Differential Fault Analysis of Twofish

Sk Subidh Ali and Debdeep Mukhopadhyay

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

Abstract. In this paper we propose Differential Fault Analysis (DFA) of Twofish which was one of the five AES finalists. It uses the concept of key-dependent S-boxes and Pseudo-Hadamard Transform, which make the cipher secure against differential attack. Each S-box is dependent on key because of which the S-box is not known to the attacker. Therefore, the existing DFA techniques which use the differential properties of S-box are not directly applicable to Twofish. We propose DFA based on an approximation technique. The attack retrieves the secret key using around 320 pairs of fault-free and faulty ciphertexts with attack time complexity of 2^{40}. To the best of author's knowledge this is the first time a DFA attack is proposed on a cipher like Twofish which uses key-dependent S-box.

Keywords: AES, Twofish, Differential Fault Analysis, DFA, Fault Model.

1 Introduction

Modern day ciphers are constructed to save guard against known classical cryptanalysis techniques. But when these ciphers are implemented on hardware platforms such as smart cards, may leak information in the form of side-channels [1]. The attack which uses these implementation based weakness of the ciphers are known as side-channel cryptanalysis. There is another kind of attack which induces faults into the crypto-devices and then analyzes the faulty output of the cipher to ascertain the secret key. Fault based attacks were originally introduced in [2] to break the RSA crypto-system. Subsequently, a more strong form of the attack, known as Differential Fault Analysis (DFA), was proposed in [3]. DFA uses the concepts of conventional differential attack in context to fault attack. The first DFA was mounted on DES crypto-systems and the result showed that the secret key of DES can be retrieved by analysing 50 to 200 faulty ciphertexts generated from known but related plaintexts.

DFA gained significant attention in the research community when it was shown in [4], that faults can easily be injected in a crypto-chip using some less expensive devices like flashgun or laser pointer. Afterward, DFA was mounted against many crypto-systems like AES [5–12], Triple-DES [13], CLEFIA [14,15], IDEA [16], RSA [17–19]. In the same lines there is significant research in practical fault injection techniques. The recent results show that fault can also be

M. Kutyłowski and M. Yung (Eds.): Inscrypt 2012, LNCS 7763, pp. 10–28, 2013.

injected using glitches in the clock input line [20, 21], power glitch [22], underpowering [23, 24], laser beam [25] or electromagnetic radiations [26].

In this paper we investigate DFA on Twofish [27]. Twofish is a 128 bit symmetric block cipher. It supports key length of 128 bits, 192 bits, and 256 bits. Twofish was introduced by Schneier *et al.* as one of the AES candidates. The cipher was selected as one of the five AES finalist. Currently it is being used by many applications like PGP, SSH Tectia, Sentry NT/2000/XP, SQL 2000 DBA Toolkit [28]. Still there are hardly few reported attacks on Twofish. The designers claimed that impossible differential attack is possible up to 6-rounds of Twofish [29]. A Saturation attack was proposed on reduced round of Twofish (upto 7 round with full whitening and 8 rounds with pre-whitening) using upto 2^{127} plaintexts [30]. The observations made on the key-dependent S-boxes and the differential cryptanalysis performed in [31] claimed that 8-round Twofish can be attacked. Most recently, a truncated differential cryptanalysis of Twofish was shown in [32]. The authors have found out truncated differential for 12-rounds and 16-rounds, but a complete attack was not shown based on these results. State-of-the-art shows Twofish is secure against conventional cryptanalysis techniques and because of its design poses new challenges for cryptanalysis.

Literature shows no reported fault based analysis of the Twofish cipher. However, because of its rather uncharacteristic structure of key-dependent S-boxes, combination of integer modulo operations with XORs, they pose significant challenges to fault analysis as known methods of DFA on block ciphers do not directly apply. In this paper we propose DFA of Twofish based on approximation techniques. In this paper we targeted Twofish (with 128-bit key). The proposed attack uses 320 pairs of fault-free and faulty ciphertext and uniquely determines the secret key with attack time complexity 2^{40}. Apart from the specific objective of performing DFA on Twofish, it is also a case study to show that cipher structure has an impact on the robustness against DFA.

Organization

The paper is organized as follow: We start with Section 2, where we describe the preliminaries to this paper. In Section 3, we explain the motivations behind this work. Section 4 describes the proposed DFA method whereas the proposed DFA procedure is described in Section 5. The attack analysis and the detail experimental results are given in Section 6. Finally, we conclude in Section 7.

2 Preliminaries

2.1 Twofish

Twofish is a 128-bit symmetric key block cipher. It uses 16-round Feistel network with a bijective 'F' function. The cipher supports three different key lengths of 128, 192, and 256 bits. For brevity in this paper we only consider Twofish with 128 bits key. The structure of the cipher is shown in Figure 1. The 128 bit

plaintext P is split into four 32-bit words $P_0, \ldots, P_3$ and XORed with the four words $K_0, \ldots, K_3$ of the whitening key (one rotated by 1 bit towards left) and followed by 16 rounds. Each round of the cipher, two most significant input words (one by rotating 8 bits towards left) are fed into the F function. Each F function consists of g function followed by Pseudo-Hadamard Transform (PHT) and key word addition. The g function consists of four byte-wide key-dependent S-boxes followed by linear mixing operation with the 4×4 MDS matrix. The two output words (one rotated by 1 bit towards right) of the F function are then XORed with the two least significant words of the round input. Here addition ($\boxplus$) defines the addition modulo 2^{32} operation.

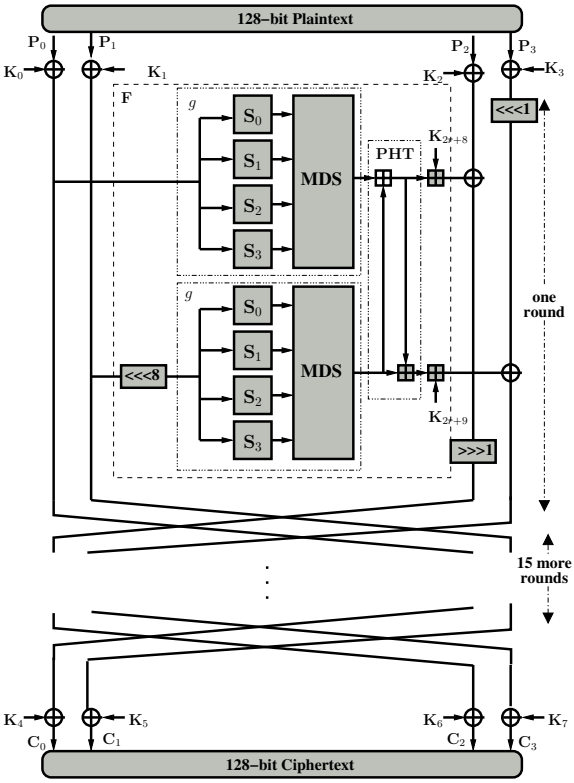

Fig. 1. Block diagram of Twofish

Pseudo-Hadamard Transform (PHT). It is a mixing operation. Given two inputs a and b, output of the PHT is defined as follows:

