# Differential Fault Analysis of AES-128 Key Schedule Using a Single Multi-byte Fault

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 an improved multi-byte differential fault analysis of AES-128 key schedule using a single pair of fault-free and faulty ciphertexts. We propose a four byte fault model where the fault is induced at ninth round key. The induced fault corrupts all the four bytes of the first column of the ninth round key which subsequently propagates to the entire tenth round key. The elegance of the proposed attack is that it requires only a single faulty ciphertext and reduce the search space of the key to $2^{32}$ possible choices. Using two faulty ciphertexts the attack uniquely determines the key. The attack improves the existing DFA of AES-128 key schedule, which requires two faulty ciphertexts to reduce the key space of AES-128 to $2^{32}$, and four faulty ciphertexts to uniquely retrieve the key. Therefore, the proposed attack is more lethal than the existing attack as it requires lesser number of faulty ciphertexts. The simulated attack takes less than 20 minutes to reveal 128-bit secret key; running on a 8 core Intel Xeon E5606 processor at 2.13 GHz speed.

**Keywords:** Differential Fault Analysis, Fault Attack, Advanced Encryption Standard, Key Schedule, DFA.

## 1 Introduction

External noise such electromagnetic radiation, glitch in the input clock line, variation in the supply voltage can create fault in the electronic devices such as smart card. These properties of electronic devices are being exploited by the attackers. An attacker can deliberately inject fault into a device running a cryptographic algorithm. Then by analysing the faulty output he can reveal the secret key. This kind of attack is known as fault attack which was originally introduced by Boneh et. al. [7] in 1996 to break RSA crypto-system running on a smart-card . Subsequently, Biham and Shamir showed a modified form of the attack which is known as Differential Fault Analysis (DFA) based on combination of Differential Cryptanalysis and fault analysis [5]. The DFA was applied on DES crypto-system which successfully retrieved the secret key.

In 2001, NIST standardised Rijndael as the Advanced Encryption Standard (AES) [1]. Subsequently, many DFA were proposed on AES cryptosystem [6, 11, 14, 15, 17] with the aim to reduced the number of faulty ciphertext required by the attack. However the DFA on AES can be divided into two categories. One in

E. Prouff (Ed.): CARDIS 2011, LNCS 7079, pp. 50–64, 2011.

which the fault is induced in AES states, another in which the fault is induced in the key schedule. The literature is rich in attacks on the states. The most recent among these attacks is the attack on AES-128 proposed by M. Tunstall et. al. in [21, 22]. They proposed an attack where a single byte fault induced at the input of eighth round can reduce the AES-128 key space to $2^8$ choices. However, there is not much research on attacks of AES key schedule.

In 2003, Giraud first proposed a DFA against the AES key schedule [10], which combined both kind of fault attack; the fault analysis in AES states as well as in key schedule. The attack was subsequently improved by Chen and Yen in [9]. Chen et. al. attack required to induce fault at the ninth round key. The attack required less than thirty faulty ciphertexts to successfully retrieve the secret key. In 2006 Peacham and Thomas in [16], proposed an improved DFA against the AES key schedule, which reduced the number of faulty ciphertext required by the attack to 12. In Peacham's attack, fault was induced at the ninth round key during execution of the key schedule operation, so that the induced fault subsequently propagates to tenth round key.

Finally, Takahashi et. al. in [20], proposed an attack, which can reduce the search space of the key to $2^{48}$ choices using two faulty ciphertexts. The attack can reduce the key space to $2^{16}$ choices using four faulty ciphertexts and can determine the key uniquely by using seven faulty ciphertexts. In 2008, Kim et. al. proposed an improved DFA by inducing 3-byte fault at the first column of ninth round key. Kim's attack required two faulty ciphertexts and a brute-force search of $2^{32}$. With four faulty ciphertexts the attack can uniquely determine the secret key.

In this paper we propose an improved attack against the AES-128 key schedule. We propose a fault model where the induced fault corrupts all the four bytes of the first column of ninth round key and the fault subsequently propagates to the entire tenth round key. Our attack requires only one faulty ciphertext to reduces the search space of the AES-128 key to $2^{32}$ choices. Using two faulty ciphertext our attack can uniquely determine the key. We present extensive simulation results which shows that the attack takes less than 20 minutes to uniquely retrieve the 128-bit secret key.

