Differential Fault Analysis of the Advanced Encryption Standard using a Single Fault

Michael Tunstall[1], Debdeep Mukhopadhyay[2], and Subidh Ali[2]

[1] Department of Computer Science, University of Bristol,
Merchant Venturers Building, Woodland Road,
Bristol BS8 1UB, United Kingdom.
tunstall@cs.bris.ac.uk
[2] Computer Sc. and Engg, IIT Kharagpur, India.
{debdeep,subidh}@cse.iitkgp.ernet.in

Abstract. In this paper we present a differential fault attack that can be applied to the AES using a single fault. We demonstrate that when a single random byte fault is induced at the input of the eighth round, the AES key can be deduced using a two stage algorithm. The first step has a statistical expectation of reducing the possible key hypotheses to 2^{32}, and the second step to a mere 2^8. Furthermore, we show that, with certain faults, this can be reduced to two key hypothesis.

Keywords: Differential Fault Analysis, Fault Attack, Advanced Encryption Standard.

1 Introduction

The Advanced Encryption Standard (AES) [21] has been a de-facto standard for symmetric key cryptography since October 2000. Smart cards and secure microprocessors, therefore, typically include implementations of AES to protect the confidentiality and the integrity of sensitive information. To satisfy the high throughput requirements of such applications, these implementations are typically VLSI devices (crypto-accelerators) or highly optimized software routines (crypto-libraries).

The use of faults to deduce a cryptographic key was first presented in September 1996 by Boneh et al. [6, 7]. This fault attack was applicable to public key cryptosystems, specifically RSA [26] when computed using the Chinese Remainder Theorem. Subsequently, the idea of analyzing faults in an implementation of a cryptographic algorithm was applied to block ciphers, such as DES [20], this technique is referred to as Differential Fault Analysis (DFA) [4].

With the reported work on inducing faults, such as optical fault induction reported in [28], research in the field of fault-based cryptanalysis of AES has gained considerable attention. Other methods for fault injection include variations in the power supply to create a glitch or spike [5], characteristics of the supplied clock [2], laser light [3] and eddy currents [27].

Several applications of DFA to AES have been reported in the literature, and most attacks exploit the properties of the encryption function. In [11], authors describe an analysis based on faults induced in one byte of the ninth round of AES that requires 250 faulty ciphertexts. An attack reported in [5] allows an attacker to recover the secret key with around 128 to 256 faulty ciphertexts. In [9], Dusart et al. show that using a fault which affects one byte anywhere between the eighth round MixColumn and ninth round MixColumn, an attacker would be able to derive the secret key using 40 faulty ciphertexts. The authors of [25] describe an attack on AES with single byte faults that requires two faulty outputs, where a fault is induced in the input of the eighth or ninth round. In [18], the authors present a fault attack

on AES when the fault is induced in a 32-bit word of AES in the ninth round. The authors propose two models for fault induction. In the first model, they assume that at least one of the bytes among the four targeted bytes are uncorrupted. While in the second model they assume that four bytes are corrupted. The former fault would require 6 faulty ciphertexts to deriver the secret key, while the latter would require around 1500 faulty ciphertexts to derive the key. Other authors have considered faults in the key schedule [29, 30], where the most recent publication has demonstrated that the secret key can be derived with two faulty ciphertexts [13].

We can note that when the assumptions are on the value of a byte (either it being faulty or uncorrupted) the number of faulty pairs is quite small. However, it is difficult to be able to affect a given value with any certainty. When numerous faulty ciphertexts are required this problem is amplified, since an attacker needs to find a method of determining which faulty ciphertexts correspond to the desired model. We can, therefore, state that the attacks that are most likely to be realizable require the least faulty ciphertexts and assumptions on the effect of the fault.

In [19] a fault attack against AES was proposed, which suggested that a secret key can be derived using a single *byte* fault induction at the input of the eighth round. The attack exploited the inter-relations between the fault values in the state matrix after the ninth round `MixColumn` operation and reduced the number of possible keys to around 2^{32}. However it may be noted that this work, like the previous fault attacks on AES does not use the effect of the fault maximally in an information theoretic sense [16]. The work proposed in this paper improves the previous fault analysis on AES-128 and reduces the key space to its minimal possible set of hypotheses attainable using a single byte fault. In this paper, we describe the extended version of this attack, where an attacker could reduce the exhaustive search to 2^8.

We also show that with certain assumptions on the fault induced the key can be reduced to two.

Notation

In this paper, multiplications are considered to be polynomial multiplications over $\mathbb{F}_{2^8}$ modulo the irreducible polynomial $x^8 + x^4 + x^3 + x + 1$. It should be clear from the context when a mathematical expression contains integer multiplication.

Organization

The paper is organized as follows: In Section 2 we describe the background to this paper. In Section 3 we describe an attack based on one of the fault models given in Section 2. In Section 4 we extend this attack by using a different model. In Section 5 we described some experimental results. In Section 6 we compare the attacks described in this paper with previous work, and we conclude in Section 7.

2 Background

2.1 The Advanced Encryption Standard

The structure of the Advanced Encryption Standard (AES) , as used to perform encryption, is illustrated in Algorithm 1. Note that we restrict ourselves to considering AES-128 and that the description above omits a permutation typically used to convert the plaintext $P = (p_1, p_2, \ldots, p_{16})_{(256)}$ and key $K = (k_1, k_2, \ldots, k_{16})_{(256)}$ into a 4×4 array of bytes, known

Algorithm 1: The AES-128 encryption function.

Input: The 128-bit plaintext block P and key K.
Output: The 128-bit ciphertext block C.