$$a' = a + b \; mod \; 2^{32}$$
$$b' = a + 2b \; mod \; 2^{32}$$

Key-Dependent S-Boxes. Twofish uses four key-dependent S-boxes. The four S-boxes use two 32 bits words Γ_0 and Γ_1 of the key material. The words are generated from the 128 bits Twofish key as follows:

$$\begin{pmatrix} \tau_{i,0} \\ \tau_{i,1} \\ \tau_{i,2} \\ \tau_{i,3} \end{pmatrix} = \begin{pmatrix} \cdot & \cdots & \cdot \\ \vdots & RS & \vdots \\ \cdot & \cdots & \cdot \end{pmatrix} \cdot \begin{pmatrix} k_{8i} \\ k_{8i+1} \\ k_{8i+2} \\ k_{8i+3} \\ k_{8i+4} \\ k_{8i+5} \\ k_{8i+6} \\ k_{8i+7} \end{pmatrix}$$

and,

$$\Gamma_i = \sum_{j=0}^{3} \tau_{i,j}.2^{8j}$$

where $i \in \{0, 1\}$ and $k_0, \ldots, k_{15}$ are the 16 bytes of the key. The RS matrix is given as follows:

$$RS = \begin{pmatrix} \texttt{01 A4 55 87 5A 58 DB 9E} \\ \texttt{A4 56 82 F3 1E C6 68 E5} \\ \texttt{02 A1 FC C1 47 AE 3D 19} \\ \texttt{A4 55 87 5A 58 DB 9E 03} \end{pmatrix}$$

The four S-boxes are generated as follows:

$$\begin{aligned} y_0 &= q_1[q_0[q_0[x_0] \oplus \tau_{0,0}] \oplus \tau_{1,0}] \\ y_1 &= q_0[q_0[q_1[x_1] \oplus \tau_{0,1}] \oplus \tau_{1,1}] \\ y_2 &= q_1[q_1[q_0[x_2] \oplus \tau_{0,2}] \oplus \tau_{1,2}] \\ y_3 &= q_0[q_1[q_1[x_3] \oplus \tau_{0,3}] \oplus \tau_{1,3}] \end{aligned}$$

q_0 and q_1 are fixed 8-bit permutations and $X = \{x_0, \ldots, x_3\}$ and $Y = \{y_0, \ldots, y_3\}$ are the input and output words of the S-boxes, of dimension 8 bits each.

***h*-Function.** h function plays an important role in Twofish cipher. The function is used in the key schedule as well as to derive the g function. Figure 2 shows an overview of the function. It takes the 32 bit input and a list $L = (L_0, L_1)$ of two 32-bit words and applies the S-box operation on the input where the list L, is used in reversed order. The S-box output is followed by a linear mixing operation with the MDS matrix. It can also be observed that the path of each 8 bits of the input in the function h can be visualized as application of either of the four S-boxes, $S_0, \ldots, S_3$ (as denoted by dotted line in Figure 2).

The MDS matrix is given as follows:

$$MDS = \begin{pmatrix} \texttt{01 EF 5B 5B} \\ \texttt{5B EF EF 01} \\ \texttt{EF 5B 01 EF} \\ \texttt{EF 01 EF 5B} \end{pmatrix}$$

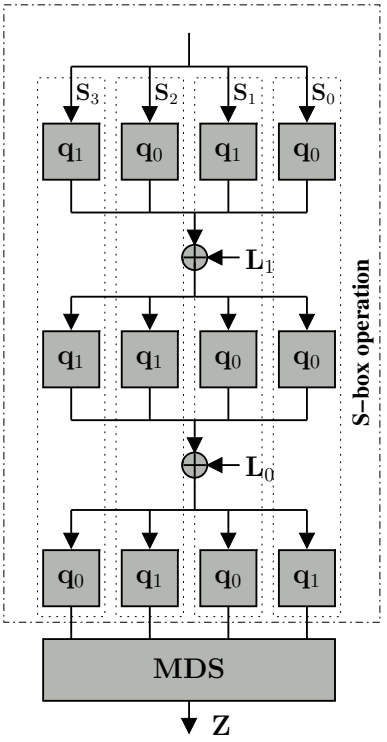

Fig. 2. The function h

Key Schedule. The key schedule provides the 40 expanded key words $K_0, \ldots, K_{39}$; first 8 key words are used for whitening purpose and the rest of the 32 key words are used in 16 rounds. The initial 128 bits key is divided into four words $W_0, \ldots, W_3$. Then these words are again divided into two list W_e, W_o, where $W_e = (W_0, W_2)$ and $W_o = (W_1, W_3)$. The expanded key words are defined as follows:

$$\begin{aligned} \rho &= 2^{24} + 2^{16} + 2^8 + 2^0 \\ A_i &= h(2i\rho, W_e) \\ B_i &= ROL(h((2i+1)\rho, W_o), 8) \\ K_{2i} &= (A_i + B_i) mod\ 2^{32} \\ K_{2i+1} &= ROL((A_i + 2B_i) mod\ 2^{32}, 9) \end{aligned}$$

where $i = 0, \ldots, 19$ and ROL is a rotation function that rotates its first argument towards left by the number of bits specified in the second argument. More details of the cipher can be found in [27].

2.2 Notation

In this section we define some parameters that we will use in rest of the paper.

$C = \{C_0, \ldots, C_3\}$: The 128 bit fault-free ciphertext, where $C_0 \ldots C_3$ are four eight byte words of the fault-free 128 bit ciphertext.

$C^* = \{C_0^*, \ldots, C_3^*\}$: The 128 bit faulty ciphertext. Here $C_0^* \ldots C_3^*$ are four eight byte words of the 128 bit faulty ciphertext.

$X_{(i)}$: refers the i^{th} byte of the word X, where a word is of 4 bytes.

