Differential Cache Trace Attack Against CLEFIA

Chester Rebeiro and Debdeep Mukhopadhyay

Dept. of Computer Science and Engineering
Indian Institute of Technology Kharagpur, India
{*chester,debdeep*}*@cse.iitkgp.ernet.in*

Abstract. The paper presents a differential cache trace attack against CLEFIA, a 128 bit block cipher designed by Sony Corporation. The attack shows that such ciphers based on the generalized Feistel structures leak information of the secret key if the cache trace pattern is revealed to an adversary. The attack that we propose is a three staged attack and reveals the entire key with 2^{43} CLEFIA encryptions. The attack is simulated on an Intel Core 2 Duo Processor with a cache architecture with 32 byte lines as a target platform.

1 Introduction

Cache memory is a small high speed memory which stores recently used data and instructions. Data present in the cache are accessed more quickly compared to data not present in cache. Additionally, power consumption while accessing data inside the cache is less than accesses outside the cache. The differential behavior in cache memory with respect to time and power have been used to attack crypto systems [1–7, 10, 11, 13, 14, 16, 17]. Such attacks are known as cache attacks. Depending on the attackers capabilities, cache attacks are classified into three : timing attacks, access attacks, and trace attacks. In *cache timing attacks*[3, 5, 9, 16, 17] the attacker needs to have access to the encryption time. *Cache access attacks*[8, 10] require knowledge of the cache access patterns. These patterns are generally obtained from a spy process running on the same system as the cryptographic algorithm.

Cache trace attacks monitor the behavior of cache memory by means of its power consumption. Bertoni et. al. showed that power profiles of the cache reveal secret information about the cryptographic algorithm being executed [4]. Their attack was simulated on a power analysis tool and targeted on a single table AES implementation. A first round cache trace attack on the standard software implementation of AES [12] was presented in [7]. This work was extended in [1, 2] to a two round attack. A final round attack on AES was also described here.

All published cache trace attacks have targeted structures in the cipher such as Figure 1. The figure shows two accesses to table S with indices $(in_0 \oplus k_0)$ and $(in_1 \oplus k_1)$. A cache hit occurs when $(in_0 \oplus k_0) = (in_1 \oplus k_1)$. This results in lower power consumption and reveals information about the ex-or of the key

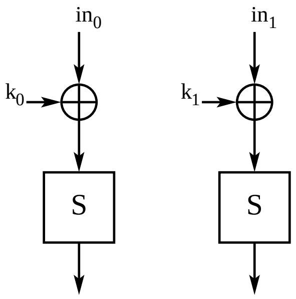

Fig. 1. Lookup Structure in AES

bits: $(k_0 \oplus k_1) = (in_0 \oplus in_1)$. In AES, in order to obtain the actual AES key the cache trace patterns for two rounds need to be considered.

CLEFIA[15] is a 128 bit block cipher developed by Sony for copyright protection and authentication. The table access pattern for CLEFIA is depicted in Figure 2. A straight forward adaptation of the existing cache trace attacks for CLEFIA is capable of revealing only the value of $k_0 \oplus k_1 \oplus k_2$, and thus has more ambiguity about the actual key value than for the AES algorithm. Hence in order to obtain the CLEFIA key, one needs to modify the techniques and supplement them using the internal round structures or properties in the key scheduling algorithm of CLEFIA.

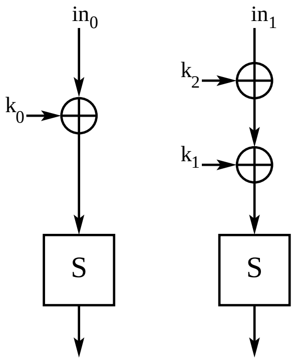

Fig. 2. Lookup Structure in CLEFIA

In this paper we present a cache trace attack for CLEFIA based on the observations made by Bertoni et. al.[4] that the power consumption gives information about the cache hit and miss. The work assumes the presence of an Oracle which returns the cache hit and miss patterns for a given plaintext. The attack makes use of two chosen plaintexts and exploits key expansion algorithm of CLEFIA. The complexity of the attack in terms of CLEFIA encryptions is 2^{43} on a cache with 32 byte cache line.