$X \leftarrow \texttt{AddRoundKey}(P, K)$
for $i \leftarrow 1$ **to** 10 **do**
  $X \leftarrow \texttt{SubBytes}(X)$
  $X \leftarrow \texttt{ShiftRows}(X)$
  if $i \neq 10$ **then**
    $X \leftarrow \texttt{MixColumns}(X)$
  end
  $K \leftarrow \texttt{KeySchedule}(K)$
  $X \leftarrow \texttt{AddRoundKey}(X, K)$
end
$C \leftarrow X$
return C

as the state matrix. For example, the 128-bit plaintext input block to AES is arranged in the following fashion

$$\begin{pmatrix} p_1 & p_5 & p_9 & p_{13} \\ p_2 & p_6 & p_{10} & p_{14} \\ p_3 & p_7 & p_{11} & p_{15} \\ p_4 & p_8 & p_{12} & p_{16} \end{pmatrix}$$

The corresponding fault free (CT) and faulty ciphertexts (CT') are respectively:

$$\mathbf{CT} = \begin{pmatrix} x_1 & x_5 & x_9 & x_{13} \\ x_2 & x_6 & x_{10} & x_{14} \\ x_3 & x_7 & x_{11} & x_{15} \\ x_4 & x_8 & x_{12} & x_{16} \end{pmatrix} \quad \mathbf{CT'} = \begin{pmatrix} x'_1 & x'_5 & x'_9 & x'_{13} \\ x'_2 & x'_6 & x'_{10} & x'_{14} \\ x'_3 & x'_7 & x'_{11} & x'_{15} \\ x'_4 & x'_8 & x'_{12} & x'_{16} \end{pmatrix}$$

where $x_i \in \{0, \ldots, 255\}$ $\forall i \in \{1, \ldots, 16\}$.

We also define the key matrix for the subkeys used in the ninth and tenth round as:

$$\mathbf{K_{10}} = \begin{pmatrix} k_1 & k_5 & k_9 & k_{13} \\ k_2 & k_6 & k_{10} & k_{14} \\ k_3 & k_7 & k_{11} & k_{15} \\ k_4 & k_8 & k_{12} & k_{16} \end{pmatrix} \quad \mathbf{K_9} = \begin{pmatrix} k'_1 & k'_5 & k'_9 & k'_{13} \\ k'_2 & k'_6 & k'_{10} & k'_{14} \\ k'_3 & k'_7 & k'_{11} & k'_{15} \\ k'_4 & k'_8 & k'_{12} & k'_{16} \end{pmatrix}$$

The encryption itself is conducted by the repeated use of a number of round functions:

- The `SubBytes` function is the only non-linear step of the block cipher. It is a bricklayer permutation consisting of an S-box applied to the bytes of the state. Each byte of the state matrix is replaced by its multiplicative inverse, followed by an affine mapping. Thus the input byte x is related to the output y of the S-Box by the relation, $y = A\,x^{-1} + B$, where A and B are constant matrices. In the remainder of this paper we will refer to the function S as the SubBytes function and S^{-1} as the inverse of the SubBytes function.
- The `ShiftRows` function is a byte-wise permutation of the state.
- The `KeySchedule` function generates the next round key from the previous one. The first round key is the input key with no changes, subsequent round keys are generated using the `SubBytes` function and XOR operations. This is shown in Algorithm 2 which shows how the r^{th} round key is computed from the $(r-1)^{th}$ round key. The value h_r is a constant defined for the r^{th} round, and $<<$ is used to denote a bitwise left shift.
- The `MixColumn` is a bricklayer permutation operating on the state column by column. Each column of the state matrix is considered as a 4-dimensional vector where each element belongs to $\mathbb{F}(2^8)$. A 4×4 matrix M whose elements are also in $\mathbb{F}(2^8)$ is used to map this column into a

new vector. This operation is applied on all the 4 columns of the state matrix. Here M and its inverse M^{-1} are defined as:

$$M = \begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix} \quad M^{-1} = \begin{pmatrix} 14 & 11 & 13 & 9 \\ 9 & 14 & 11 & 13 \\ 13 & 9 & 14 & 11 \\ 11 & 13 & 9 & 14 \end{pmatrix}$$

All the elements in M and M^{-1} are elements of $\mathbb{F}(2^8)$ expressed as a decimal digit.

– `AddRoundKey`: Each byte of the array is XORed with a byte from a corresponding array of round subkeys.

Algorithm 2: The AES-128 `KeySchedule` function.

Input: $(r-1)^{th}$ round key ($X = x_i$ for $i \in \{1, \ldots, 16\}$).
Output: r^{th} round key X.

for $i \leftarrow 0$ **to** 3 **do**
  $x_{(i<<2)+1} \leftarrow x_{(i<<2)+1} \oplus S(x_{(((i+1)\wedge 3)<<2)+4})$
end
$x_1 \leftarrow x_1 \oplus h_r$
for $i \leftarrow 1$ **to** 16 **do**
  if $(i-1) \mod 4 \neq 0$ **then**
    $x_i \leftarrow x_i \oplus x_{i-1}$
  end
end
return X

2.2 The Fault Model

The implementation of AES we target is an iterative one, as described in [1]. The literature shows that unrolled or pipelined designs of AES are unpopular because they do not allow a block cipher to operate in Output Feedback Mode (OFB) or Cipher Block Chaining (CBC) mode [17].

Since designs are typically synchronous, an attacker can expect to be able to predict at what point in time certain events take place, e.g. when a particular round commences. Moreover, the time certain events take can often be determined by analyzing a suitable side channel. For example, the power consumption of a FPGA or microprocessor has been shown to reveal the details of an implementation (e.g. see [15, 24]).

In this paper we discuss two different fault models, that we use to build a method for Differential Fault Analysis of AES.

Random Effect on One Byte. The first fault model that we consider is the same as that used in many other papers, for example [19], where we assume that the effect of an induced fault is to change one byte to a random value.

For example, an attacker could attempt to use a glitch in the clock to create a fault at the input of a particular round with a certain probability. An iterative design helps in this regard, as the attacker is able to control the timing of fault induction by simply counting the number of clock edges from the start of an encryption. Also, it may be noted that our experiments show that the registers internal to a FPGA device take a certain amount of time to change their value to the next correct value. During the migration, there is a certain amount of time where, if the clock terminates too soon, the correct value will not be written to the registers. This effect applied to a microprocessor is described in [2].

Fixing a Byte Value. The second fault that we consider is where an instruction is missed in a process and the potential effects this can have on an implementation of AES. Specifically, where this missing of instruction implies that one byte becomes a known, or predictable value.

It has been shown that a glitch in the clock or voltage applied to a microprocessor can be used to make the value returned from one specific instruction constant [2] or skip an instruction [8] respectively. In [8], the authors describe an attack where the round counter of AES is modified to reduce the number of rounds to one. In our case we will consider a more subtle effect where the loop implementing a round function is terminated early so that one byte of the current state matrix is not overwritten. In this case the value is unknown, but can be computed for a hypothesized key.

3 The Fault Analysis

In this section we define the strategy to perform a fault analysis, where we assume that an adversary has induced a fault in a byte of the input to the eighth round. The first step of the fault attack is equivalent to the analysis described in [19], and extended in the second step. We are also assuming that the fault corresponds to the first fault model discussed above where this byte becomes some random and unknown value.

