus(WK_{0,1}\oplus RK_{2,1})\oplus F_0(RK_0,P_{(0-3)})_1 \qquad (8)$$

$$I_{S0,5}^{2,10}=P_6\oplus(WK_{0,2}\oplus RK_{2,0})\oplus F_0(RK_0,P_{(0-3)})_2 \qquad I_{S1,5}^{2,11}=P_7\oplus(WK_{0,3}\oplus RK_{2,3})\oplus F_0(RK_0,P_{(0-3)})_3$$

$$I_{S1,6}^{2,12}=P_{12}\oplus(WK_{1,0}\oplus RK_{3,0})\oplus F_1(RK_1,P_{(8-11)})_0 \qquad I_{S0,6}^{2,13}=P_{13}\oplus(WK_{1,1}\oplus RK_{3,1})\oplus F_1(RK_1,P_{(8-11)})_1$$

$$I_{S1,7}^{2,14}=P_{14}\oplus(WK_{1,2}\oplus RK_{3,2})\oplus F_1(RK_1,P_{(8-11)})_2 \qquad I_{S0,7}^{2,15}=P_{15}\oplus(WK_{1,3}\oplus RK_{3,3})\oplus F_1(RK_1,P_{(8-11)})_3$$

When the mth ($4\leq m<8$) access to S_j table generates a Cache miss, at most $16m$ key candidates can be eliminated. Using the attack model in Section 2.2, after analysis with 4 times S_0 lookup, $WK_{0,0}\oplus RK_{2,0}$, $WK_{0,2}\oplus RK_{2,2}$, $WK_{1,1}\oplus RK_{3,1}$, $WK_{1,3}\oplus RK_{3,3}$ can be guessed in sequence together, after analysis with 4 times S_1 table lookup, $WK_{0,1}\oplus RK_{2,1}$, $WK_{0,3}\oplus RK_{2,3}$, $WK_{1,0}\oplus RK_{3,0}$, $WK_{1,2}\oplus RK_{3,2}$ can be guessed in sequence together. Note that the key recovery efficiency of the second round attack is much better than the first round due to the increase of m.
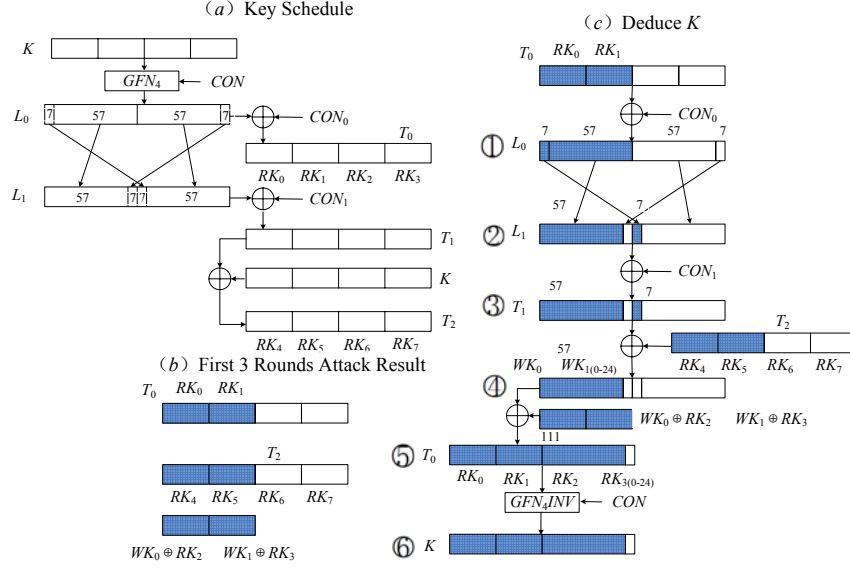
4.3.3 Determining RK_4 and RK_5