The outline of the paper is as follows: Section 2 has a brief description of the block cipher CLEFIA. Section 3 gives an outline of the proposed attack on the cipher. Section 4 presents the complexity of the attack. Section 5 has the experimental results. The work is concluded in section 6.

2 The CLEFIA Block Cipher

CLEFIA is a 128 bit block cipher with a generalized Feistel structure. The specification [15] defines three key lengths of 128, 192, and 256 bits. For brevity, this paper considers 128 bit keys though the results are valid for the other key sizes. The structure of CLEFIA is shown in Figure 3. The input has 16 bytes P_0 to P_{15}, grouped into four 4 byte words. There are 18 rounds and in each round the first and third words are fed into nonlinear functions $F0$ and $F1$ respectively. The output of $F0$ and $F1$, collectively known as F functions, are ex-ored with the second and fourth words. Additionally, the second and fourth words are also whitened at the beginning and end of the encryption.

The non-linearity in the F functions are created by two sboxes $S0$ and $S1$. These sboxes are in the form of 256 byte look-up tables, and are invoked twice in each F function, making a total of eight table look-ups per round and 144 $(= 8*18)$ look-ups per encryption. Equations for functions $F0$ and $F1$ are shown in Equation 1.

$$\begin{aligned} \mathbf{F}0 :&(y_0\ y_1\ y_2\ y_3) = (z_0\ z_1\ z_2\ z_3) \cdot M0 \\ \mathbf{F}1 :&(y_0\ y_1\ y_2\ y_3) = (z'_0\ z'_1\ z'_2\ z'_3) \cdot M1 \end{aligned} \tag{1}$$

where

$$z_0 = S0[x_0 \oplus k_0] \quad z_1 = S1[x_1 \oplus k_1] \quad z_2 = S0[x_2 \oplus k_2] \quad z_3 = S1[x_2 \oplus k_2]$$

and

$$z'_0 = S1[x_0 \oplus k_0] \quad z'_1 = S0[x_1 \oplus k_1] \quad z'_2 = S1[x_2 \oplus k_2] \quad z'_3 = S0[x_2 \oplus k_2]$$

The F functions take 4 input bytes, x_0, x_1, x_2, and x_3, and 4 round keys, k_0, k_1, k_2, and k_3. After the sbox look-ups, the bytes are diffused by multiplying them with (4×4) matrices $M0$ and $M1$ respectively. The $M0$ and $M1$ matrices are defined as follows:

$$M0 = \begin{pmatrix} 1 & 2 & 4 & 6 \\ 2 & 1 & 6 & 4 \\ 4 & 6 & 1 & 2 \\ 6 & 4 & 2 & 1 \end{pmatrix} \qquad M1 = \begin{pmatrix} 1 & 8 & 2 & A \\ 8 & 1 & A & 2 \\ 2 & A & 1 & 8 \\ A & 2 & 8 & 1 \end{pmatrix} \tag{2}$$

The CLEFIA encryption has 4 whitening keys WK_0, WK_1, WK_2, and WK_3, and 36 round keys $RK_0, \cdots, RK_{35}$. Key expansion is a two step process. First