3 What Makes DFA on Twofish Different from Other Ciphers?

In order to understand why DFA on Twofish is different with respect to other studied ciphers, we first observe a generalized working of DFA on these reported cryptographic algorithms. DFA uses both the concept of differential attacks and fault attacks together. The attacker is expected to induce a fault into a particular round of encryption in-order to generate certain differences. Then following the differential characteristic, the attacker deduces some differential equations related to the input-output differential of the S-box. As the S-box is known to the attacker, therefore, he can get the input of the S-box using the difference distribution table which in turn gives the key.

Figure 3 shows the basic structure of r-round Simple Permutation Network (SPN) cipher with block length n-byte. The i^{th} round consists of confusion layer S, and diffusion layer D^i, followed by an addition with the i^{th} round key K^i. There is an addition with the whitening key WK at the beginning of the encryption called key-whitening phase. The confusion layer is generally provided by S-box operation which is a non-linear transformation. The diffusion layer is provided by some linear transformation like multiplication with MDS matrix followed by a rotation operation. Due to the diffusion operation the induced fault spreads to more number of bytes which depends on the branch number of the diffusion layer. For example in AES, inducing a single byte difference at the input of diffusion layer will spread to four bytes at the output as the branch number of the diffusion layer is five [33]. The attacker uses this property in-order to generate differential equations.

Suppose that a single byte fault is induced at the input of $(r-1)^{th}$ round and the corresponding difference at the input of D_{r-1} is α. If the branch number of the diffusion layer is b then the input byte-fault will spread to $b-1$ bytes $(\alpha_{\pi_0}, \ldots, \alpha_{\pi_{b-2}})$ at the output of D_{r-1}, where π denotes the transformation of the diffusion layer. Therefore, the attacker can represent this output bytes in terms of a pair of fault-free and faulty ciphertexts (C, C^*) as follows:

$$\alpha_{\pi_j} = S^{-1}(C_{\pi_j} \oplus K^r_{\pi_j}) \oplus S^{-1}(C^*_{\pi_j} \oplus K^r_{\pi_j}) \quad (1)$$

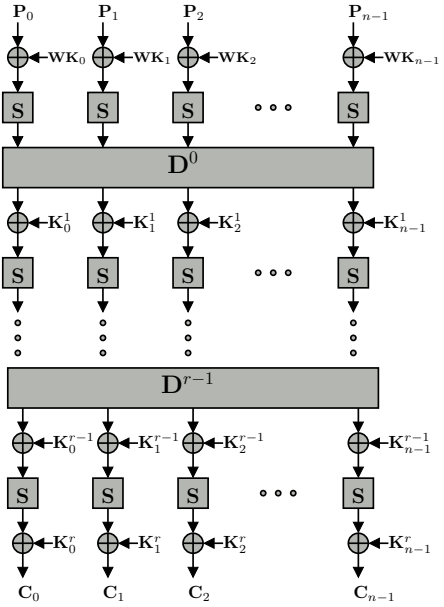

Fig. 3. Basic structure of SPN ciphers

where $j \in \{0, \ldots, b-2\}$ and S^{-1} represents the inverse of the S-box operation. Now the attacker knows the S-box input difference $C_{\pi_j} \oplus C^*_{\pi_j}$. From the difference distribution table he knows on average few values satisfy a chosen $(\alpha_{\pi_j}, C_{\pi_j} \oplus C^*_{\pi_j})$ pair. Further, because of the linear mapping in D_{r-1}, α_{π_j} depends linearly on α. Therefore, the attacker guesses the value of α and get the values of α_{π_j} i.e. the output differences. Using the input-output difference he retrieves the value $C_{\pi_j} \oplus K_{\pi_j}$ from the difference distribution table of the S-box. As C_{π_j} and $C^*_{\pi_j}$ is known to the attacker, hence he can retrieve the value of K_{π_j}. Because of the S-boxes in most of the modern day ciphers, the attacker can retrieve the entire b bytes of the key using two pair of fault-free and faulty ciphertexts [6].

Same technique is also applicable for ciphers based on Generalized Feistel Network [34]. Figure 4 shows the structure of a Feistel cipher highlighting the i^{th} round. The input is divided into two parts X_i, and Z_i. The first part passes through the F after being XORed with key component, K_i. The output of F is XORed with the second part, Z_i and then is swapped. In most of the Feistel networks, the F function consists of key addition followed by an S-box operation.

In case of these ciphers, the attacker induces faults in such a way so that all the bytes of X in the last round, i.e. X_{r-1} gets corrupted. So, he can directly get the input-output difference of the S-boxes from the fault-free and the faulty ciphertext pair. Hence he can get the last round key using the difference distribution table of the S-box. Once the last round key is retrieved, he can do one round decryption and apply the technique again to get the penultimate round key. This technique is repeated until he gets sufficient amount of round keys from which he can retrieve the master key.

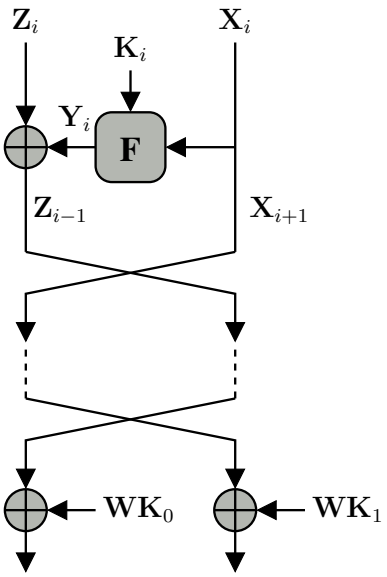

Fig. 4. Basic structure of Feistel ciphers

However, Twofish has significant difference from these structures. The first difference lies in the use of PHT operation in the F function. The PHT operation is specifically added to the design to protect the cipher against differential attacks [27]. Due to addition $mod\ 2^{32}$ operation in PHT, the difference does not pass through unchanged as in case of XOR operation. Therefore, it is impossible to obtain a suitable differential characteristic.

The second difference is in the round key addition. Unlike other Feistel ciphers, in Twofish the round key is added at the end of F function. As the addition is not at the input of S-box operation, therefore, even if the attacker retrieves the input-output difference pair of an S-box, he can not retrieve the key. Because, these differences are not directly related to the key as in case of SPN and Feistel ciphers shown above. Further, the round keys are applied by $mod\ 2^{32}$ addition.