3.1 The First Step of the Fault Attack

If a fault is induced in a byte of the state matrix, which is then input to the eighth round, the `MixColumn` operation at the end of the round propagates this fault to the entire column of the state. The `ShiftRow` operation at the beginning of the following round will then shift these bytes to occupy different columns. The next `MixColumn` operation will then propagate the fault to the remaining twelve bytes.

This process is shown in Figure 1 where we show the diffusion of a byte fault induced at the input of the eighth round. The XOR difference of the state matrices of the two results, one fault free and the other faulty, is shown. This is what we use as basis for a differential fault analysis.

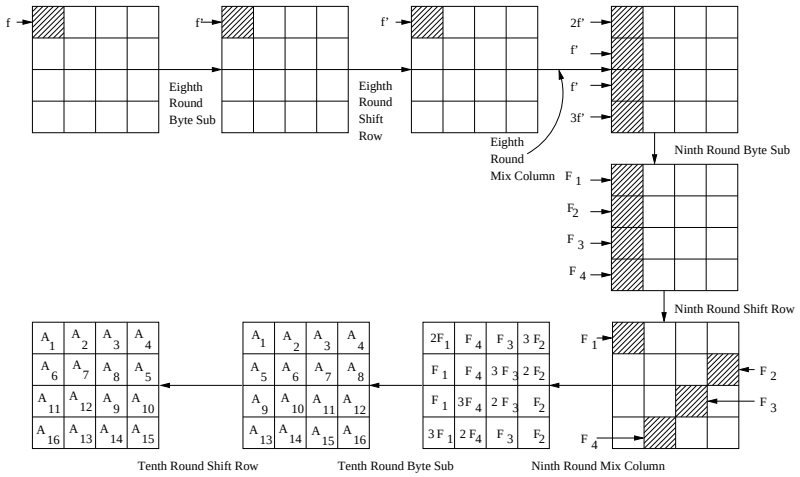

Fig. 1: Propagation of Fault Induced in the input of eighth round of AES.

If, given a fault in the input to the eighth round, we consider the state of the differences after the ninth round shift row, we can obtain the following set of equations that include the values of

the key bytes k_1, k_8, k_{11} and k_{14}, thus giving an expression for 32 bits of $\mathbf{K_{10}}$.

$$\left.\begin{aligned} 2\,\delta_1 &= S^{-1}(x_1 \oplus k_1) \oplus S^{-1}(x'_1 \oplus k_1) \\ \delta_1 &= S^{-1}(x_{14} \oplus k_{14}) \oplus S^{-1}(x'_{14} \oplus k_{14}) \\ \delta_1 &= S^{-1}(x_{11} \oplus k_{11}) \oplus S^{-1}(x'_{11} \oplus k_{11}) \\ 3\,\delta_1 &= S^{-1}(x_8 \oplus k_8) \oplus S^{-1}(x'_8 \oplus k_8) \end{aligned}\right\},$$

Where δ_1, k_1, k_8, k_{11} and k_{14} are all unknown values $\in \{0, \ldots, 255\}$.

The above system of equations can be used to reduce the possibilities for these 32 bits of the key. An attacker would select a value for δ_1 and determine which values of k_1, k_8, k_{11} and k_{14} satisfy the equations using four independent exhaustive searches. Each equation will return 0, 2, or 4 hypotheses [23]. If any of the four equations cannot be satisfied, i.e. there is an impossible differential [14], then any hypotheses for that value of δ_1 can be discarded.

As noted in [12, 18] one can apply the same technique to recover information on the remaining bytes of the last sub key. That is, information on the remaining key bytes can be derived by using the following sets of equations: In order to obtain information on k_2, k_5, k_{12} and k_{15} an attacker can use

$$\left.\begin{aligned} 3\,\delta_2 &= S^{-1}(x_5 \oplus k_5) \oplus S^{-1}(x'_5 \oplus k_5) \\ 2\,\delta_2 &= S^{-1}(x_2 \oplus k_2) \oplus S^{-1}(x'_2 \oplus k_2) \\ \delta_2 &= S^{-1}(x_{15} \oplus k_{15}) \oplus S^{-1}(x'_{15} \oplus k_{15}) \\ \delta_2 &= S^{-1}(x_{12} \oplus k_{12}) \oplus S^{-1}(x'_{12} \oplus k_{12}) \end{aligned}\right\}.$$

In order to obtain information on k_3, k_6, k_9 and k_{16} an attacker can use the following equations:

$$\begin{aligned} \delta_3 &= S^{-1}(x_9 \oplus k_9) \oplus S^{-1}(x'_9 \oplus k_9) \\ 3\,\delta_3 &= S^{-1}(x_6 \oplus k_6) \oplus S^{-1}(x'_6 \oplus k_6) \\ 2\,\delta_3 &= S^{-1}(x_3 \oplus k_3) \oplus S^{-1}(x'_3 \oplus k_3) \\ \delta_3 &= S^{-1}(x_{16} \oplus k_{16}) \oplus S^{-1}(x'_{16} \oplus k_{16}) \end{aligned}$$

Finally, in order to obtain information on k_4, k_7, k_{10} and k_{13} an attacker can use the following equations:

$$\begin{aligned} \delta_4 &= S^{-1}(x_{13} \oplus k_{13}) \oplus S^{-1}(x'_{13} \oplus k_{13}) \\ \delta_4 &= S^{-1}(x_{10} \oplus k_{10}) \oplus S^{-1}(x'_{10} \oplus k_{10}) \\ 3\,\delta_4 &= S^{-1}(x_7 \oplus k_7) \oplus S^{-1}(x'_7 \oplus k_7) \\ 2\,\delta_4 &= S^{-1}(x_4 \oplus k_4) \oplus S^{-1}(x'_4 \oplus k_4) \end{aligned}$$

It can be noted that the equations have an identical structure, and, therefore, the solutions are of similar nature. An evaluation of each set of equations will be expected to return 2^8 unique hypotheses for the key bytes concerned. Therefore, an attacker would expect to have 2^{32} key hypotheses for the secret key used.

3.2 Analysis of the first step of the fault attack

The first step of the fault attack uses four sets of equations to reduce the key space of AES. In this section we determine the expected number of key hypotheses that an attacker will have at each stage of an attack.