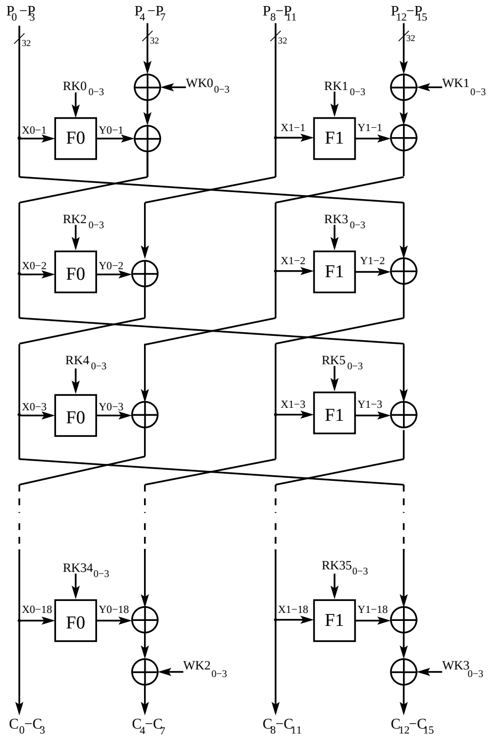

Fig. 3. CLEFIA Block Diagram

a 128 bit intermediate key L is generated from the secret key K using a GFN function [15]. From this the round keys and whitening keys are generated as shown below.

```
*Step 1:* $WK_0|WK_1|WK_2|WK_3 \leftarrow K$
*Step 2:* For $i \leftarrow 0$ to 8
    $T \leftarrow L\oplus(CON_{24+4i}|CON_{24+4i+1}|CON_{24+4i+2}|CON_{24+4i+3})$
    $L \leftarrow \Sigma(L)$
    if $i$ is odd: $T \leftarrow T \oplus K$
    $RK4i|RK4i+1|RK4i+2|RK4i+3 \leftarrow T$
```

The function Σ, known as the *double swap* function, rearranges the bits of L as shown in Equation 3.

$$\Sigma(L) \leftarrow L_{(7\cdots 63)}|L_{(121\cdots 127)}|L_{(0\cdots 6)}|L_{(64\cdots 120)} \tag{3}$$

From the structure of CLEFIA it is obvious that the knowledge of any set of 4 round keys $(RK4i, RK4i+1, RK4i+2, RK4i+3)$, where $i \bmod 2 = 0$, is sufficient to revert the key expansion process to obtain the secret key. In the attack on CLEFIA described in this paper, round keys $RK0$, $RK1$, $RK2$, and

$RK3$ are determined from which L is computed. K can then be obtained from L by the inverse GFN function.

3 The Attack

The steps involved in the attack is to first obtain the round keys $RK0$ and $RK1$ by tracing cache accesses in the first and second round. Then $RK4$ and $RK5$ are computed using the $RK0$ and $RK1$ that were obtained. From this, $RK2$ and 25 bits of $RK3$ are derived by exploiting CLEFIA's key expansion algorithm.

3.1 Determining $RK0$ and $RK1$

The sbox accesses in the first round of CLEFIA have a structure (Figure 1) favourable for cache attacks. The equations for the indices to the tables in the first round is given by:

$$\begin{aligned} I1^0_{s0} &= P_0 \oplus RK0_0 & I1^0_{s1} &= P_1 \oplus RK0_1 \\ I1^1_{s0} &= P_2 \oplus RK0_2 & I1^1_{s1} &= P_3 \oplus RK0_3 \\ I1^2_{s0} &= P_9 \oplus RK1_1 & I1^2_{s1} &= P_8 \oplus RK1_0 \\ I1^3_{s0} &= P_{11} \oplus RK1_3 & I1^3_{s1} &= P_{10} \oplus RK1_2 \end{aligned} \tag{4}$$

Note that $I\alpha^i_{s\beta}$ denotes the index to the i^{th} access to table $s\beta$ in round α.

If we make the assumption that no part of the table is present in cache before the start of encryption, then the first access to each table, ie. $I1^0_{s0}$ and $I1^0_{s1}$, results in cache misses. A maximum of 6 cache hits (3 in each table) can be obtained in the first round. This can be forced by starting with random values for P_0 and P_1, then varying P_2, P_3, P_8, P_9, P_{10}, and P_{11} in an order such that P_a is varied before P_b if and only if P_a and P_b access the same sbox and P_a accesses the sbox before P_b.

In the second round, the indices to the tables $S0$ and $S1$ in $F0$ are given by equations in (5), where $P_{(0,1,2,3)}$ indicates the concatenation of P_0, P_1, P_2, and P_3.

$$\begin{aligned} I2^0_{s0} &= P_4 \oplus WK0_0 \oplus F0(RK0, P_{(0,1,2,3)})_0 \oplus RK2_0 \\ I2^0_{s1} &= P_5 \oplus WK0_1 \oplus F0(RK0, P_{(0,1,2,3)})_1 \oplus RK2_1 \\ I2^1_{s0} &= P_6 \oplus WK0_2 \oplus F0(RK0, P_{(0,1,2,3)})_2 \oplus RK2_2 \\ I2^1_{s1} &= P_7 \oplus WK0_3 \oplus F0(RK0, P_{(0,1,2,3)})_3 \oplus RK2_3 \end{aligned} \tag{5}$$