**Organization**

The paper is organized as follows: We start with Section 2 where we describe the preliminaries to this paper. In Section 3 we briefly describe the existing attack. We propose our attack in Section 4. In Section 5 we present experimental results. In Section 6 we compare our work presented in this paper with the previous works, and we conclude in Section 7.

## 2 Preliminaries

### 2.1 The AES Algorithm

AES [1] is a 128-bit symmetric key block cipher comes in three different versions AES-128, AES-192, and AES-256 with key length 128-bit, 192-bit, and 256-bit

respectively. The 128-bit intermediate results are represented as $4 \times 4$ matrix, known as state. Each elements of the matrix is a byte. The algorithm is divided into round function. Each round function except the last round consists of four transformations namely: `SubBytes`, `ShiftRows`, `MixColumns` and `AddRoundKey`. The three versions of AES has three different number of rounds : AES-128 has 10 rounds, AES-192 has 12 rounds, and AES-256 has 14 rounds. The last round of each of the three versions of AES does not have `MixColumns` operation. The four basic transformations are described as follows:

`SubBytes` : It is the only non-linear byte-wise transformation in AES. Each element of the state matrix is replaced by its inverse and followed by an affine mapping. All the operations are under $\mathbf{F_{2^8}}$.

`ShiftRows` : It is a row-wise transformations where the $i^{th}$ row is cyclically shifted by $i$ bytes towards left where $0 \leq i \leq 3$ .

`MixColumns` : It is a column level linear transformation of the state matrix. Each column of the state matrix is considered as a polynomial of degree 3 with coefficient in $\mathbf{F_{2^8}}$ and multiplied with the polynomial $\{03\}x^3+\{01\}x^2+\{01\}x\ +\{02\}$.

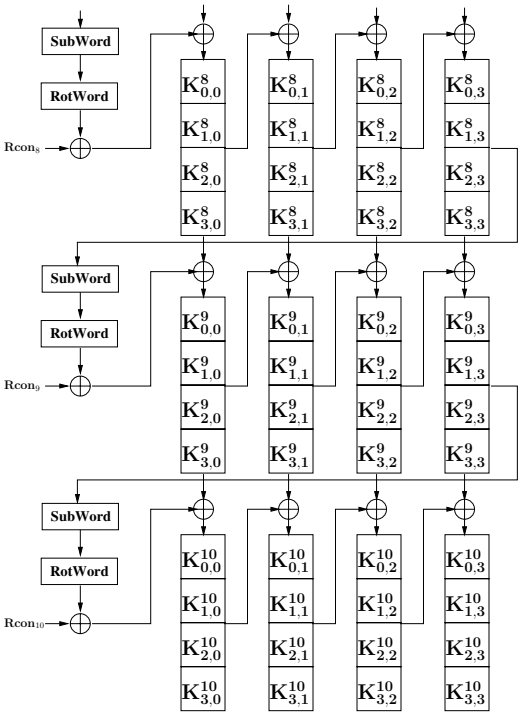


**Fig. 1.** Last three rounds of AES-128 key scheduling algorithm

`AddRoundKey`: In this transformation the 128-bit round key is bit-wise XOR-ed with the 128-bit state.

An additional `AddRoundKey` operation is performed at the beginning of first round which is known as key whitening phase. Each round key is generated by the AES key scheduling algorithm. Figure 1 shows the generation of last three rounds according to the AES-128 key schedule. For more detail on the AES key scheduling, one can refer the AES specification [1]

### 2.2 Notations

In the rest of the paper we use following symbols and notations: `SubBytes`, `ShiftRows`,and `MixColumns` will be referred as $SB, SR$ and $MC$ respectively and the corresponding inverse functions as $SB^{-1}, SR^{-1}$ and $MC^{-1}$.

$K^r_{i,j}$: Represent $\{i, j\}$ byte of the $r^{th}$ round key $K^r$.
$C_{i,j}$ : Represent $\{i, j\}$ byte of the fault-free ciphertext $C$.
$C^*_{i,j}$: Represent $\{i, j\}$ byte of the faulty ciphertext $C^*$.