In order to analyze the number of valid hypotheses in the first stage of the attack we consider the first set of equations given in Section 3.1. In this set of equations δ_1 is $\in \{1, \ldots, 255\}$. If δ_1 is equal to zero then one could say that the expected fault has not been injected. If δ_1 is zero it would imply that x_1 is equal to x'_1 and all 256 key hypotheses are possible. Let us first consider the first equation in this set:

$$2\,\delta_1 = S^{-1}(x_1 \oplus k_1) \oplus S^{-1}(x'_1 \oplus k_1)$$

We know the values of x_1 and x'_1 from the correct and faulty ciphertexts respectively. For a given value of $2\,\delta_1$ there will 0, 2 or 4 valid key hypotheses. The mean hypotheses for all $\delta_1 \in \{1, \ldots, 255\}$ is approximately one, and, therefore, 256 key hypotheses when all $\delta_1 \in \{1, \ldots, 255\}$ are considered.

The same can be said for each of the four equations in the set given above. However, for a given value of δ_1 each of the four equations would be expected to return approximately one hypothesis for a key byte. These values will give one hypothesis for the quartet of key bytes $\{k_1, k_8, k_{11}, k_{14}\}$. Given that an attacker will have to take into account all the values in $\{0, \ldots, 255\}$ there will be 256 possible values for the quartet $\{k_1, k_8, k_{11}, k_{14}\}$. After an attacker has analyzed the four equations defined in Section 3.1 there would be an expected 2^{32} key hypotheses.

3.3 The Second Step of the Fault Attack

In order to further reduce the key hypotheses we use the relationship between the ninth round key and the tenth round key.

We consider the key-scheduling algorithm (see Algorithm 2), the ninth round key, K_9, generates the tenth round key, K_{10}. The key schedule is invertible and $\mathbf{K_9}$ can be expressed in terms of elements of $\mathbf{K_{10}}$. The value of $\mathbf{K_9}$ can be expressed as

$$\begin{pmatrix} k_1 \oplus S(k_{14} \oplus k_{10}) \oplus h_{10} & k_5 \oplus k_1 & k_9 \oplus k_5 & k_{13} \oplus k_9 \\ k_2 \oplus S(k_{15} \oplus k_{11}) & k_6 \oplus k_2 & k_{10} \oplus k_6 & k_{14} \oplus k_{10} \\ k_3 \oplus S(k_{16} \oplus k_{12}) & k_7 \oplus k_3 & k_{11} \oplus k_7 & k_{15} \oplus k_{11} \\ k_4 \oplus S(k_{13} \oplus k_9) & k_8 \oplus k_4 & k_{12} \oplus k_8 & k_{16} \oplus k_{12} \end{pmatrix}.$$

We can observe that the fault values in the first column of the state matrix at the output of the eighth round `MixColumn` is $(2\,f', f', f', 3\,f')$, where f' is a non-zero arbitrary value in $\mathbb{F}_{2^8}$. Using the `InverseMixColumn` operation and using the inter-relations between the fault values, we can define the following equation:

$$\begin{aligned} 2\,f' = {} & S^{-1}\Big(\mathtt{14}\left(S^{-1}(x_1 \oplus k_1) \oplus k'_1\right) \oplus \mathtt{11}\Big(S^{-1}(x_{14} \oplus k_{14}) \oplus k'_2\Big) \oplus \\ & \mathtt{13}\left(S^{-1}(x_{11} \oplus k_{11}) \oplus k'_3\right) \oplus \mathtt{9}\left(S^{-1}(x_8 \oplus k_8) \oplus k'_4\right)\Big) \oplus \\ & S^{-1}\Big(\mathtt{14}\left(S^{-1}(x'_1 \oplus k_1) \oplus k'_1\right) \oplus \mathtt{11}\Big(S^{-1}(x'_{14} \oplus k_{14}) \oplus k'_2\Big) \oplus \\ & \mathtt{13}\left(S^{-1}(x'_{11} \oplus k_{11}) \oplus k'_3\right) \oplus \mathtt{9}\left(S^{-1}(x'_8 \oplus k_8) \oplus k'_4\right)\Big) \\ = {} & S^{-1}\Big(\mathtt{14}\left(S^{-1}(x_1 \oplus k_1) \oplus ((k_1 \oplus S(k_{14} \oplus k_{10}) \oplus h_{10}))\right) \oplus \\ & \mathtt{11}\left(S^{-1}(x_{14} \oplus k_{14}) \oplus (k_2 \oplus S(k_{15} \oplus k_{11}))\right) \oplus \\ & \mathtt{13}\left(S^{-1}(x_{11} \oplus k_{11}) \oplus (k_3 \oplus S(k_{16} \oplus k_{12}))\right) \oplus \\ & \mathtt{9}\left(S^{-1}(x_8 \oplus k_8) \oplus (k_4 \oplus S(k_{13} \oplus k_9))\right)\Big) \oplus \\ & S^{-1}\Big(\mathtt{14}\left(S^{-1}(x'_1 \oplus k_1) \oplus ((k_1 \oplus S(k_{14} \oplus k_{10}) \oplus h_{10}))\right) \oplus \\ & \mathtt{11}\left(S^{-1}(x'_{14} \oplus k_{14}) \oplus (k_2 \oplus S(k_{15} \oplus k_{11})\right) \oplus \\ & \mathtt{13}\left(S^{-1}(x'_{11} \oplus k_{11}) \oplus (k_3 \oplus S(k_{16} \oplus k_{12}))\right) \oplus \\ & \mathtt{9}\left(S^{-1}(x'_8 \oplus k_8) \oplus (k_4 \oplus S(k_{13} \oplus k_9))\right)\Big) \end{aligned}$$