The third and the major difference in Twofish cipher is in the use of key-dependent S-boxes. Each of the S-boxes use two bytes of the key material. Depending on the key material, there could be 2^{16} possibilities of each S-boxes. The attacker does not have the knowledge which S-box is being used. Hence he does not have the access to the difference distribution table of the corresponding S-box during the encryption.

In the next section we propose a DFA on Twofish which is based on approximation technique. In brief, we first target the key dependent S-boxes.

4 Proposed DFA Method

In this section we describe the method used to retrieve Twofish key. We show how to target the key-dependent S-boxes to the proposed DFA. We use approximation

techniques to determine the differential characteristics of the cipher. Using these differential characteristics we attack the Twofish S-box key instead of attacking the round key which is generally done in case of known DFA on SPN and Feistel ciphers as discuss in Section 3.

4.1 Fault Model Used

For the DFA on Twofish we use popular single byte fault model. The single byte fault is induced at the input of last round. The single byte difference passes through corresponding S-box and then spreads to all four bytes of the MDS output. The PHT operation will mix the difference to the two output words of the F function.

The proposed fault model can be injected in hardware design of the cipher where an attacker can precisely determine the round operation and then using techniques like glitches in the clock input line [20, 21, 35], laser beam [4], or under-powering the device [22, 36], he can induce faults.

4.2 Attack Assumptions

We made the following assumption in our attack. For the sake of simplicity we first target the S-boxes in the first g function as depicted in Figure 6.

- We assume that the attacker has the ability to induce single byte faults at any particular byte of the inputs of F function.
- The attacker does not need to know the value of the faults.

4.3 Idea of the Proposed Attack

As discussed Twofish has two important components, the $mod\, 2^{32}$ additions and key-dependent S-boxes. The proposed attack is based on two observations regarding these primitives. The first observation is the approximate differential property of the $mod\ 2^{32}$ addition and the second observation is on the key-dependent S-boxes.

Approximate Differential of Modulo Addition. As Twofish uses $mod\ 2^{32}$ addition, therefore, we can not directly get the input-output differential of an S-box from the fault-free and faulty ciphertexts. However, there is a probability that an XOR differential passes through the $mod\ 2^{32}$ operation unchanged. If such probability is p, then it is expected that after $\frac{1}{p}$ faults, one can get at least one differential characteristic where the differential remain the same across the $mod\ 2^{32}$ operation. The attacker can use this differential to get the input-output difference of the S-box.

Properties of Key-Dependent S-Box. Twofish uses key-dependent S-boxes, because of which it is difficult to obtain differential characteristics of the cipher which can be exploited for fault attacks. However, the S-box itself contains the

key materials. Each of the S-boxes consists of two bytes of the key. It can be observed that given an input-output difference pair, all 2^{16} S-boxes are not equally likely. This implies that the robustness of the cipher against differential attack are not the same for all the keys. In fact, using multiple input-output differences we can reduce the possible choices of the S-box. Finally we can uniquely determine the S-box which corresponds to a unique pair of key bytes. Once we determine all the four S-boxes, we can get the eight bytes of the key material.

4.4 Twofish S-Box Analysis

Each of the four Twofish S-boxes can be considered as a function $S_{k_0,k_1}(x)$ of x, where k_0 and k_1 are the two bytes of the key material (Figure 5) . Depending on the values of k_0 and k_1 there could be 2^{16} such functions possible. Our objective is to determine the S-box based on the input-output difference. Given an input-output difference $(\Delta_{in}, \Delta_{out})$, we can write the following differential equation:

$$\Delta_{out} = S_{k_0,k_1}(x \oplus \Delta_{in}) \oplus S_{k_0,k_1}(x) \tag{2}$$

As per the Twofish specifications, each of the possible 2^{16} S-boxes poses good differential properties. For a given value of k_0, $k_1, \Delta_{in}, \Delta_{out}$, the above differential equation will have on average one solution of x. This implies, for a given value of $(\Delta_{in}, \Delta_{out})$ on average 2^{16} out of the 2^{24} values of the triplet $\{x, k_0, k_1\}$ will satisfy the above equation. Hence, one pair of input-output difference will reduce the search space of the triplet by 2^8. Therefore, on average we need three pairs of input-output difference to reduce the search space of the triplet to a unique value.

An exhaustive search is done to validate the above analysis. Results show that using single pair of input-output difference the search space of the triplet remains within a range of 2^{15} to 2^{16}. Using three pairs, in most of the cases the search space reduces to one whereas using four pairs the search space always reduces to one.

The results suggest that if the attacker is able to get four pairs of input-output difference of an S-box, he can uniquely determine the corresponding two key bytes and the input byte. Algorithm 1 summarizes the way to recover them.

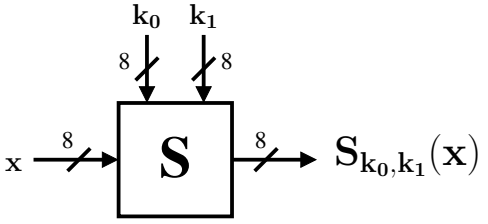

Fig. 5. Basic structure of one byte S-box

Algorithm 1. Deduce S-box key bytes k_0, k_1 and the input x

Input: $(\Delta_{in0}, \Delta_{out0}), (\Delta_{in1}, \Delta_{out1}),$
$(\Delta_{in2}, \Delta_{out2}), (\Delta_{in3}, \Delta_{out3})$
Output: k_0 , k_1 x

Solve the following simultaneous equations $\Delta_{out0} = S_{k_0,k_1}(x \oplus \Delta_{in0}) \oplus S_{k_0,k_1}(x)$
$\Delta_{out1} = S_{k_0,k_1}(x \oplus \Delta_{in1}) \oplus S_{k_0,k_1}(x)$
$\Delta_{out2} = S_{k_0,k_1}(x \oplus \Delta_{in2}) \oplus S_{k_0,k_1}(x)$
$\Delta_{out3} = S_{k_0,k_1}(x \oplus \Delta_{in3}) \oplus S_{k_0,k_1}(x)$
if k_0, k_1, *and* x *are uniquely determined* **then**
  return k_0 *and* k_1
end
else error

4.5 Determining the Approximate Differential of Modulo Addition

The PHT transformation was added in the design to thwart differential attack. The differential characteristic across the addition modulo 2^{32} is not the same as in case of XOR operation. There is a analysis on the differential properties of addition modulo 2^n given in [37,38]. However, in our case the required differential equation is different. We assume a single byte fault is induced at the first input of the F function. Figure 6 shows the flow of faults where the byte-fault is induced at the least significant byte of the first input word of the F function. The two input words of the two g functions are X_0 and X_1, and the corresponding output words are Y_0 and Y_1 respectively. Z_0 and Z_1 refer to the output words of the F function. The single byte fault is induced at the first byte of X_0 and the corresponding fault value is referred as f.