Four cache hits can be forced in $F0$ of round two by varying P_4, P_5, P_6, and P_7 in a order as previously defined. This results in a total of 5 cache hits in table $S0$. The indices to the table are all the same, ie. $I1^0_{s0} = I1^1_{s0} = I2^0_{s0} = I2^1_{s0}$. We therefore get the following equalities.

$$\begin{aligned} P_0 \oplus P_4 &= F0(RK0, P_{(0,1,2,3)})_0 \oplus RK0_0 \oplus WK0_0 \oplus RK2_0 \\ P_2 \oplus P_6 &= F0(RK0, P_{(0,1,2,3)})_2 \oplus RK0_2 \oplus WK0_2 \oplus RK2_2 \end{aligned} \tag{6}$$

Similarly the 5 cache hits in table $S1$ result in the following equalities.

$$\begin{aligned} P_1 \oplus P_5 &= F0(RK0, P_{(0,1,2,3)})_1 \oplus RK0_1 \oplus WK0_1 \oplus RK2_1 \\ P_3 \oplus P_7 &= F0(RK0, P_{(0,1,2,3)})_3 \oplus RK0_3 \oplus WK0_3 \oplus RK2_3 \end{aligned} \tag{7}$$

For another plaintext Q, with $Q_0 \neq P_0$ and $Q_1 \neq P_1$, equations similar to (6) and (7) can be obtained by tracing cache hits in the first and second rounds. These are shown in (8).

$$\begin{aligned} Q_0 \oplus Q_4 &= F0(RK0, Q_{(0,1,2,3)})_0 \oplus RK0_0 \oplus WK0_0 \oplus RK2_0 \\ Q_2 \oplus Q_6 &= F0(RK0, Q_{(0,1,2,3)})_2 \oplus RK0_2 \oplus WK0_2 \oplus RK2_2 \\ Q_1 \oplus Q_5 &= F0(RK0, Q_{(0,1,2,3)})_1 \oplus RK0_1 \oplus WK0_1 \oplus RK2_1 \\ Q_3 \oplus Q_7 &= F0(RK0, Q_{(0,1,2,3)})_3 \oplus RK0_3 \oplus WK0_3 \oplus RK2_3 \end{aligned} \tag{8}$$

From equations in (6),(7),and (8), the fact that $P_0 \oplus P_2 \oplus P_4 \oplus P_6 = Q_0 \oplus Q_2 \oplus Q_4 \oplus Q_6$, and $P_1 \oplus P_3 \oplus P_5 \oplus P_7 = Q_1 \oplus Q_3 \oplus Q_5 \oplus Q_7$ the following equations can be generated:

$$\begin{aligned} P_0 \oplus P_4 \oplus Q_0 \oplus Q_4 &= F0(RK0, P_{(0,1,2,3)})_0 \oplus F0(RK0, Q_{0,1,2,3)})_0 \\ P_2 \oplus P_6 \oplus Q_2 \oplus Q_6 &= F0(RK0, P_{(0,1,2,3)})_2 \oplus F0(RK0, Q_{0,1,2,3)})_2 \\ P_0 \oplus P_6 \oplus Q_0 \oplus Q_6 &= F0(RK0, P_{(0,1,2,3)})_2 \oplus F0(RK0, Q_{0,1,2,3)})_2 \\ P_2 \oplus P_4 \oplus Q_2 \oplus Q_4 &= F0(RK0, P_{(0,1,2,3)})_0 \oplus F0(RK0, Q_{0,1,2,3)})_0 \\ P_1 \oplus P_5 \oplus Q_1 \oplus Q_5 &= F0(RK0, P_{(0,1,2,3)})_1 \oplus F0(RK0, Q_{0,1,2,3)})_1 \\ P_3 \oplus P_7 \oplus Q_3 \oplus Q_7 &= F0(RK0, P_{(0,1,2,3)})_3 \oplus F0(RK0, Q_{0,1,2,3)})_3 \\ P_1 \oplus P_7 \oplus Q_1 \oplus Q_7 &= F0(RK0, P_{(0,1,2,3)})_3 \oplus F0(RK0, Q_{0,1,2,3)})_3 \\ P_3 \oplus P_5 \oplus Q_3 \oplus Q_5 &= F0(RK0, P_{(0,1,2,3)})_1 \oplus F0(RK0, Q_{0,1,2,3)})_1 \end{aligned} \tag{9}$$