As RK_0, RK_1, $RK_2 \oplus WK_0$, $RK_3 \oplus WK_1$ can be recovered in Section 4.3.1 and 4.3.2, the 8 table lookup index of the third CLEFIA round is given by equation (9):

$$I_{S0,8}^{3,16}=F_0(F_0(RK_0,P_{(0-3)})\oplus P_{(4-7)}\oplus WK_0,RK_2)\oplus P_8\oplus RK_{4,0} \qquad (9)$$

$$I_{S1,8}^{3,17}=F_0(F_0(RK_0,P_{(0-3)})\oplus P_{(4-7)}\oplus WK_0,RK_2)\oplus P_9\oplus RK_{4,1}$$

$$I_{S0,9}^{3,18}=F_0(F_0(RK_0,P_{(0-3)})\oplus P_{(4-7)}\oplus WK_0,RK_2)\oplus P_{10}\oplus RK_{4,2}$$

$$I_{S1,9}^{3,19}=F_0(F_0(RK_0,P_{(0-3)})\oplus P_{(4-7)}\oplus WK_0,RK_2)\oplus P_{11}\oplus RK_{4,3}$$

$$I_{S1,10}^{3,20}=F_1(F_1(RK_1,P_{(8-11)})\oplus P_{(12-15)}\oplus WK_1,RK_3)\oplus P_0\oplus RK_{5,0}$$

$$I_{S0,10}^{3,21}=F_1(F_1(RK_1,P_{(8-11)})\oplus P_{(12-15)}\oplus WK_1,RK_3)\oplus P_1\oplus RK_{5,1}$$

$$I_{S1,11}^{3,22}=F_1(F_1(RK_1,P_{(8-11)})\oplus P_{(12-15)}\oplus WK_1,RK_3)\oplus P_2\oplus RK_{5,2}$$

$$I_{S0,11}^{3,23}=F_1(F_1(RK_1,P_{(8-11)})\oplus P_{(12-15)}\oplus WK_1,RK_3)\oplus P_3\oplus RK_{5,3}$$

When the mth ($8\leq m<12$) access to S_j table generates a Cache miss, at most $16m$ key candidates can be eliminated. Using the attack model in Section 2.2, after analysis with 4 times S_0 lookup, $WK_{0,0}\oplus RK_{2,0}$, $WK_{0,2}\oplus RK_{2,2}$, $WK_{1,1}\oplus RK_{3,1}$, $WK_{1,3}\oplus RK_{3,3}$ can be guessed in sequence together, after analysis with 4 times S_1 table lookup, $WK_{0,1}\oplus RK_{2,1}$, $WK_{0,3}\oplus RK_{2,3}$, $WK_{1,0}\oplus RK_{3,0}$, $WK_{1,2}\oplus RK_{3,2}$ can be guessed in sequence together. Note that the key recovery efficiency of the third round attack is much better than the first and second round due to the increase of m.

4.3.4 Recover Initial Key K

（a）Key Schedule

K

$GFN_4 \leftarrow CON$

L_0 | 7 | 57 | 57 | 7 | $\oplus \leftarrow CON_0$

T_0

RK_0 RK_1 RK_2 RK_3

L_1 | 57 | 7 7 | 57 | $\oplus \leftarrow CON_1$

T_1

K

T_2

RK_4 RK_5 RK_6 RK_7

（b）First 3 Rounds Attack Result

T_0 RK_0 RK_1

T_2

RK_4 RK_5 RK_6 RK_7

$WK_0 \oplus RK_2$ $WK_1 \oplus RK_3$

（c）Deduce K

T_0 RK_0 RK_1

$\oplus \leftarrow CON_0$

① L_0 | 7 | 57 | 57 | 7

② L_1 | 57 | 7

③ T_1 | 57 | 7

$\oplus \leftarrow CON_1$

T_2

RK_4 RK_5 RK_6 RK_7

WK_0 $WK_{1(0,24)}$

④ $\oplus$

$WK_0 \oplus RK_2$ $WK_1 \oplus RK_3$

⑤ T_0

111

RK_0 RK_1 RK_2 $RK_{3(0,24)}$

$GFN_4INV \leftarrow CON$

⑥ K

Fig. 6. CLEFIA-128 key recovery procedure

According to the key expansion algorithm, the initial key K can be obtained as shown as shown in Fig .7 and also as follows:

Step 1. Recover $L_{0(0,63)}$ by Xor between CON_0 and (RK_0,RK_1)

Step 2. Recover $L_{1(0,56)}$ by the ReveseDoubleSwap function $\sum^{-1}(L_{0(0,63)})$