After the S-box operation f changes to f' and subsequently spreads to all four bytes of the MDS output. The difference at the output of the MDS is given by $\alpha = (5Bf'|5Bf'|EFf'|f')$ where 1, $5B$, EF, and EF are the elements of the first column of the MDS matrix in hexadecimal format. These four-byte fault value again changes to $\beta = (d|c|b|a)$ after the round key addition. Now, we would like to find the probability of $\alpha = \beta$, which means the difference across the PHT and key addition remain the same. The relation between α and β is given by following equation:

$$\beta = (Y_0 + Y_1 + K_{38}) \oplus ((Y_0 \oplus \alpha) + Y_1 + K_{38}) \tag{3}$$

Now consider the operation: $S = Y_0 + Y_1 + K_{38}$ and $S' = Y_0 \oplus \alpha + Y_1 + K_{38}$. Substitute, $Y_1 + K_{38} = Y_1' \Rightarrow S = Y_0 + Y_1'$ and $S' = Y_0 \oplus \alpha + Y_1'$

The integer additions $y = x + k \ mod \ 2^{32}$ can be approximated as follows: $y[i] = x[i] \oplus k[i] \oplus k[i-1]$ with probability $\frac{3}{4}$ [39], where $y[i]$, $x[i]$, $k[i]$ represent the i^{th} bit of y, x, and k.
$\Rightarrow S[i] = Y_0[i] \oplus Y_1'[i] \oplus Y_1'[i-1]$
and $S'[i] = Y_0[i] \oplus \alpha[i] \oplus Y_1'[i] \oplus Y_1'[i-1]$
Thus, $\beta[i] = S[i] \oplus S'[i] = \alpha[i]$ holds with probability $\frac{3}{4}$. It may be noted that $\beta[0] = \alpha[0]$ occurs with probability one. Therefore, α and β match in the first

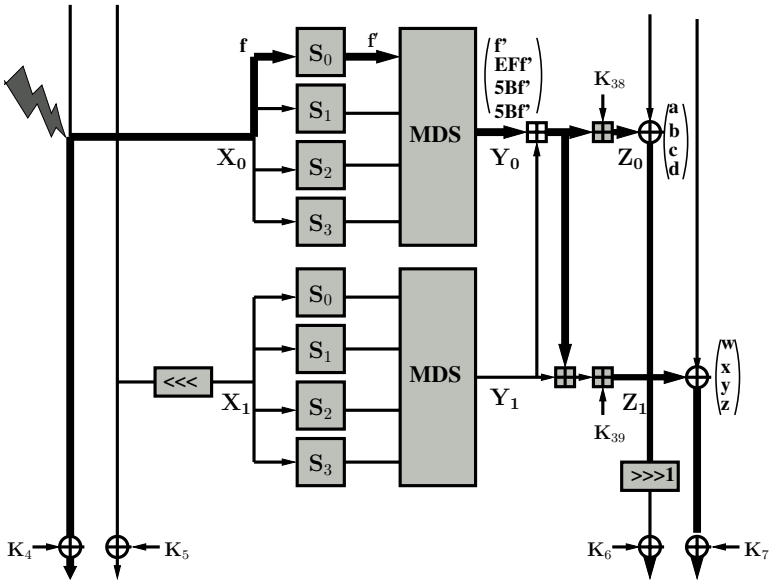

Fig. 6. Flow of single byte fault induced at the last round input

eight bits with probability $(3/4)^7 \approx 0.13$. In fact, we observe experimentally that for a random choice of Y_0, Y_1, K_{38}, and α, the probability[1] that α and β matches in the first byte is 0.13177.

This implies that if we induce ten random byte-faults at the input of the last round, there is a high chance that at least in one case the first byte (the least significant byte) of α and β matches. The value of β can be calculated from the fault-free and faulty ciphertexts. Therefore, if there is a hit in the first byte of α and β, then we can get the first byte of α from the fault-free and faulty ciphertexts. Therefore, we can get the input-output difference of the corresponding S-box. We call this input-output difference as exploitable input-output difference. The value of the first byte of α is f' when the byte-fault is induced at the first byte of X_0, as depicted in Figure 6.

From the MDS matrix we can also say that when the byte-fault is induced at the second, third, and fourth bytes, the value of the first byte of α will be $5Bf'$, EFf', and EFf' respectively where f' refers to the corresponding S-box output difference. From the above results we can say that in-order to get one exploitable input-output difference of an S-box, the attacker has to induce on average ten random byte-faults.

In-order to determine the S-box keys of the second g function we have to apply little different approach. In this case the fault is in second word of the F function. Therefore, in this case equation (3) will changed to

[1] We perform 100000 simulation. On an average in 13177 cases the match was found in the least significant byte of α and β .

$$\beta' = (Y_0 + 2Y_1 + K_{39}) \oplus (Y_0 \oplus 2(Y_1 \oplus \alpha') + K_{39}) \quad (4)$$

where α' and β' are the fault values corresponding to the second g function. It may be observed that here Y_1 and $Y_1 \oplus \alpha'$ are multiplied by 2. Therefore, in-order to get the first byte of α' from β' we have to test the cases when first byte of $(\beta' >>> 1)$ i.e. one bit right shift of β', matches with the first byte of α. This implies we have to consider first byte of $(\beta' >>> 1)$ as the possible output difference to the corresponding S-box. Rest of the analysis is the same as the of first g function.

5 The Proposed DFA Procedure

In this section we propose a DFA on Twofish using around 320 faulty ciphertexts. The proposed attack is described in three steps. In the first step the faulty ciphertexts are collected to get the input-output differences of the corresponding S-box. In the second step the two key bytes of each S-box is retrieved. In the third step the 128-bit Twofish key is recovered from the S-box key.