Similarly, we can define the following equations:

$$
\begin{aligned}
f' = S^{-1}\Big(&\mathtt{9}\left(S^{-1}(x_{13} \oplus k_{13}) \oplus (k_4 \oplus k_9)\right) \oplus \\
&\mathtt{14}\left(S^{-1}(x_{10} \oplus k_{10}) \oplus (k_{10} \oplus k_{14}))\right) \oplus \mathtt{11}\left(S^{-1}(x_7 \oplus k_7) \oplus \right. \\
&\left.(k_{15} \oplus k_{11})\right) \oplus \mathtt{13}\left(S^{-1}(x_4 \oplus k_4) \oplus (k_{16} \oplus k_{12})\right)\Big) \oplus \\
&S^{-1}\Big(\mathtt{9}\left(S^{-1}(x'_{13} \oplus k_{13}) \oplus (k_{13} \oplus k_9)\right) \oplus \\
&\mathtt{14}\left(S^{-1}(x'_{10} \oplus k_{10}) \oplus (k_{10} \oplus k_{14}))\right) \oplus \mathtt{11}\left(S^{-1}(x'_7 \oplus k_7) \oplus \right. \\
&\left.(k_{15} \oplus k_{11})\right) \oplus \mathtt{13}\left(S^{-1}(x'_4 \oplus k_4) \oplus (k_{16} \oplus k_{12})\right)\Big)
\end{aligned}
$$

$$
\begin{aligned}
f' = S^{-1}\Big(&\mathtt{13}\left(S^{-1}(x_9 \oplus k_9) \oplus (k_9 \oplus k_5)\right) \oplus \\
&\mathtt{9}\left(S^{-1}(x_6 \oplus k_6) \oplus (k_{10} \oplus k_6))\right) \oplus \mathtt{14}\left(S^{-1}(x_3 \oplus k_3)\right. \\
&\left.\oplus (k_{11} \oplus k_7)\right) \oplus \mathtt{11}\left(S^{-1}(x_{16} \oplus k_{16}) \oplus (k_{12} \oplus k_8)\right)\Big) \oplus \\
&S^{-1}\Big(\mathtt{13}\left(S^{-1}(x'_9 \oplus k_9) \oplus (k_9 \oplus k_5)\right) \oplus \\
&\mathtt{9}\left(S^{-1}(x'_6 \oplus k_6) \oplus (k_{10} \oplus k_6))\right) \oplus \mathtt{14}\left(S^{-1}(x'_3 \oplus k_3) \oplus \right. \\
&\left.(k_{11} \oplus k_7)\right) \oplus \mathtt{11}\left(S^{-1}(x'_{16} \oplus k_{16}) \oplus (k_{12} \oplus k_8)\right)\Big)
\end{aligned}
$$

$$
\begin{aligned}
3\,f' = S^{-1}\Big(&\mathtt{11}\left(S^{-1}(x_5 \oplus k_5) \oplus (k_5 \oplus k_1)\right) \oplus \\
&\mathtt{13}\left(S^{-1}(x_2 \oplus k_2) \oplus (k_6 \oplus k_2))\right) \oplus \mathtt{9}\left(S^{-1}(x_{15} \oplus k_{15}) \oplus \right. \\
&\left.(k_7 \oplus k_3)\right) \oplus \mathtt{8}\left(S^{-1}(x_{12} \oplus k_{12}) \oplus (k_8 \oplus k_4)\right)\Big) \oplus \\
&S^{-1}\Big(\mathtt{11}\left(S^{-1}(x'_5 \oplus k_5) \oplus (k_5 \oplus k_1)\right) \oplus \\
&\mathtt{13}\left(S^{-1}(x'_2 \oplus k_2) \oplus (k_6 \oplus k_2))\right) \oplus \mathtt{9}\left(S^{-1}(x'_{15} \oplus k_{15}) \oplus \right. \\
&\left.(k_7 \oplus k_3)\right) \oplus \mathtt{14}\left(S^{-1}(x'_{12} \oplus k_{12}) \oplus (k_8 \oplus k_4)\right)\Big)
\end{aligned}
$$

The second stage of the attack is coupled with the first stage, and can be used to further reduce the number of key hypotheses.

3.4 Analysis of the second step of the fault attack

The expected number of hypotheses produced by the second step of the attack follows a similar reasoning to the analysis of the first step, given in Section 3.2.

If we consider the second equation defined in Section 3.3, it can be rewritten as

$$f' = A \oplus B\,,$$

where A and B are defined as

$$
\begin{aligned}
A = S^{-1}\Big(&\mathtt{9}\left(S^{-1}(x_{13} \oplus k_{13}) \oplus (k_{13} \oplus k_9)\right) \oplus \\
&\mathtt{14}\left(S^{-1}(x_{10} \oplus k_{10}) \oplus (k_{10} \oplus k_{14}))\right) \oplus \mathtt{11}\left(S^{-1}(x_7 \oplus k_7) \oplus \right. \\
&\left.(k_{15} \oplus k_{11})\right) \oplus \mathtt{13}\left(S^{-1}(x_4 \oplus k_4) \oplus (k_{16} \oplus k_{12})\right)\Big)
\end{aligned}
$$

and

$$
\begin{aligned}
B = S^{-1}\Big(&\mathtt{9}\left(S^{-1}(x'_{13} \oplus k_{13}) \oplus (k_{13} \oplus k_9)\right) \oplus \\
&\mathtt{14}\left(S^{-1}(x'_{10} \oplus k_{10}) \oplus (k_{10} \oplus k_{14}))\right) \oplus \mathtt{11}\left(S^{-1}(x'_7 \oplus k_7) \oplus\right. \\
&\left.(k_{15} \oplus k_{11})\right) \oplus \mathtt{13}\left(S^{-1}(x'_4 \oplus k_4) \oplus (k_{16} \oplus k_{12})\right)\Big)
\end{aligned}
\quad .
$$

We can consider A and B to be random values in $\mathbb{F}_{2^8}$. For a given values of f' the difference between A and B will be equal to f' with a probability of $\frac{1}{2^8}$. Using the same reasoning, the probability of all four equations being valid is $\left(\frac{1}{2^8}\right)^4 = \frac{1}{2^{32}}$.

We have to consider all the possible values of f', i.e. $\{0, \ldots, 255\}$. A given key hypothesis will, therefore, be valid for some arbitrary value of f' with a probability of $2^8 \times \frac{1}{2^{32}} = \frac{1}{2^{24}}$. The first step of the attack is expected to return 2^{32} hypotheses each of which still be under consideration at the end of the second step with a probability of $\frac{1}{2^{24}}$. One would, therefore, expect the second step of the attack to produce 2^8 possible key hypotheses.

3.5 Attacking Other Bytes

In the previous sections we describe an attack where we base our Differential Fault Analysis on the knowledge that a fault has been induced in the first byte of the state matrix. However, we can note that the analysis returns a very small number of hypotheses. We can, therefore, conduct 16 independent analyses under the assumption that a fault is induced each of the 16 bytes of of the state at the beginning of the eighth round. An attacker would expect this to produce $2^4 \times 2^8 = 2^{12}$ valid key hypotheses, which is still a trivial exhaustive search.