### 2.3 Fault Model Used

It is clear from the past research on DFA of AES that the fault model is the key to successful fault analysis. Slightest change in the fault model can drastically increase or decrease the complexity of the fault analysis. This fact was clearly

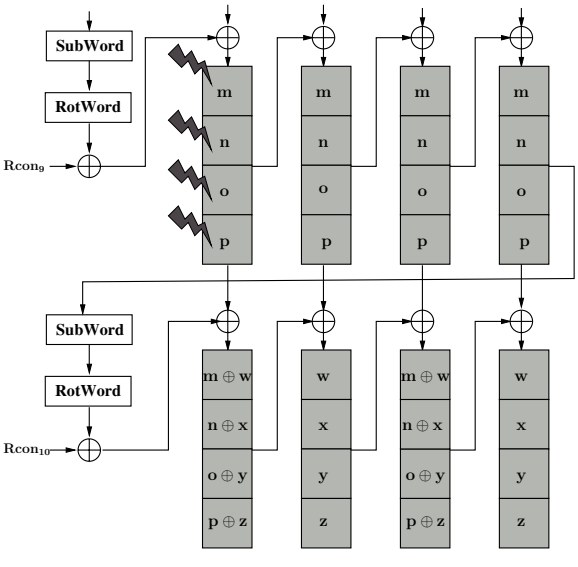


**Fig. 2.** Fault induced in $9^{th}$ round key corrupting entire first column

depicted in [3] where it was shown that with the increase of number of byte faults the search space of the key drastically increases. In our proposed attack we follow the fault model of Peachman and Thomas [16] where the fault is expected to induce in the ninth round key while it is being executed and subsequently propagated to the entire tenth round key. We assume that the induced fault corrupts all the four bytes of the first column of the ninth round key. Figure 2 show the flow of fault in the last two round keys.

## 3 Existing Fault Analysis

The most recent attack against the AES-128 key schedule shows that two faulty ciphertexts are enough to reduce the key space to $2^{32}$ choices [13]. The attack assumed a fault model where the induced fault corrupts three out of four bytes of the first column of the ninth round key $K^9$. As Figure 3 depicts, the induced fault subsequently propagates to the tenth round key. The three key bytes : $K^9_{0,0}, K^9_{1,0}$, and $K^9_{2,0}$ are the modified bytes and induced differences due to the faults are $a, b$, and $c$ respectively. As per the AES-128 key scheduling algorithm the fault is propagated to all the subsequent three bytes of the same row of the ninth round key. As the fault is induced during the generation of key therefore the fault in the ninth round key also propagated to tenth round key bytes.

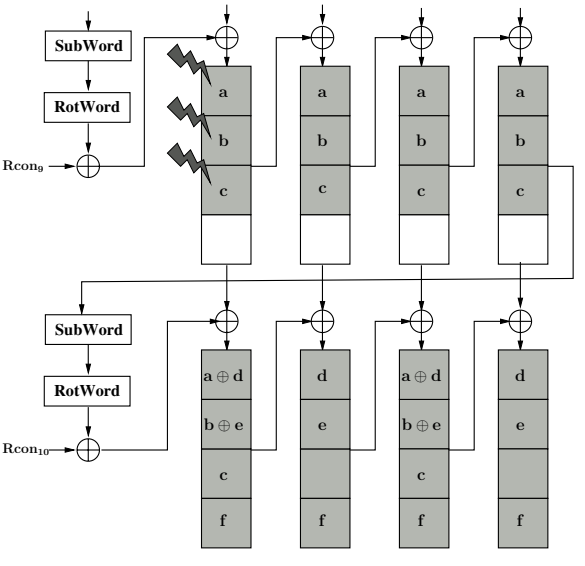


**Fig. 3.** Fault induced in $9^{th}$ round key

The fault values $d, e$, and $f$ can be calculated as follows:

$$\begin{aligned} d &= SB(K^9_{1,3}) \oplus SB(K^9_{1,3} \oplus b) \\ &= SB(K^{10}_{1,3} \oplus K^{10}_{1,2}) \oplus SB(K^{10}_{1,3} \oplus K^{10}_{1,2} \oplus b) \end{aligned} \tag{1}$$

$$\begin{aligned} e &= SB(K^9_{2,3}) \oplus SB(K^9_{2,3} \oplus c) \\ &= SB(K^{10}_{2,3} \oplus K^{10}_{2,2}) \oplus SB(K^{10}_{2,3} \oplus K^{10}_{2,2} \oplus c) \end{aligned} \tag{2}$$

$$\begin{aligned} f &= SB(K^9_{0,3}) \oplus SB(K^9_{0,3} \oplus a) \\ &= SB(K^{10}_{0,3} \oplus K^{10}_{0,2}) \oplus SB(K^{10}_{0,3} \oplus K^{10}_{0,2} \oplus a) \end{aligned} \tag{3}$$