5.1 Getting the Input-Output Differences

For each of the S-box S_i where $0 \leq i \leq 3$, repeatedly induce byte-faults so that the faults affect the targeted S-box input. A list L_i is maintained corresponding to each S-box S_i. The lists will hold the input-output differences of the corresponding S-boxes. For j^{th} fault induction at the i^{th} S-box, the input-output difference is extracted from the fault-free and faulty ciphertexts pair (C, C^*) as follows:

$$\begin{aligned} \Delta_{in_j} &= C_{0(i)} \oplus C^*_{0(i)} \\ \Delta_{out_j} &= ROL(C_{2(i)} \oplus C^*_{2(i)}) \\ L_i[j] &= \{\Delta_{in_j}, \Delta_{out_j}\} \end{aligned}$$

Once we have the list of input-output differences, we can recover the S-box key using the differential properties of the S-box as explained in Section 4.4.

5.2 Retrieving the S-Box Key

It is already described in Section 4.4, that if the attacker is able to retrieve four input-output differences, he can uniquely determine the two key bytes of the S-box. However, in this regard the attacker has a list of input-output differences and it is not known which are the exploitable input-output differences. The attacker only expect that out of ten input-output differences there is at least one exploitable input-output difference. Therefore, he makes exhaustive search on all possible differences. In-order to do that he performs following three steps

Step 1. Choose any possible four input-output differences from L_i.
Step 2. Apply Algorithm 1 to the four input-output differences to determine corresponding S-box key. If the differences produce a pair of key bytes (k_0, k_1) and input x of corresponding S-box S_i, store it in the list SK_i.
Step 3. Repeat Step 1 and Step 2 for all possible four pairs of differences generated from the list L_i.

The attacker is expected to induce ten faults in-order to get one pair of exploitable input-output difference. Therefore, in-order to get four such input-output differences for a particular S-box, the attacker must induce at least forty faults. Therefore, L_i must contain at least forty input-output differences. From a set of forty elements one can choose four elements in ${}^{40}C_4$ ways. This implies, that the Step 1 and 2 is repeated for at least ${}^{40}C_4$ times. We get 2^8 hypotheses of (x', k_0, k_1) corresponding to each S-box.

The same technique is also followed to retrieve the S-box key from the second g function. Therefore, from two g functions we get two sets of values of each pair of S-box key bytes. We take the intersection of these two set based on the pair of key bytes and uniquely determine it . From, the unique pair of S-box key we also determine the S-box inputs x and x' corresponding to the two g functions. Following the same technique for all the S-boxes we uniquely determine the S-box key as well as the input to the F function.

5.3 Recovering Master Key

Now by applying the techniques shown in the previous section, we get the two input words X_0, and X_1 of the F function and the S-box key. We also know the ciphertexts. Therefore, using these two input words we can retrieve the two whitening keys K_4 and K_5, as $K_4 = X_0 \oplus C_0$ and $K_5 = X_1 \oplus C_1$, where C_0 and C_1 are the most significant two words of the ciphertext.

Again using X_0 and X_1 we get the two output words Y_0 and Y_1, of the F function. Now we can get following two relations of the two round keys K_{38} and K_{39},

$$\begin{aligned} \beta &= (Y_0 + Y_1 + K_{38}) \oplus (Y_0 \oplus \alpha + Y_1 + K_{38}) \\ \beta' &= (Y_0 + 2Y_1 + K_{39}) \oplus (Y_0 + 2(Y_1 \oplus \alpha') + K_{39}) \end{aligned} \tag{5}$$

, where (α, β) and (α', β') correspond to any exploitable input-output differences we already determined while retrieving the S-box keys. By solving the above two equations we get the least significant 31 bits of K_{38} and K_{39} [16, §6.1]. we get total of four hypotheses of (K_{38}, K_{39}).

In-order to get the master key from the whitening keys and the round keys we use the Twofish key schedule. Using the key schedule we get the values of (A_2, B_2) from the value of (K_4, K_5). Similarly, we get the values of (A_{19}, B_{19}) from (K_{38}, K_{39}). If we see the h function in Figure 2 we observe it is an S-box operation followed by a MDS operation. Therefore, if we do inverse MDS operation on A_i we get the S-box output where W_e is the S-box key. The input to the S-box is the value $2i\rho$. Therefore, we know the input output pair of the S-box.

We have two values of A_i i.e. A_2, A_{19} and corresponding two input output pairs. Say, the two input output pairs of a particular eight bits S-box are (In_0, Out_0) and (In_1, Out_1). In-order to get the corresponding pair of S-box key we solve following two equations

$$\begin{aligned} Out_0 &= S_{k_0,k_1}(In_0) \\ Out_1 &= S_{k_0,k_1}(In_1) \end{aligned} \tag{6}$$

Here k_0 and k_1 are the two S-box key bytes. The probability that a value of (k_0, k_1) satisfies the above two equations is $(\frac{1}{2^8})^2$. We have 2^{16} hypotheses of (k_0, k_1), of which only $\frac{2^{16}}{2^{16}} = 1$ is expected to satisfy the above two equations. We apply this technique to all the four S-boxes and determine the possible values of W_e.

It may be observed that we have four possible choices of A_2, A_{19} corresponding to four choices of K_{38}, K_{39}. As per the above analysis the expected number of value of W_e is four and rest will be discarded. We follow the same technique to determine the values of W_o from the four possible choices of B_2, B_{19}. By ordering the words of W_e and W_o we get the master key. This implies we will get $4 \times 4 = 16$ choices of master key. We have the plaintext, therefore, we can determine the exact master key out of the 16 choices.

6 Attack Analysis and Simulation Results

The attacker is expected to induce forty byte faults to retrieve the two key bytes of an S-box. However, from the fault-free and the faulty ciphertexts the attacker could not guess whether the pair of fault-free and the faulty ciphertexts lead to a exploitable input-output difference of the S-box. Therefore, he has to test all the faulty ciphertexts. In-order to do that he makes possible four input-output differences out of the forty differences generated from forty faulty ciphertexts. One can choose four out of forty elements in ${}^{40}C_4 = 91390 \approx 2^{16}$ possible ways. Therefore, the attacker will test Algorithm 1 for 2^{16} times. The time complexity of Algorithm 1 is 2^{24}, as the attacker has to try all possible values of x, k_0, and k_1. Therefore, for a particular S-box key recovery phase has a time complexity of $2^{16} \times 2^{24} = 2^{40}$.