Step 3. Recover $T_{1(0,56)}$ by Xor between CON_1 and $L_{1(0,56)}$

Step 4. Recover $WK_0,WK_{1(0,24)}$ by Xor between $T_{1(0,56)}$ and (RK_4,RK_5)

Step 5. Recover $RK_2,RK_{3(0,24)}$ by Xor between $WK_0,WK_{1(0,24)}$ and $(WK_0 \oplus RK_2, WK_1 \oplus RK_3)$

Step 6. Recover 111 bit of K by $GFN_4INV(RK_0,RK_1,RK_2,RK_{3(0,24)}, CON)$

4.5 Experiment Results

As to $\delta=16$, in the first round attack of CLEFIA-128, the relationship between sample size and RK_0, RK_1 key searching space for different offset ℓ is shown in Fig.7. It's clear to see that when the CLEFIA S-box is aligned perfectly ($\ell=0$), after the first round attack, the adversary can reduce RK_0, RK_1 key searching space down to 2^{40}, but if the S-box is misaligned, after the first round attack, 40 and 80 samples are enough to reduce 64-bit first round key searching space to 2^{32} and 2^{16} respectively.
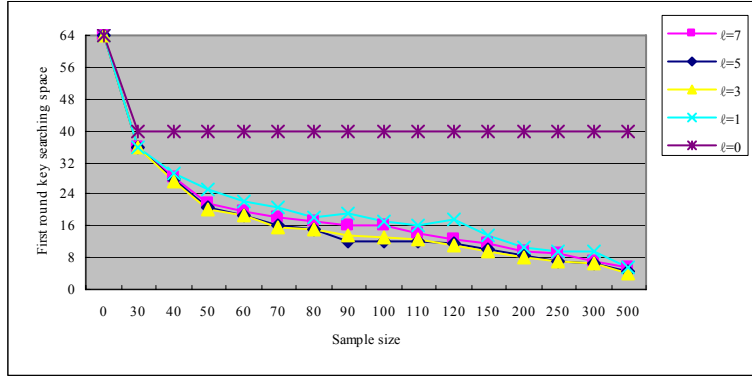
Fig. 7. Sample size and first round key searching space for different offset ℓ

Fig.8 depicts the relationship between sample size and first two rounds key searching space. It's clear to see that, 100 and 200 samples are enough to reduce first two rounds key(RK_0, RK_1, $RK_2 \oplus WK_0$, $RK_3 \oplus WK_1$) searching space from 2^{128} to 2^{16} and 2^{10} respectively.
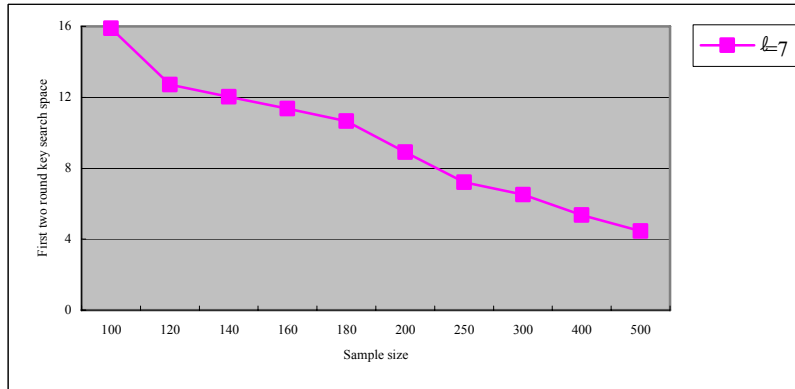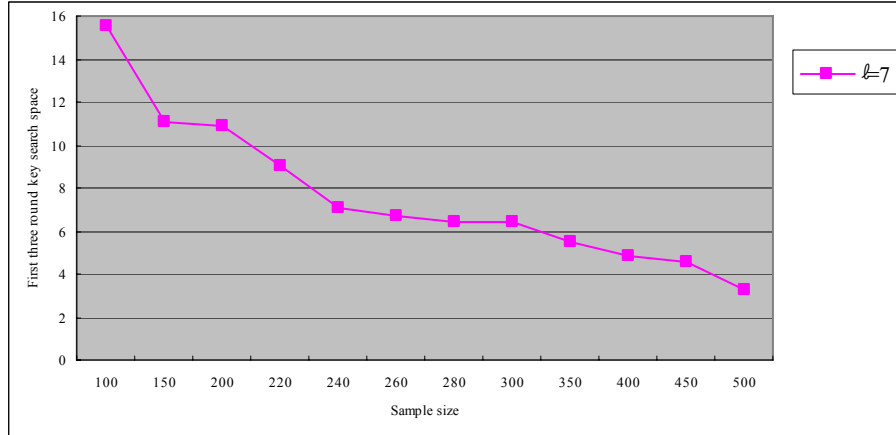