The fault in the ninth round key affects the AES state after the ninth round *AddRoundKey*. Figure 4 shows the flow of fault in the last two rounds.

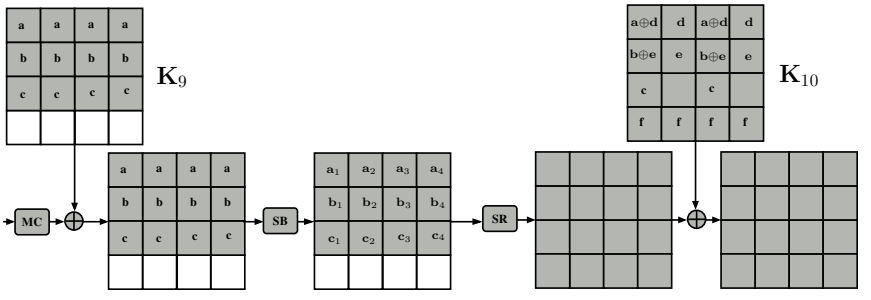


**Fig. 4.** Propagation of Faults in the AES-datapath due to the 3-byte Fault Induction in the first column of $K^9$

The fault-free and faulty ciphertexts $(C, C^*)$ are known to the attacker. Therefore, he can represent the fault values $\{c, c, c, c\}$, at the input of ninth round using following equations:

$$\begin{aligned} c &= SB^{-1}(C_{2,2} \oplus K^{10}_{2,2}) \oplus SB^{-1}(C^*_{(2,2)} \oplus K^{10}_{2,2} \oplus c) \\ &= SB^{-1}(C_{2,3} \oplus K^{10}_{2,3}) \oplus SB^{-1}(C^*_{(2,3)} \oplus K^{10}_{2,3}) \\ &= SB^{-1}(C_{2,0} \oplus K^{10}_{2,0}) \oplus SB^{-1}(C^*_{(2,0)} \oplus K^{10}_{2,0} \oplus c) \\ &= SB^{-1}(C_{2,1} \oplus K^{10}_{2,1}) \oplus SB^{-1}(C^*_{(2,1)} \oplus K^{10}_{2,1}) \end{aligned} \tag{4}$$

In the above set of equations the attacker guesses the values of $c$ and gets the corresponding values of key quartet $\langle K^{10}_{2,2}, K^{10}_{2,3}, K^{10}_{2,0}, K^{10}_{2,1} \rangle$. As there are $2^8$ possibilities of $c$ therefore the above set of equations will reduce the possible choices of $\langle K^{10}_{2,2}, K^{10}_{2,3}, K^{10}_{2,0}, K^{10}_{2,1} \rangle$ to $2^8$ choices from $2^{32}$ choices. For more details on the analysis, one can refer to [13, 21]. Similarly, the attacker uses another faulty ciphertext and again reduces the possible choices of $\langle K^{10}_{2,2}, K^{10}_{2,3}, K^{10}_{2,0}, K^{10}_{2,1} \rangle$ to $2^8$. Then he takes the intersection of two sets of values of $\langle K^{10}_{2,2}, K^{10}_{2,3}, K^{10}_{2,0}, K^{10}_{2,1} \rangle$ generated from two different faulty ciphertexts. The intersection uniquely determine the key quartet. Now the attacker uses equation (2) and determines the values of $e$ from the values of $c$, $K^{10}_{2,2}$ and $K^{10}_{2,3}$. He follows the same technique to uniquely

determine the quartets $\langle K^{10}_{0,0}, K^{10}_{0,1}, K^{10}_{0,2}, K^{10}_{0,3}\rangle$ and $\langle K^{10}_{1,0}, K^{10}_{1,1}, K^{10}_{1,2}, K^{10}_{1,3}\rangle$. Therefore, using two faulty ciphertexts the attacker can retrieve 12 bytes of the tenth round key $K^{10}$. This implies that the attacker retrieve 96-bit and need to perform 32-bit brute-force search to get the master key.