4 Extending the Fault Attack

In this section we demonstrate that the fault attack described in the previous attack can be extended using the second fault model defined in Section 2.2. The model requires more assumptions but allows the number of hypotheses to be further reduced.

In the second model defined in Section 2.2 we assume that a fault is injected that modifies the opcodes being processed in a microprocessor. Specifically, we consider a fault that reduces the number of bytes processed by the `MixColumn` function in the eighth round from 16 bytes to 15 bytes, i.e. there is one byte that remains unchanged by this function.

The advantage of this type of fault is that an attacker would expect to be able to identify when the desired fault has been induced by observing a suitable side channel. For example, Figure 2 shows the power consumption of an ARM7TDMI microprocessor [22] towards the end of a computation of the `MixColumn` function. The black trace shows the power consumption where the `MixColumn` function treats all 16 bytes. The gray trace shows the power consumption where the `MixColumn` function treats 15 bytes. The traces have an almost identical power consumption on the left side of the figure, and diverge towards the middle of the figure. This difference could be seen by an attacker and the corresponding faulty ciphertext could then be used to derive information on the secret key using the method described in this paper.

An attacker could treat this acquired faulty ciphertext and the correct ciphertext to conduct the attack described in Section 3. After generating an expected 2^8 possible key hypotheses, an attacker could proceed to verify that the effect of the fault is possible with the generated key hypotheses, i.e. the input of the relevant byte to the `MixColumn` function is the same as the output when a fault is induced. This verification can be conducted in parallel with the evaluation of the second set of equations described in Section 3.3.

The expected number of hypotheses that would be returned by this verification would be approximately two. This can be seen that if we consider that of the 2^8 possible hypotheses returned by the analysis described in Section 3, one of these hypotheses will be the correct key and each of the remaining hypotheses will have a probably of satisfying the test with a probability of $\frac{1}{256}$. The expected number of key hypotheses is, therefore, $1\frac{255}{256}$.

As with the attack described in Section 3, the same analysis could be applied if an attacker is not able to determine what byte has been affected. In this case, an attacker would expect to have $16\frac{255}{256}$ key hypotheses (if a set of 2^8 hypotheses does not contain the correct key value, random values would be valid with a probability of $\frac{1}{256}$).

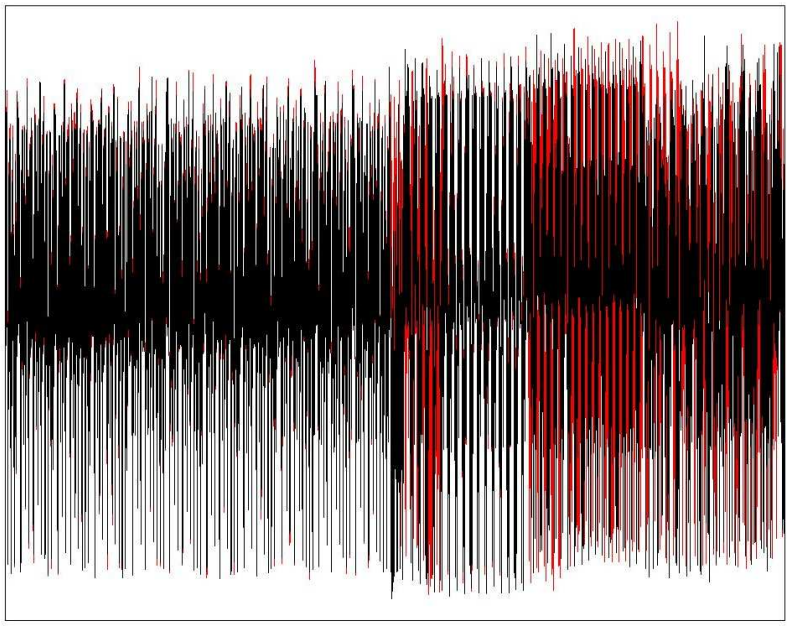

Fig. 2: Overlaid power consumption traces taken during the computation of the end of the `MixColumn` function when treating 16 bytes (black) and 15 bytes (gray).

5 Experimental Results

In this section we present the results of two implementations of the attacks described above. The first targets an FPGA and uses a glitch in the supplied clock to induce a fault. The second targets a microprocessor and uses a glitch in the power supply to induce a fault.

5.1 Experiments on a FPGA

An iterative AES-128 was implemented on a Xilinx Spartan-3E xc3s500E device using Verilog HDL which required an operating clock speed of 36 MHz. The experimental set up consisted of two different clocks, one of which was too high for the design and would therefore induce a fault. Experiments similar to this are described in more detail by Fukunaga and Takahashi [10].

When a 36 MHz clock is used the implementation functions as expected. However, when the system is switched to a clock with a higher frequency faults are generated. The frequency of the faster clock was chosen to ensure that the clock frequency violates the set up time requirement by the design. This frequency was determined using the Xilinx ChipScope 7.1 Pro tool.

In order to control the timing of the fault, as per the requirements of our attack, we switched from the slow to the faster clock at the beginning of the eighth round for one clock cycle. In table 1 we provide observations on the nature of the random one byte faults generated using different clock frequencies. We started the experiment with a high clock frequency set to 72 MHz, which did not create any faults.

The clock frequency was incrementally increased by 0.2 MHz for subsequent experiments and we performed 512 attacks for each iteration. Our experiments showed that a clock frequency of 72.6 MHz is the lowest clock frequency that would inject a one byte fault, and the same behavior was observed with a clock frequencies up to 73.8 MHz. Frequencies higher than this induced faults that affected multiple byte faults. The table further shows that the probability of a fault being induced and the number of bytes affected by an induced fault increases as the clock frequency is increased.

The faulty ciphertexts obtained were analyzed by a software program written in C to derive the key using the two phases of the attack.

Table 1: Fault Induction on AES running on Xilinx Spartan-3E using Clock Glitchings

Clock Frequency (MHz.)	Number with no Fault	Number of multiple Byte Faults	Number of one Byte Faults
72.0	512	0	0
72.2	512	0	0
72.6	510	0	2
72.8	511	0	1
73.0	508	0	4
73.2	504	0	8
73.4	507	0	5
73.6	490	0	22
73.8	489	0	23
74	419	14	79
76	158	163	191
77	0	492	20

5.2 Experiments on a Microprocessor

A second set of experiments was conducted where we injected a fault into an implementation of AES on an ARM7TDMI microprocessor using a glitch on the power supply to the core of the chip. Specifically, we reduced the core power supply of an NXP LPC2124 [22] microprocessor, which is typically set to 1.8, while the clock frequency was set to 29.5 MHz. The voltage was lowered for four clock cycles and then returned to 1.8 volts.