$RK0$ can be determined by testing the 2^{32} possibilities against the 8 conditions in (9). In a similar way $RK1$ can be determined by analyzing cache hits in $F1$. The set of equations that should satisfy $RK1$ are in (10).

$$\begin{aligned} P_8 \oplus P_{12} \oplus Q_8 \oplus Q_{12} &= F1(RK0, P_{(8,9,10,11)})_0 \oplus F1(RK0, Q_{(8,9,10,11)})_0 \\ P_{10} \oplus P_{14} \oplus Q_{10} \oplus Q_{14} &= F1(RK0, P_{(8,9,10,11)})_2 \oplus F1(RK0, Q_{(8,9,10,11)})_2 \\ P_8 \oplus P_{14} \oplus Q_8 \oplus Q_{14} &= F1(RK0, P_{(8,9,10,11)})_2 \oplus F1(RK0, Q_{(8,9,10,11)})_2 \\ P_{10} \oplus P_{12} \oplus Q_{10} \oplus Q_{12} &= F1(RK0, P_{(8,9,10,11)})_0 \oplus F1(RK0, Q_{(8,9,10,11)})_0 \\ P_9 \oplus P_{13} \oplus Q_9 \oplus Q_{13} &= F1(RK0, P_{(8,9,10,11)})_1 \oplus F1(RK0, Q_{(8,9,10,11)})_1 \\ P_{11} \oplus P_{15} \oplus Q_{11} \oplus Q_{15} &= F1(RK0, P_{(8,9,10,11)})_3 \oplus F1(RK0, Q_{(8,9,10,11)})_3 \\ P_9 \oplus P_{15} \oplus Q_9 \oplus Q_{15} &= F1(RK0, P_{(8,9,10,11)})_3 \oplus F1(RK0, Q_{(8,9,10,11)})_3 \\ P_{11} \oplus P_{13} \oplus Q_{11} \oplus Q_{13} &= F1(RK0, P_{(8,9,10,11)})_1 \oplus F1(RK0, Q_{(8,9,10,11)})_1 \end{aligned} \tag{10}$$

3.2 Determining $RK4$ and $RK5$

$RK4$ and $RK5$ are determined by third round cache hits. To determine the first and third bytes of $RK4$, the plaintext bytes P_0, P_2, P_9, P_{11}, P_4, and P_6 are set in that order, so that cache hits are obtained in the first and second round in table $S0$. Cache hits in $S0$ in the third round $F0$ are obtained by varying P_8 and P_{10}. However several cache hits may be obtained this way due to collisions with indices in $F1$ the second round. However, the only values of P_8 and P_{10} that are of interest have collisions in $S0$ accesses of the $F0$ functions in the first, second, and third rounds. That is, $I1^0_{S0} = I1^1_{S0} = I2^0_{S0} = I2^1_{S0} = I3^0_{S0} = I3^0_{S1}$. From $I1^0_{S0} = I3^0_{S0}$ we obtain,

$$RK4_0 = P_0 \oplus P_8 \oplus RK0_0 \oplus F0(RK2, P_{(4,5,6,7)} \oplus WK0 \oplus F0(RK0, P_{(0,1,2,3)}))_0 \tag{11}$$

where $WK0 \oplus RK2$ is obtained from (6) and (7).

$$WK0 \oplus RK2 = P_{(0,1,2,3)} \oplus P_{(4,5,6,7)} \oplus F0(RK0, (P_{(0,1,2,3)}) \oplus RK0 \tag{12}$$

Similarly $RK4_2$ is computed.

$$RK4_2 = P_0 \oplus P_{10} \oplus RK0_0 \oplus F0(RK2, P_{(4,5,6,7)} \oplus WK0 \oplus F0(RK0, (P_{(0,1,2,3)}))_2 \tag{13}$$

The two other bytes, $RK4_1$ and $RK4_2$, can be similarly computed by considering cache hits in $S1$ in function $F0$ in the first, second, and third round.