### 3.1 Limitation of Existing Attack

The existing attack nicely reveals the secret key of AES-128 crypto-system using only two faulty ciphertexts. However, the attack still needs two faulty ciphertexts which is the bottle neck of the attack. First of all the induced faults need to corrupt three out of the four bytes of the first row of the ninth round key. However, in real life the probability of such an event is small. Fault induction methods such as described in [3,4,12,18,19] gives an attacker an extent of control, but cannot precisely realize the number of faults. The probability of success reduces further if more than one fault inductions are necessary. If probability of getting such fault is $p$ then the probability of getting two such fault is $p^2$. It its quite obvious from the experimental result reported in [3, 4, 12, 18, 19] that the value of $p$ is quite small. Therefore, ideally an attacker would want an attack which require one faulty ciphertext.

In the next section we propose an improved attack which will produce the same results as in the existing attack using only one faulty ciphertext.

## 4 Proposed Attack Using Single Faulty Ciphertext

In our proposed attack, the attacker is expected to induce a fault which corrupts all four bytes of the first column of the ninth round key as depicted in Figure 2. The proposed fault model increased the fault coverage where eventually all the bytes of the ninth round key and subsequently all the bytes of the tenth round key are corrupted by the induced fault. In the previous attack only 12 out of 16 bytes of ninth round key were corrupted by the induced fault. As in Figure 3 only the first three rows of the ninth round input state matrix is corrupted by the faulty ninth round key $K^9$. Therefore, only the first three rows of $K^{10}$ participate in forming differential equations. The last row of $K^{10}$ is not related to any fault values. Hence there is no trace of getting the last row's values. But in our fault model there is no such limitations.

### 4.1 Attack Principle

The proposed attack exploits the relation between the faulty byte at the input of the ninth round. Figure 5 depicts the flow of faults. At the input of ninth round, all the bytes of the state matrix $S_0$ are corrupted. However, each of the rows have same fault values, which helps in deducing the differential equations. We have the fault-free and faulty ciphertexts $(C, C^*)$.

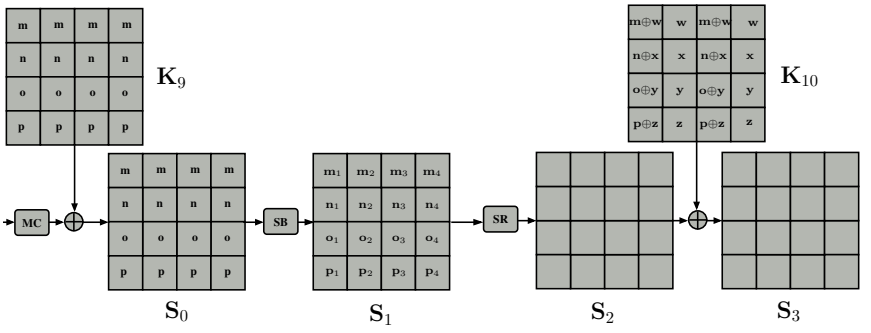


**Fig. 5.** Flow 4 Bytes Fault Induced At First Column of $K^9$

Therefore, we can represent the fault values $\{m, m, m, m\}$ at first the row of the state matrix $S_0$ as follows:

$$\begin{aligned} m &= SB^{-1}(C_{0,0} \oplus K^{10}_{0,0}) \oplus SB^{-1}(C^*_{(0,0)} \oplus K^{10}_{0,0} \oplus m \oplus w) \\ &= SB^{-1}(C_{0,1} \oplus K^{10}_{0,1}) \oplus SB^{-1}(C^*_{(0,1)} \oplus K^{10}_{0,1} \oplus w) \\ &= SB^{-1}(C_{0,2} \oplus K^{10}_{0,2}) \oplus SB^{-1}(C^*_{(0,2)} \oplus K^{10}_{0,2} \oplus m \oplus w) \\ &= SB^{-1}(C_{0,3} \oplus K^{10}_{0,3}) \oplus SB^{-1}(C^*_{(0,3)} \oplus K^{10}_{0,3} \oplus w) \end{aligned} \tag{5}$$