The approximate location of the `MixColumn` function computed in the seventh round was located approximately using Simple Power Analysis, i.e. by observing the pattern created by the round function in the power consumption over time as described in [15]. A window of 501 clock cycles was selected that encompassed this the `MixColumn` function. Ten attempts to inject a fault were conducted at each of these 501 points in time (the leading edge of the glitch was set to this point in time).

This process was initially conducted using a glitch that reduced the voltage supplied to the core of the microprocessor from 1.8 volts to 1.45 volts. The process was then repeated reducing the voltage by increments of 0.05 volts until 1.10 volts. The results of these experiments are summarized in Table 2.

Table 2: Fault Induction on AES running on ARM7TDMI using a glitch in the Vcc.

Glitch Voltage	Number with no Fault	Number of multiple Byte Faults	Number of one Byte Faults
1.45	5009	1	0
1.40	4709	128	173
1.35	3984	448	295
1.30	3450	1356	204
1.25	2989	1288	733
1.20	2370	1803	837
1.15	2369	1813	828
1.10	2311	1926	773

As in the previous section, the faulty ciphertexts obtained were analyzed by a software program written in C to derive the key that was used to generate the ciphertexts.

5.3 Comments on the Experimental Results

In our experiments we have seen a large proportion of faults that only affect one byte, thus demonstrating that the attack is possible and the first fault model described in Section 2.2 is realistic. Erroneous ciphertexts that could correspond to the second fault model described in Section 2.2 were observed. However, these erroneous ciphertexts were not observed in sufficient quantity to be statistically significant. That is, the frequency of these ciphertexts will appear to correspond to the second model with a probability of 1/256. Such faults have been demonstrated to be possible in practice [2], but our observations show that these faults were not in large numbers on the present target processor. The second fault model described in Section 2.2, and the attack described in Section 4, thus applies to specific microprocessors.

6 Comparison with Previous Work

There are several versions of fault-based differential cryptanalysis that are able to reduce the number of key hypotheses from two faults injected into an implementation of AES, as described in [13,19,25]. However, the analysis proposed in this paper is more effective, since the resulting exhaustive search can be reduced to a trivial size using one fault. The number of key hypotheses returned by previous work would be somewhat time consuming. The advantage of the proposed attack is that it does not need to reproduce a successful attack in order to able to determine a secret key. Acquiring multiple faulty ciphertexts can be problematic as faults are only successful with a certain probability, and the effect cannot always be predetermined. This would mean that an attacker could potentially have to search among numerous faulty ciphertexts to find a pair that both have the desired fault.

In our experiments it has taken approximately 50 minutes to generate all the possible key hypotheses, which would mean that an attacker would expect to find a given secret key after 25 minutes if they tested each possible key as it was generated. When compared to the attacks described in [13, 19, 25], it is somewhat time consuming but it reduces the key space from around 2^{32} to a mere 2^8 values. The reduction in key space has two advantages. First, the attack works even if the location of the faulty byte in the state matrix is unknown, as it still keeps the key space trivial in size. The main contribution of the work is that the fault attack is successful with one single fault. Hence the attack proposed in this paper does allow an attacker to minimize the number of attacks that are required to derive a secret key. This is important as each fault injected into a given device may also render that device unusable. This is because each fault will stress a device and there will be some probability that it will produce a permanent, rather than transient, fault.

7 Conclusion

This paper proposes a fault-based differential cryptanalysis of AES, that is an extended version of the attack described in [19]. An attacker would expect to be able to reduce the number of key hypotheses from 2^{128} to 2^8 with one well placed fault. As noted in [18], these attacks can be conducted without any knowledge of the plaintext being enciphered, as an attacker would just need to know the plaintexts were the same. Furthermore we have demonstrated that this attack can be successfully applied to FPGA and microprocessors and the the amount of faults produced that correspond to the required model make the attack very practical.

We also present an extension to our attack based on a fault model presented in [2]. However, we were unable to demonstrate that this attack could be used to attack an AES implementation on either a FPGA or a microprocessor. This attack is, therefore, only likely to be possible in very specific circumstances. That is, a combination of a specific fault an microprocessor, since a mechanism for injecting a fault will have a particular effect on a microprocessor. It is typically not possible to predict this effect without extensive experimentation, or reverse engineering, and it the faults produced will not correspond to all the models used in the literature.

There are many descriptions of a fault-based differential cryptanalysis of AES that could be prevented by repeating the last two or three rounds of an implementation of AES, to verify that no exploitable fault has been inserted [5, 9, 11, 25, 30]. However, to prevent the attack described in this paper the last four rounds would need to be repeated to check no fault was injected. Moreover, given how much information can be gleaned from one fault, one would expect there are attacks that require more faulty ciphertexts that would be able to make use of faults in earlier rounds. One would, therefore, suggest that in order to protect an implementation of AES the last five rounds should be protected against fault injection.

Acknowledgements

The work described in this paper has been supported in part by the European Commission IST Programme under Contract ICT-2007-216676 ECRYPT II and EPSRC grant EP/F039638/1 "Investigation of Power Analysis Attacks". The second author would like to acknowledge the support of Department of Science and Technology (DST) India under the Fast Track Proposals for Young Scientists for the proposal entitled "Design and Analysis of Side Channel Attack Resistant Symmetric Key Cryptosystems".'