For, a particular S-box the output difference is generated by inducing a difference to the input. The attacker only varies the input difference by inducing different faults at the input of the S-box. Therefore, the input x, to the S-box remains fixed. Only the input difference Δ_{in}, is varied. According to the differential properties of the S-box the input output difference mapping is one-to-one for a fixed input. Therefore, for a fixed input of an S-box, there are only 2^8 input-output differences possible. The input and output difference are two bytes which can have 2^{16} possible values. Therefore, the probability of one input-output difference satisfying an S-box with fixed input is $\frac{2^8}{2^{16}} = \frac{1}{2^8}$. Therefore, four such input-output differences satisfy the S-box with probability $\frac{1}{(2^8)^4} = \frac{1}{2^{32}}$. In our case the S-box and its input is not known. Therefore, we have to try all possible values of input and the two bytes key material of the S-box. Hence the probability is given by $\frac{(2^8)^3}{2^{32}} = \frac{1}{2^8}$.

Algorithm 2. DFA on Twofish

Input: (L_0, L_1, L_2, L_3),(L'_0, L'_1, L'_2, L'_3)
Output: 128-bit master key K

/*L_i and L'_i are the list of input-output differences corresponding to two different g-functions */

for $i = 0 \ldots 4$ **do**
  for *Each possible four differences of* L_i **do**
    Test Algorithm 1
    if (k_0, k_1, x) *found* **then**
      Save (k_0, k_1, x) in SK_i.
    end
  end
end

for $i = 0 \ldots 3$ **do**
  for *Each possible four differences of* L'_i **do**
    Test Algorithm 1.
    if (k_0, k_1, x') *found* **then**
      Save (k_0, k_1, x') in SK'_i.
    end
  end
end

for $i = 0 \ldots 3$ **do**
  for *Each elements of* SK_i **do**
    for *Each elements of* SK'_i **do**
      if (k_0, k_1) *of* SK_i *is equal to* (k_0, k_1) *of* SK'_i **then**
        Save (k_0, k_1, x, x').
      end
    end
  end
end
Get the value of X_0 and X_1 by combining the S-box inputs.
Get Y_0 and Y_1 from X_0 and X_1.
Get the corresponding (α, β) and (α', β').
Get K_4 and K_5 from $K_4 = X_0 \oplus C_0$ and $K_5 = X_1 \oplus C_1$.
Get the possible values of K_{38} and K_{39} by solving equation (5).
for *Each candidates of* (K_{38}, K_{39}) **do**
  Get (A_2, B_2), and (A_{19}, B_{19}).
  Test equations (6) for (A_2, A_{19}) and (B_2, B_{19}).
  if *Both solutions found* **then**
    Order (W_e, W_o) and get K.
  end
end

return K

We have 2^{16} choices of the four input-output differences out of which the number of candidates giving the key is $\frac{2^{16}}{2^8} = 2^8$. Each of these candidates will give one hypotheses of the pair of S-box key bytes. This implies for a particular S-box, we will have on average 2^8 hypotheses of the pair of S-box key bytes. Similarly, we get 2^8 hypotheses of the same pair of key bytes from the second g function. The intersection of these two sets will uniquely determine the two key bytes and the S-box input. Finally, we have unique choice of the four S-box

key and the two input words of the F function. From these values we uniquely determine the master key.

In-order to validate the analysis we have simulated the attack. A 3GHz Intel Core 2 Duo processor with 2GB RAM was used to perform the simulated attack. The code was written in C-programming language and compiled using gcc-4.4.3. The simulation was performed on 100 random keys. In each case forty random faults are induced in each of the S-boxes. The attack used total of 320 faulty ciphertexts and a fault-free ciphertexts. On an average the attack took 8 hours to reveal the secret key.

7 Conclusions

This is the first reported DFA on the AES finalist:Twofish. The proposed cipher, due to its integer addition and key-dependent S-boxes pose challenge to a differential analysis. The paper shows how a combination of approximation strategy and the observation that the key-dependent S-boxes make the differential properties stochastically inequivalent among the possible keys can reveal the key when the byte faults are induced in the cipher. The attack takes on average 320 faulty ciphertexts and a fault-free ciphertext to uniquely determine the master key with attack time complexity 2^{40}. The simulation result shows that the attack is indeed practical, taking around 8 hours on a standard platform.

References

1. Kocher, P.C., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
2. Boneh, D., DeMillo, R.A., Lipton, R.J.: On the Importance of Checking Cryptographic Protocols for Faults (Extended Abstract). In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 37–51. Springer, Heidelberg (1997)
3. Biham, E., Shamir, A.: Differential Fault Analysis of Secret Key Cryptosystems. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 513–525. Springer, Heidelberg (1997)
4. Skorobogatov, S.P., Anderson, R.J.: Optical Fault Induction Attacks. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 2–12. Springer, Heidelberg (2003)
5. Giraud, C.: DFA on AES. In: Dobbertin, H., Rijmen, V., Sowa, A. (eds.) AES 2005. LNCS, vol. 3373, pp. 27–41. Springer, Heidelberg (2005)
6. Piret, G., Quisquater, J.-J.: A Differential Fault Attack Technique against SPN Structures, with Application to the AES and KHAZAD. In: Walter, C.D., Koç, Ç.K., Paar, C. (eds.) CHES 2003. LNCS, vol. 2779, pp. 77–88. Springer, Heidelberg (2003)
7. Moradi, A., Shalmani, M.T.M., Salmasizadeh, M.: A Generalized Method of Differential Fault Attack Against AES Cryptosystem. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 91–100. Springer, Heidelberg (2006)
8. Mukhopadhyay, D.: An Improved Fault Based Attack of the Advanced Encryption Standard. In: Preneel, B. (ed.) AFRICACRYPT 2009. LNCS, vol. 5580, pp. 421–434. Springer, Heidelberg (2009)