Similarly, the fault values in the rest of the three rows of $S_0$ can be represented by the following equations:

$$\begin{aligned} n &= SB^{-1}(C_{1,3} \oplus K^{10}_{1,3}) \oplus SB^{-1}(C^*_{(1,3)} \oplus K^{10}_{1,3} \oplus x) \\ &= SB^{-1}(C_{1,0} \oplus K^{10}_{1,0}) \oplus SB^{-1}(C^*_{(1,0)} \oplus K^{10}_{1,0} \oplus x \oplus n) \\ &= SB^{-1}(C_{1,1} \oplus K^{10}_{1,1}) \oplus SB^{-1}(C^*_{(1,1)} \oplus K^{10}_{1,1} \oplus x) \\ &= SB^{-1}(C_{1,2} \oplus K^{10}_{1,2}) \oplus SB^{-1}(C^*_{(1,2)} \oplus K^{10}_{1,2} \oplus x \oplus n) \end{aligned} \tag{6}$$

$$\begin{aligned} o &= SB^{-1}(C_{2,2} \oplus K^{10}_{2,2}) \oplus SB^{-1}(C^*_{(2,2)} \oplus K^{10}_{2,2} \oplus y \oplus o) \\ &= SB^{-1}(C_{2,3} \oplus K^{10}_{2,3}) \oplus SB^{-1}(C^*_{(2,3)} \oplus K^{10}_{2,3} \oplus y) \\ &= SB^{-1}(C_{2,0} \oplus K^{10}_{2,0}) \oplus SB^{-1}(C^*_{(2,0)} \oplus K^{10}_{2,0} \oplus y \oplus o) \\ &= SB^{-1}(C_{2,1} \oplus K^{10}_{2,1}) \oplus SB^{-1}(C^*_{(2,1)} \oplus K^{10}_{2,1} \oplus y) \end{aligned} \tag{7}$$

$$\begin{aligned} p &= SB^{-1}(C_{3,1} \oplus K^{10}_{3,1}) \oplus SB^{-1}(C^*_{(3,1)} \oplus K^{10}_{3,1} \oplus z) \\ &= SB^{-1}(C_{3,2} \oplus K^{10}_{3,2}) \oplus SB^{-1}(C^*_{(3,2)} \oplus K^{10}_{3,2} \oplus z \oplus p) \\ &= SB^{-1}(C_{3,3} \oplus K^{10}_{3,3}) \oplus SB^{-1}(C^*_{(3,3)} \oplus K^{10}_{3,3} \oplus z) \\ &= SB^{-1}(C_{3,0} \oplus K^{10}_{3,0}) \oplus SB^{-1}(C^*_{(3,0)} \oplus K^{10}_{3,0} \oplus z \oplus p) \end{aligned} \tag{8}$$

As per the AES-128 key scheduling algorithm the fault value $w$ in the tenth round key $K^{10}$ can be express in terms of $n$ by the following equations:

$$\begin{aligned} w &= S(K^9_{1,3}) \oplus S(K^9_{1,3} \oplus n) \\ &= S(K^{10}_{1,3} \oplus K^{10}_{1,2}) \oplus S(K^{10}_{1,3} \oplus K^{10}_{1,2} \oplus n) \end{aligned} \tag{9}$$

Similarly, we can represent $x, y, z$ by $n, o, p$ respectively and $K^{10}$ using the following equations:

$$x = S(K^{10}_{2,3} \oplus K^{10}_{2,2}) \oplus S(K^{10}_{2,3} \oplus K^{10}_{2,2} \oplus o) \tag{10}$$

$$y = S(K^{10}_{3,3} \oplus K^{10}_{3,2}) \oplus S(K^{10}_{3,3} \oplus K^{10}_{3,2} \oplus p) \tag{11}$$

$$z = S(K^{10}_{0,3} \oplus K^{10}_{0,2}) \oplus S(K^{10}_{0,3} \oplus K^{10}_{0,2} \oplus m) \tag{12}$$