Fig. 8. Sample size and first two rounds key searching space for ℓ=7

Fig.9 depicts the relationship between sample size and first three rounds key searching space. About 220 samples are enough to reduce first three rounds key(RK_0, RK_1, $RK_2 \oplus WK_0$, $RK_3 \oplus WK_1$, RK_4 and RK_5) searching space from 2^{192} to 2^9, and as recover the initial CLEFIA key needs extra 2^7 brute force searching, so 220 samples can reduce the whole CLEFIA key searching space from 2^{128} to 2^{16}.

Fig. 9. Sample size and first three rounds key searching space for ℓ=7

As there is only one trace driven Cache attack on CLEFIA in [11].

Compared with [11], our attack has follow advantages:

(1) We do not assume the cache is completely clean prior to encryption, while attack in [11] assumes a completely "clean" cache prior to encryption. Attack in [11] means that no CLEFIA table entries are already loaded into cache. If an attacker can run code on the target machine between encryptions, this can be achieved by reading enough garbage data from memory. However, if the attacker cannot run code, there may be no way to ensure all CLEFIA tables are out of cache. A single table entry left in cache is enough to foil the attack in [11] if it assumes hits are generated only from the encryption itself, this is a significant limitation. This paper does not make a clean cache assumption and explores the effects of a partially pre-loaded cache. In particular, this means ignoring hits in the cache trace, since it is unknown whether or not they are "false hits" due to a pre-loaded cache.

（2）Attack in [11] is only applicable to the case that the S-box elements are perfect aligned case, while the attack in this paper are applicable to all the cases.

(3) Attack in [11] utilized the Cache hit information in the CLEFIA encryption to analyze the key, while the attack in this paper is focus on the Cache miss information, requires fewer samples than [11]. Compared with the 2^{43} encryptions samples in [11], only 220 samples are needed in this paper to obtain full 128-bit CLEFIA key within seconds.

5　Conclusion

This paper presents an improved Cache trace attack on AES and CLEFIA by considering Cache miss trace information and S-box misalignment. Contrary to the belief that it's impossible to obtain full 128-bit AES key and CLEFIA round key by the first round trace driven Cache attack, we show that due to the S-box misalignment feature and by considering Cache miss trace information, about 200 samples are enough to obtain full AES-128 key, 220 samples are enough to obtain full CLEFIA-

128 key. Detailed analysis and experimentation have been performed on an Athlon 3000+ CPU single core processor to establish that the AES-128 and CLEFIA-128 key can be revealed.