References

1. M. Alam, S. Ray, D. Mukhopadhyay, S. Ghosh, D. Roy C., and I. Sengupta. An area optimized reconfigurable encryptor for AES-Rijndael. In R. Lauwereins and J. Madsen, editors, *Design, Automation and Test in Europe Conference and Exposition — DATE 2007*, pages 1116–1121. ACM, 2007.
2. F. Amiel, C. Clavier, and M. Tunstall. Collision fault analysis of DPA-resistant algorithms. In L. Breveglieri, I. Koren, D. Naccache, and J.-P. Seifert, editors, *Fault Diagnosis and Tolerance in Cryptography 2006 — FDTC 06*, volume 4236 of *Lecture Notes in Computer Science*, pages 223–236. Springer-Verlag, 2006.
3. H. Bar-El, H. Choukri, D. Naccache, M. Tunstall, and C. Whelan. The sorcerer's apprentice guide to fault attacks. *Proceedings of the IEEE*, 94(2):370–382, 2006.
4. E. Biham and A. Shamir. Differential fault analysis of secret key cryptosystems. In B. S. Kaliski, editor, *Advances in Cryptology — CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 513–525. Springer-Verlag, 1997.
5. J. Blömer and J.-P. Seifert. Fault based cryptanalysis of the advanced encryption standard (AES). In R. N. Wright, editor, *Financial Cryptography — FC 2003*, volume 2742 of *Lecture Notes in Computer Science*, pages 162–181. Springer-Verlag, 2003.
6. D. Boneh, R. DeMillo, and R. Lipton. On the importance of checking cryptographic protocols for faults. In W. Fumy, editor, *Advances in Cryptology — EUROCRYPT '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 37–51. Springer-Verlag, 1997.
7. D. Boneh, R. DeMillo, and R. Lipton. On the importance of checking cryptographic protocols for faults. *Journal of Cryptology*, 14(2):101–119, 2001.
8. H. Choukri and M. Tunstall. Round reduction using faults. In L. Breveglieri and I. Koren, editors, *Workshop on Fault Diagnosis and Tolerance in Cryptography 2005 — FDTC 05*, pages 13–24, 2005.
9. P. Dusart, G. Letourneux, and O. Vivolo. Differential fault analysis on A.E.S. In J. Zhou, M. Yung, and Y. Han, editors, *Applied Cryptography and Network Security — ACNS 2003*, volume 2846 of *Lecture Notes in Computer Science*, pages 293–306. Springer-Verlag, 2003.
10. T. Fukunaga and J. Takahashi. Practical fault attack on a cryptographic LSI with ISO/IEC 18033-3 block ciphers. In D. Naccache and E. Oswald, editors, *Fault Diagnosis and Tolerance in Cryptography — 2009*, pages 84–92, 2009.
11. C. Giraud. DFA on AES. In H. Dobbertin, V. Rijmen, and A. Sowa, editors, *International Conference Advanced Encryption Standard — AES 2004*, volume 3373 of *Lecture Notes in Computer Science*, pages 27–41. Springer-Verlag, 2004.
12. C. Giraud and A. Thillard. Piret and Quisquater's DFA on AES revisited. Cryptology ePrint Archive, Report 2010/440, 2010. `http://eprint.iacr.org/`.

13. C. H. Kim and J.-J. Quisquater. New differential fault analysis on aes key schedule: Two faults are enough. In G. Grimaud and F.-X. Standaert, editors, *Smart Card Research and Advanced Applications — CARDIS 2008*, volume 5189 of *Lecture Notes in Computer Science*, pages 48–60. Springer-Verlag, 2008.
14. L. Knudsen. Deal — a 128-bit block cipher. Technical report no. 151. Department of Informatics, University of Bergen, Norway, 1998.
15. P. C. Kocher, J. Jaffe, and B./Jun. Differential power analysis. In M. J. Wiener, editor, *Advances in Cryptology — CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer-Verlag, 1999.
16. Yang Li, Shigeto Gomisawa, Kazuo Sakiyama, and Kazuo Ohta. An information theoretic perspective on the differential fault analysis against aes. Cryptology ePrint Archive, Report 2010/032, 2010. `http://eprint.iacr.org/`.
17. A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.
18. A. Moradi, M. T. Manzuri Shalmani, and M. Salmasizadeh. A generalized method of differential fault attack against AES cryptosystem. In L. Goubin and M. Matsui, editors, *Cryptographic Hardware and Embedded Systems — CHES 2006*, volume 4249 of *Lecture Notes in Computer Science*, pages 91–100. Springer-Verlag, 2006.
19. D. Mukhopadhyay. An improved fault based attack of the advanced encryption standard. In B. Preneel, editor, *Progress in Cryptology — AFRICACRYPT 2009*, volume 5580 of *Lecture Notes in Computer Science*, pages 421–434. Springer-Verlag, 2009.
20. National Institute of Standards and Technology (NIST). Data encryption standard (DES). FIPS Publication 46-3, available for download at `http://www.itl.nist.gov/fipspubs/`, 1999.
21. National Institute of Standards and Technology (NIST). Advanced Encryption Standard (AES). FIPS Publication 197, available for download at `http://www.itl.nist.gov/fipspubs/`, 2001.
22. NXP B.V. LPC2114/2124 single-chip 16/32-bit microcontrollers. `http://www.nxp.com/documents/data\_sheet/LPC2114\_2124.pdf`, 2007.
23. K. Nyberg. Differentially uniform mappings for cryptography. In T. Helleseth, editor, *Advances in Cryptology — EUROCRYPT '93*, volume 765 of *Lecture Notes in Computer Science*, pages 55–64. Springer-Verlag, 1993.
24. S. B. Ors, E. Oswald, and B. Preneel. Power-analysis attacks on an FPGA — first experimental results. In C. D. Walter, Ç. K. Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems — CHES 2003*, volume 2779 of *Lecture Notes in Computer Science*, pages 35–50. Springer-Verlag, 2003.
25. G. Piret and J.-J. Quisquater. A differential fault attack technique against SPN structure, with application to the AES and KHAZAD. In C. D. Walter, Ç. K. Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems — CHES 2003*, volume 2779 of *Lecture Notes in Computer Science*, pages 77–88. Springer-Verlag, 2003.
26. R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public key cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.
27. D. Samyde, S. P. Skorobogatov, R. J. Anderson, and J.-J. Quisquater. On a new way to read data from memory. In *Proceedings of the First International IEEE Security in Storage Workshop*, pages 65–69, 2002.
28. S. Skorobogatov and R. Anderson. Optical fault induction attacks. In B. S. Kaliski, Ç. K. Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems — CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 2–12. Springer-Verlag, 2002.
29. J. Takahashi and T. Fukunaga. Differential fault analysis on the AES key schedule. Cryptology ePrint Archive, Report 2007/480, 2007. `http://eprint.iacr.org/`.
30. J. Takahashi, T. Fukunaga, and K. Yamakoshi. DFA mechanism on the AES schedule. In *Fault Diagnosis and Tolerance in Cryptography 2007 — FDTC 07*, pages 62–72, 2007.