$$\begin{aligned} RK4_1 &= P_1 \oplus P_9 \oplus RK0_1 \oplus F0(RK2, P_{(4,5,6,7)} \oplus WK0 \oplus F0(RK0, P_{(0,1,2,3)}))_1 \\ RK4_3 &= P_1 \oplus P_{11} \oplus RK0_1 \oplus F0(RK2, P_{(4,5,6,7)} \oplus WK0 \oplus F0(RK0, P_{(0,1,2,3)}))_3 \end{aligned} \tag{14}$$

In a similar manner, $RK5$ can be determined by considering cache hits in the first, second, and third rounds in function $F1$. The equations for computing $RK5$ are presented in (15).

$$\begin{aligned} RK5_0 &= P_0 \oplus P_8 \oplus RK0_0 \oplus F1(RK2, P_{(12,13,14,15)} \oplus WK1 \oplus F1(RK1, P_{(8,9,10,11)}))_0 \\ RK5_1 &= P_1 \oplus P_9 \oplus RK0_1 \oplus F1(RK2, P_{(12,13,14,15)} \oplus WK1 \oplus F1(RK1, P_{(8,9,10,11)}))_1 \\ RK5_2 &= P_0 \oplus P_{10} \oplus RK0_0 \oplus F1(RK2, P_{(12,13,14,15)} \oplus WK1 \oplus F1(RK1, P_{(8,9,10,11)}))_2 \\ RK5_3 &= P_1 \oplus P_{11} \oplus RK0_1 \oplus F1(RK2, P_{(12,13,14,15)} \oplus WK1 \oplus F1(RK1, P_{(8,9,10,11)}))_3 \end{aligned} \tag{15}$$

3.3 Determining *RK*2 and *RK*3

In the key expansion algorithm, if $i = 0$ then $T = RK_0|RK_1|RK_2|RK_3$, and

$$T = L \oplus (CON_{24}|CON_{25}|CON_{26}|CON_{27})$$

64 bits of the key dependent constant L can be computed using the values of RK_0 and RK_1, which were determined in the first stage of the attack.

$$(L_0|L_1) = (RK_0|RK_1) \oplus (CON_{24}|CON_{25}) \tag{16}$$

The double swap operation on L places 57 known bits of L in the lower bit positions. Let the new L after double swap be denoted by L'.

$$L'_{(0\cdots 56)} = L_{(7\cdots 63)} \tag{17}$$

Again, in the key expansion algorithm, if $i = 1$, then $T = RK_4|RK_5|RK_6|RK_7$. This is represented in equation form as

$$T = L' \oplus (CON_{28}|CON_{29}|CON_{30}|CON_{31}) \oplus (WK_0|WK_1|WK_2|WK_3) \tag{18}$$

Therefore,

$$WK0|WK1_{(0\cdots 24)} = L'(0\cdots 56) \oplus (CON_{28}|CON_{29(0\cdots 25)}) \oplus (RK_4|RK_5) \tag{19}$$

Thus it is possible to ascertain $WK0$ and 25 bits of $WK1$. $WK0 \oplus RK2$ and $WK1 \oplus RK3$ have been determined in the second step of the attack. With the knowledge of $WK0$ and $WK1_{0\cdots 24}$ the whole of $RK2$ and 25 bits of $RK3$ can be determined.

4 Analysis of the Attack

Cache memories in practice have a cache line consisting of m bytes (m is a power of 2, say 2^l). Two table indices are considered equivalent if they access the same cache line. In such cases only the $(8 - m)$ most significant bits of the indices are equal. The result is an ambiguity in the detection of cache hits. A cache hit would mean 2^l possible indices for access.

Determining $RK0$ requires cache hits in the table $S0$ for the accesses $I1^1_{s0}$, $I1^2_{s0}$, $I1^3_{s0}$, $I2^0_{s0}$, and $I2^1_{s0}$, and in the table $S1$ for accesses $I1^1_{s1}$, $I1^2_{s1}$, $I1^3_{s1}$, $I2^0_{s1}$, $I2^1_{s1}$. Finding a cache hit is done by iterating the respective plaintext byte through the 2^8 possibilities. The accesses for each table $S0$ and $S1$ are independent and can be done in parallel. Obtaining all the cache hits would therefore require $5 \times 2^8 < 2^{11}$ CLEFIA invocations. Hence for P and Q plaintexts finding all cache hits would need 2^{12} encryptions. These cache hits define values for the plaintext bytes P_2, P_3, P_4, P_5, P_6, P_7 and Q_2, Q_3, Q_4, Q_5, Q_6, Q_7. Considering the cache line, there are 2^{12l} possible combinations of these plaintext bytes. Each of these combinations have to be tested in the eight equations in (9). The tests are done for each of the 2^{32} possible $RK0$ values.