Acknowledgements

References

1. Paul C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. CRYPTO, volume 1109 of Lecture Notes in Computer Science, 1996, pages: 104–113, Springer. (1996).
2. P. Kocher, J.Jaffe, B. Jun. Differential power analysis[A]. Proc. of Advances in Cryptology - CRYPTO '99 (M. Wiener, ed.), Springer-Verlag, 1999, LNCS 1666: 388-397.(1999)
3. J.J. Quisquater, D. Samyde. Electromagnetic analysis (EMA): measures and countermeasures for smart cards, Smart cards programming and security (e-Smart 2001), Lectures Notes in Computer Science, vol. 2140: 200-210. Springer. (2001)
4. Shamir, A. and Tromer, E. (2004). Acoustic cryptanalysis: On nosy people and noisy machines. Rump session of EuroCrypt 2004. Available on line: http://www.wisdom.weizmann.ac.il/~tromer/acoustic/. (2004)
5. Boneh, D., DeMillo, R.A., Lipton, R.J. On the importance of checking cryptographic protocols for faults. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 37–51. Springer, Heidelberg (1997).
6. Bertoni, G., Zaccaria, V., Breveglieri, L., Monchiero, M., Palermo, G.: AES Power Attack Based on Induced Cache Miss and Countermeasure. In: ITCC (1). pp. 586–591. IEEE Computer Society (2005)
7. Acıı¸cmez, O., C¸ etin Kaya Ko¸c: Trace-driven cache attacks on aes. Cryptology ePrint Archive, Report 2006/138, Available online at http://eprint.iacr.org/2006/138.pdf. (2006)
8. O. Acııçmez, Ç.K. Koç, Trace driven cache attack on AES (short paper), in Proc. International Conference on Information and Communications Security (ICICS) 2006. LNCS, vol. 4296, pp. 112–121. Springer, Heidelberg. (2006)
9. Joseph Bonneau. Robust Final-Round Cache-Trace Attacks Against AES. Cryptology ePrint Archive, Report 2006/374, Available online at http://eprint.iacr.org/2006/374.pdf. (2006)
10. Rebeiro, C., Mukhopadhyay, D., Takahashi, J., Fukunaga, T.: Cache Timing Attacks on Clefia. In: Roy, B., Sendrier, N. (eds.) INDOCRYPT 2009. LNCS, vol. 5922, pp. 104–118. Springer, Heidelberg (2009).

11. Chester Rebeiro and Debdeep Mukhopadhyay. Differential Cache Trace Attack Against CLEFIA. Cryptology ePrint Archive, Report 2010/012, Available online at http://eprint.iacr.org/2010/012.pdf. (2010)

12. Dan Page. Theoretical use of cache memory as a cryptanalytic side-channel. Technical Report CSTR-02-003, Department of Computer Science, University of Bristol, 2002. 1-47.

13. Yukiyasu Tsunoo, Teruo Saito, Tomoyasu Suzaki, Maki Shigeri, and Hiroshi Miyauchi. Cryptanalysis of DES Implemented on Computers with Cache. In: Proc. of the Cryptographic Hardware and Embedded Systems - CHES 2003. Lecture Notes in Computer Science 2779. Berlin: Springer-Verlag, 2003. 62-76.

14. Daniel J. Bernstein. Cache-timing attacks on AES, 2005. Available online at http://cr.yp.to/papers.html\#cachetiming

15. Joseph Bonneau and Ilya Mironov. Cache-collision timing attacks against AES. In Louis Goubin and Mitsuru Matsui eds. Cryptographic Hardware and Embedded Systems - CHES 2006. Lecture Notes in Computer Science 4249. Berlin: Springer-Verlag, 2006. 201-215.

16. Colin Percival. Cache missing for fun and profit. In: Proc. of the Technical BSD Conference 2005, Ottawa, 2005. 1-13.

17. Dag Arne Osvik, Adi Shamir, and Eran Tromer. Cache attacks and Countermeasures: the Case of AES. In: Proc. of the Topics in Cryptology - CT-RSA 2006. Lecture Notes in Computer Science 3860. Berlin: Springer-Verlag, 2006. 1-20.

18. Michael Neve and J.-P. Seifert, Advances on Access-Driven Cache Attacks on AES. In: Proc. of the Selected Areas in Cryptography - SAC2006. Lecture Notes in Computer Science. Berlin: Springer-Verlag, 2007. 147-162.

19. Joan Daemen and Vincent Rijmen. The design of Rijndael: AES—the advanced encryption standard. Springer-Verlag, 2002.

20. Xin-jie ZHAO, Tao WANG, Dong MI. Robust First Two Rounds Access Driven Cache Timing Attack on AES. In: Proc. of the International Conference on Computer Science and Software Engineering - CSSE 2008. Volume 3 December 2008:785-788.

21. Xin-jie ZHAO, Tao WANG, Yuan-yuan ZHENG. Cache Timing Attacks on Camellia Block Cipher. Cryptology ePrint Archive, Report 2009/354, Available online at http://eprint.iacr.org/2009/354.pdf. (2009)

22. Shirai, T., Shibutani, K., Akishita, T., Moriai, S., Iwata, T.: The 128-bit Block cipher CLEFIA. In: Biryukov, A. (ed.) FSE 2007, vol. 4593, pp. 181–195. Springer, Heidelberg (2007)