Now we have four sets of differential equations for the four quartets of key bytes like the attack in [21]. However, we can not directly apply the solving technique proposed in [21]. The reason is each set of equations contain six unknown variables. For, example the unknown variables in the first set of equations (5) are $\{K^{10}_{0,0}, K^{10}_{0,1}, K^{10}_{0,2}, K^{10}_{0,3}, m, w\}$. Therefore, in this case we have to guess all possible values of $m$ and $w$. For one choice of $(m, w)$ on an average we get one choice of $\langle K^{10}_{0,0}, K^{10}_{0,1}, K^{10}_{0,2}, K^{10}_{0,3}\rangle$. Therefore, for all possible $2^{16}$ choices of $(m, w)$ we get $2^{16}$ choices of the key quartet. If we apply the same technique to the other three sets of equations (6), (7), (8) we get $2^{16}$ choices for each of the three quartets. Therefore, finally we get $(2^{16})^4 = 2^{64}$ choices of the key $K^{10}$ which is not in practical limits. This means the previous solving technique is not directly applicable.

We follow divide and conquer technique to solve the above four sets of differential equations [2]. We use S-box difference table so that we can directly get the values of the key byte from the given S-box input difference and the output difference. We start with the first set of differential equations (5). In these sets of equation we guess the possible values of $(m, w)$. For one choice of $(m, w)$ we directly get on an average one choice of $\langle K^{10}_{0,0}, K^{10}_{0,1}, K^{10}_{0,2}, K^{10}_{0,3}\rangle$ using S-box difference table. Therefore, for $2^{16}$ possible choices of $(m, w)$ we get $2^{16}$ choices of the key quartet with time complexity $2^{16}$. For, each value of $m$ and the corresponding value of $K^{10}_{0,2}, K^{10}_{0,3}$ we get the values of $z$ using equation (12). Next we consider the fourth set of equations (8) where the values of $z$ are already known from the previous steps. Therefore, in (8) we only guess the values of $p$ and get the values corresponding to the quartet $\langle K^{10}_{3,0}, K^{10}_{3,1}, K^{10}_{3,2}, K^{10}_{3,3}\rangle$. For each values of $(m, w, z)$ and $p$ we get one value of $\langle K^{10}_{3,0}, K^{10}_{3,1}, K^{10}_{3,2}, K^{10}_{3,3}\rangle$. There are $2^{16}$ possible choices of $(m, w, z)$ and $2^8$ choices of $p$. Therefore, together we have $2^{24}$ choices of first and fourth quartets of key bytes from the first and fourth set of equations and the time complexity of the process is $2^{24}$.

Now we use the values of $K^{10}_{3,3}, K^{10}_{3,2}$ and $p$ to get the corresponding values of $y$ using equations (11). Similarly, we choose the third set of equations (7) and get the $2^8$ values $\langle K^{10}_{2,0}, K^{10}_{2,1}, K^{10}_{2,2}, K^{10}_{2,3}\rangle$. Therefore, from the three sets of equations (5), (7), and (8) we get $2^{32}$ choices of the first, third and fourth quartet of key bytes. Following the same technique we get the values of $x$ from

equation (10) and then get the $2^8$ values of the second quartet of key bytes $\langle K^{10}_{1,0}, K^{10}_{1,1}, K^{10}_{1,2}, K^{10}_{1,3}\rangle$ using the second set of equations (6). Therefore, now we have $2^{40}$ choices of all the four quartets of key bytes which is the tenth round key $K^{10}$.

But still we have one equation left unchecked i.e. equation (9). Each of the tenth round key and the corresponding values of $w$ is tested by equation (9). Those which satisfy are considered and the rest are discarded. Finally, $2^{32}$ out of $2^{40}$ candidates will satisfy equation (9). Therefore, finally we have $2^{32}$ possible choices of the tenth round key $K^{10}$. So, we need to do 32-bit brute-force search to get the master key.

However, the attack time complexity is $2^{40}$ which is in practical limits. But the entire attack takes around 14 hours to generate all the possible $2^{32}$ keys which is not feasible in terms of side-channel cryptanalysis.

In the next section we show a technique which further reduces the time complexity of the attack.

### 4.2 Time Complexity Reduction Technique

The proposed attack has got four steps. In the first step we deduce the possible choices $K^{10}_{0,i}$ where $0 \leq i \leq 3$ and $z$. Similarly, in second, third and fourth steps we get the possible choices of $K^{10}_{1,i}$, $K^{10}_{2,i}$,and $K^{10}_{3,i}$ respectively and the corresponding values of $y, x$, $w$.