For a given set of plaintext bytes, there are at-most 16 values of $RK0$ that satisfy all conditions in (9). To understand why 16 values of $RK0$ are obtained, consider the right hand side of (9). This can be represented in the form of the diffusion stage of the $F0$ function as follow:

$$\begin{pmatrix} 1 & 2 & 4 & 6 \\ 2 & 1 & 6 & 4 \\ 4 & 6 & 1 & 2 \\ 6 & 4 & 2 & 1 \end{pmatrix} \begin{pmatrix} zp_0 \\ zp_1 \\ zp_2 \\ zp_3 \end{pmatrix} \oplus \begin{pmatrix} 1 & 2 & 4 & 6 \\ 2 & 1 & 6 & 4 \\ 4 & 6 & 1 & 2 \\ 6 & 4 & 2 & 1 \end{pmatrix} \begin{pmatrix} zq_0 \\ zq_1 \\ zq_2 \\ zq_3 \end{pmatrix} \tag{20}$$

zp_i and zq_j ($0 \leq i \leq 3$, $0 \leq j \leq 3$) are the output of the sbox for the P and Q plaintexts respectively. Because of the injective sbox, the zp's and zq's uniquely identify an $RK0$. Moreover the result of the computation (20) is not altered if bytes of zp_i are interchanged with zq_j for $i = j$. This leads to 2^4 possible combinations, thus resulting in 16 different $RK0$ keys that satisfy all the conditions in (9).

In all there are 2^{4+12l} possible values for $RK0$ and an equal number of possible values for $RK1$. The values for $RK1$ are determined by an independent process requiring 2^{12} more CLEFIA encryptions.

Determining the bytes of $RK4$ requires cache hits in $F0$ in the first, second, and third rounds. Using the plaintext bytes found previously, required cache hits in the third round are obtained by setting appropriate values to P_8, P_9, P_{10}, and P_{11}. This requires 4×2^8 CLEFIA encryptions and produces 2^{4l} possible combinations. Solving equations (12), (13), and (14) for each of the $RK0$ determined in the previous step would result in $2^{4+12l+4l}$ possible values for $RK4$. Similarly $RK5$ will have an equal number of options.

With the knowledge of $RK0$, $RK1$, $RK4$, and $RK5$, the whole of $RK2$ and 25 bits of $RK3$ is determined. There are $2^{4+16l+7}$ possible values for all bits of $RK2$ and $RK3$. The 2^7 additional keys are due to the unsolved 7 bits in $RK3$.

All the possible keys have to be checked by brute force for correctness. The total complexity of the attack in terms of CLEFIA encryptions is thus of the order of 2^{11+16l}.

5 Experimental Results

The attack was simulated using an Oracle for the CLEFIA encryptions. The Oracle is queried with the plaintext and it returns the cache access patterns for each table $S0$ and $S1$. For every modification of the plaintext byte the Oracle is queried until the desired cache access pattern is obtained. The required cache access pattern is obtained with probability one.

Once the required cache hits are obtained, equations (9) and (10) are tested for different values of $RK0$ and $RK1$. There are 2^{32} possible values for each RK, but they can be done in parallel as they are independent. Each of the possible $RK0$ and $RK1$ key values result in 2^{4l} possible $RK4$ and $RK5$ keys. Determination of $RK4$ and $RK5$ cannot be parallelized as is seen from (12), (13), (14) and (15). From this stage in the attack, there is a single $RK2$ and 128 values of $RK3$ for each ($RK0$, $RK1$, $RK4$, $RK5$) tuple.

To determine the time required for the attack, we used a CLEFIA reference code where the sbox elements are *unsigned int* values. The objective of keeping *unsigned int* is justified to increase the speed of encryptions[16]. Most embedded microcontroller systems have a 32 byte cache line. The number of table elements sharing one cache line is therefore 4, or $l = 2$, yielding an attack complexity of 2^{43} encryptions. On an Intel Core 2 Duo the time for this attack took 1 hour and 30 minutes.