9. Tunstall, M., Mukhopadhyay, D., Ali, S.: Differential Fault Analysis of the Advanced Encryption Standard Using a Single Fault. In: Ardagna, C.A., Zhou, J. (eds.) WISTP 2011. LNCS, vol. 6633, pp. 224–233. Springer, Heidelberg (2011)
10. Ali, S.S., Mukhopadhyay, D.: Differential Fault Analysis of AES-128 Key Schedule Using a Single Multi-byte Fault. In: Prouff, E. (ed.) CARDIS 2011. LNCS, vol. 7079, pp. 50–64. Springer, Heidelberg (2011)
11. Ali, S., Mukhopadhyay, D.: A Differential Fault Analysis on AES Key Schedule Using Single Fault. In: Breveglieri, L., Guilley, S., Koren, I., Naccache, D., Takahashi, J. (eds.) FDTC, pp. 35–42. IEEE (2011)
12. Ali, S., Mukhopadhyay, D.: An Improved Differential Fault Analysis on AES-256. In: Nitaj, A., Pointcheval, D. (eds.) AFRICACRYPT 2011. LNCS, vol. 6737, pp. 332–347. Springer, Heidelberg (2011)
13. Hemme, L.: A Differential Fault Attack Against Early Rounds of (Triple-)DES. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 254–267. Springer, Heidelberg (2004)
14. Chen, H., Wu, W., Feng, D.: Differential Fault Analysis on CLEFIA. In: Qing, S., Imai, H., Wang, G. (eds.) ICICS 2007. LNCS, vol. 4861, pp. 284–295. Springer, Heidelberg (2007)
15. Takahashi, J., Fukunaga, T.: Improved Differential Fault Analysis on CLEFIA. In: Breveglieri, L., Gueron, S., Koren, I., Naccache, D., Seifert, J.-P. (eds.) FDTC, pp. 25–34. IEEE Computer Society (2008)
16. Clavier, C., Gierlichs, B., Verbauwhede, I.: Fault Analysis Study of IDEA. In: Malkin, T. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 274–287. Springer, Heidelberg (2008)
17. Trichina, E., Korkikyan, R.: Multi Fault Laser Attacks on Protected CRT-RSA. In: Breveglieri, et al. (eds.) [40], pp. 75–86
18. Coron, J.-S., Giraud, C., Morin, N., Piret, G., Vigilant, D.: Fault Attacks and Countermeasures on Vigilant's RSA-CRT Algorithm. In: Breveglieri, et al. (eds.) [40], pp. 89–96
19. Pellegrini, A., Bertacco, V., Austin, T.M.: Fault-based attack of RSA authentication. In: DATE, pp. 855–860. IEEE (2010)
20. Fukunaga, T., Takahashi, J.: Practical Fault Attack on a Cryptographic LSI with ISO/IEC 18033-3 Block Ciphers. In: Breveglieri, et al. (eds.) [41], pp. 84–92
21. Agoyan, M., Dutertre, J.-M., Naccache, D., Robisson, B., Tria, A.: When Clocks Fail: On Critical Paths and Clock Faults. In: Gollmann, D., Lanet, J.-L., Iguchi-Cartigny, J. (eds.) CARDIS 2010. LNCS, vol. 6035, pp. 182–193. Springer, Heidelberg (2010)
22. Canivet, G., Maistri, P., Leveugle, R., Clédière, J., Valette, F., Renaudin, M.: Glitch and Laser Fault Attacks onto a Secure AES Implementation on a SRAM-Based FPGA. J. Cryptology 24(2), 247–268 (2011)
23. Barenghi, A., Bertoni, G., Parrinello, E., Pelosi, G.: Low Voltage Fault Attacks on the RSA Cryptosystem. In: Breveglieri, et al. (eds.) [41], pp. 23–31
24. Barenghi, A., Hocquet, C., Bol, D., Standaert, F.-X., Regazzoni, F., Koren, I.: Exploring the Feasibility of Low Cost Fault Injection Attacks on Sub-threshold Devices through an Example of a 65nm AES Implementation. In: Juels, A., Paar, C. (eds.) RFIDSec 2011. LNCS, vol. 7055, pp. 48–60. Springer, Heidelberg (2012)
25. Agoyan, M., Dutertre, J.-M., Mirbaha, A.-P., Naccache, D., Ribotta, A.-L., Tria, A.: How to flip a bit? In: IOLTS, pp. 235–239. IEEE (2010)
26. Quisquater, J.-J., Samyde, D.: Eddy current for Magnetic Analysis with Active Sensor. Springer (2002)

27. Schneier, B., Kelsey, J., Whiting, D., Wagner, D., Hall, C.: Twofish: A 128-Bit Block Cipher, http://www.schneier.com/paper-twofish-paper.pdf
28. http://www.schneier.com/twofish-products.html
29. Ferguson, N.: Impossible Differentials in Twofish. Twofish Technical Report 5 (October 5, 1999), http://www.schneier.com/paper-twofish-impossible.pdf
30. Lucks, S.: The Saturation Attack - a Bait for Twofish. Cryptology ePrint Archive, Report 2000/046 (2000), http://eprint.iacr.org/
31. Murphy, S., Robshaw, M.J.B.: Differential Cryptanalysis, Key-dependent S-boxes, and Twofish (2000), http://csrc.nist.gov/encryption/aes/round2/comments/20000515-smurphy.pdf
32. Moriai, S., Yin, Y.L.: Cryptanalysis of Twofish (II) (2011)
33. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Springer (2002)
34. Nyberg, K.: Generalized Feistel Networks. In: Kim, K., Matsumoto, T. (eds.) ASIACRYPT 1996. LNCS, vol. 1163, pp. 91–104. Springer, Heidelberg (1996)
35. Saha, D., Mukhopadhyay, D., RoyChowdhury, D.: A Diagonal Fault Attack on the Advanced Encryption Standard. Cryptology ePrint Archive, Report 2009/581 (2009), http://eprint.iacr.org/
36. Bhasin, S., Danger, J.-L., Guilley, S., Selmane, N.: Security Evaluation of Different AES Implementations Against Practical Setup Time Violation Attacks in FPGAs. In: Tehranipoor, M., Plusquellic, J. (eds.) HOST, pp. 15–21. IEEE Computer Society (2009)
37. Lipmaa, H., Moriai, S.: Efficient Algorithms for Computing Differential Properties of Addition. In: Matsui, M. (ed.) FSE 2001. LNCS, vol. 2355, pp. 336–350. Springer, Heidelberg (2002)
38. Lipmaa, H.: On Differential Properties of Pseudo-Hadamard Transform and Related Mappings (Extended Abstract). In: Menezes, A., Sarkar, P. (eds.) INDOCRYPT 2002. LNCS, vol. 2551, pp. 48–61. Springer, Heidelberg (2002)
39. Mukhopadhyay, D.: Design and Analysis of Cellular Automata Based Cryptographic Algorithms. IACR Ph.D database (2006), http://www.iacr.org/phds/?p=detail&entry=609
40. Breveglieri, L., Joye, M., Koren, I., Naccache, D., Verbauwhede, I. (eds.): 2010 Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2010, Santa Barbara, California, USA, August 21. IEEE Computer Society (2010)
41. Breveglieri, L., Gueron, S., Koren, I., Naccache, D., Seifert, J.-P. (eds.): Sixth International Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2009, Lausanne, Switzerland, September 6. IEEE Computer Society (2009)