6 Conclusion

In this paper we have shown that the block cipher CLEFIA can be attacked using cache trace patterns. We proposed a three staged attack to ascertain the key. We have further analyzed the attack to explain the reductions in key space achieved by the steps and corresponding equations obtained in the attack. The entire attack has been simulated on a reference CLEFIA code on an Intel Core 2 Duo machine, targeting as 32 byte cache architecture. The complexity of the attack has been shown to be 2^{43} and requires around 1 hour and 30 minutes for the complete attack.

References

1. Acıiçmez, O., Çetin Kaya Koç: Trace-driven cache attacks on aes. Cryptology ePrint Archive, Report 2006/138 (2006), http://eprint.iacr.org/
2. Aciiçmez, O., Çetin Kaya Koç: Trace-Driven Cache Attacks on AES (Short Paper). In: Ning, P., Qing, S., Li, N. (eds.) ICICS. Lecture Notes in Computer Science, vol. 4307, pp. 112–121. Springer (2006)
3. Bernstein, D.J.: Cache-timing attacks on AES. Tech. rep. (2005)
4. 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)
5. Bonneau, J., Mironov, I.: Cache-Collision Timing Attacks Against AES. In: Goubin, L., Matsui, M. (eds.) CHES. Lecture Notes in Computer Science, vol. 4249, pp. 201–215. Springer (2006)
6. Kelsey, J., Schneier, B., Wagner, D., Hall, C.: Side Channel Cryptanalysis of Product Ciphers. J. Comput. Secur. 8(2,3), 141–158 (2000)
7. Lauradoux, C.: Collision attacks on processors with cache and countermeasures. In: Wolf, C., Lucks, S., Yau, P.W. (eds.) WEWoRC. LNI, vol. 74, pp. 76–85. GI (2005)
8. Neve, M., Seifert, J.P.: Advances on Access-Driven Cache Attacks on AES. In: Biham, E., Youssef, A.M. (eds.) Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 4356, pp. 147–162. Springer (2006)
9. Neve, M., Seifert, J.P., Wang, Z.: A Refined Look at Bernstein's AES Side-Channel Analysis. In: Lin, F.C., Lee, D.T., Lin, B.S., Shieh, S., Jajodia, S. (eds.) ASIACCS. p. 369. ACM (2006)
10. Osvik, D.A., Shamir, A., Tromer, E.: Cache Attacks and Countermeasures: The Case of AES. In: Pointcheval, D. (ed.) CT-RSA. Lecture Notes in Computer Science, vol. 3860, pp. 1–20. Springer (2006)
11. Page, D.: Theoretical Use of Cache Memory as a Cryptanalytic Side-Channel (2002)
12. Paulo S. L. M. Barreto: The AES Block Cipher in C++, http://www.larc.usp.br/ pbarreto/AES++.zip
13. Rebeiro, C., Mondal, M., Mukhopadhyay, D.: Pinpointing Cache Timing Attacks on AES. In: VLSI Design. IEEE Computer Society (2010), *To Appear*
14. Rebeiro, C., Mukhopadhyay, D., Takahashi, J., Fukunaga, T.: Cache Timing Attacks on Clefia. In: Roy, B., Sendrier, N. (eds.) INDOCRYPT. Lecture Notes in Computer Science, vol. 5922. Springer (2009), *To Appear.*

15. Sony Corporation: The 128-bit Blockcipher CLEFIA : Algorithm Specification (2007)
16. Tsunoo, Y., Saito, T., Suzaki, T., Shigeri, M., Miyauchi, H.: Cryptanalysis of DES Implemented on Computers with Cache. In: Walter, C.D., Çetin Kaya Koç, Paar, C. (eds.) CHES. Lecture Notes in Computer Science, vol. 2779, pp. 62–76. Springer (2003)
17. Tsunoo, Y., Tsujihara, E., Minematsu, K., Miyauchi, H.: Cryptanalysis of Block Ciphers Implemented on Computers with Cache. In: International Symposium on Information Theory and Its Applications. pp. 803–806 (2002)