In the first step, for a given value of $w$ we get $2^8$ choices of the quartet $K^{10}_{0,i}$ from which we calculate $z$. However for getting $z$ from equation (12) we need only two key bytes $(K^{10}_{0,3}, K^{10}_{0,2})$ of quartet $K^{10}_{0,i}$. Therefore, we only consider the unique values of $(K^{10}_{0,3}, K^{10}_{0,2})$ out of $2^8$ values of $K^{10}_{0,i}$. The number of unique values of $(K^{10}_{0,3}, K^{10}_{0,2})$ is given by $\frac{2^8}{2^2} = 2^6$ [2]. Similarly, in the rest of the three steps we consider $2^6$ choices of $(K^{10}_{3,3}, K^{10}_{3,2})$, $(K^{10}_{2,3}, K^{10}_{2,2})$ and $(K^{10}_{1,3}, K^{10}_{1,2})$ each for getting the values of $y, x$, and $w$ respectively. This implies, for testing equation (9) indirectly we need only eight key bytes. For a given value of $w$ the possible choices of these eight key bytes is $(2^6)^4 = 2^{24}$. For all possible $2^8$ choices of $w$ we have $2^{24} \times 2^8 = 2^{32}$ choice of the required eight bytes. Each choice of these eight key bytes are tested by equation (9) if they satisfy they are accepted, else they are discarded. Thus only $2^{24}$ out of $2^{32}$ choices satisfy the test. Those which satisfy are combined with rest of the eight key bytes $(K^{10}_{0,0}, K^{10}_{0,1})$, $(K^{10}_{1,0}, K^{10}_{1,1})$, $(K^{10}_{2,0}, K^{10}_{2,1})$, $(K^{10}_{3,0}, K^{10}_{3,1})$ . There are $2^2 \times 2^2 \times 2^2 \times 2^2 = 2^8$ choices for rest of the eight key bytes. Therefore, using this technique the equation (9) is tested for $2^{32}$ times which correspond to the time complexity of the attack. This implies, that using this technique the time complexity of the attack reduced to $2^{32}$ from $2^{40}$ and hence comes in practical limits.

The simulated, attack written in C programing language was run on a 8 core Intel Xeon E5606 processor at 2.13 GHz speed. The attack takes less than 20 minutes to deduce the master key. Algorithm 1 summarizes the attack procedure.

**Algorithm 1.** DFA on AES-128 Key Scheduling using Single Faulty Ciphertexts

**Input**: $P, C, C^*$
**Output**: 128-bit AES key
/*P is the plaintext */
**for** *Each candidates of* $w$ **do**
  **for** *Each candidates of* $m$ **do**
    Get $\{K^{10}_{0,2}, K^{10}_{0,3}\}$ from equations (5)
    Get $z$ from equation (12).
    **for** *Each candidates of* $p$ **do**
      Get $\{K^{10}_{3,2}, K^{10}_{3,3}\}$ from the equations (8).
      Get $y$ from equation (11).
      **for** *Each candidates of* $o$ **do**
        Get $\{K^{10}_{2,2}, K^{10}_{2,3}$ from equations (7)
        Get $x$ from equation (10).
        **for** *Each candidates of* $n$ **do**
          Get $\{K^{10}_{1,2}, K^{10}_{1,3}\}$ from equations (6).
          Test equation (9).
          **if** *Satisfied* **then**
            **for** *Each values of* $\{K^{10}_{0,0}, K^{10}_{0,1}, K^{10}_{1,0}, K^{10}_{1,1}, K^{10}_{2,0}, K^{10}_{2,1}, K^{10}_{3,0}, K^{10}_{3,1}\}$ **do**
              Get $K^{10}$.
              Get the AES key $K$ using AES Key Schedule.
              **if** *P=Decrypt(K,C)* **then**
                Save K;
              **end**
            **end**
          **end**
        **end**
      **end**
    **end**
  **end**
**end**

### 4.3 Analysis of the Proposed Attack

There are total 20 differential equations in the proposed attack: 16 in four sets of equations (5), (6), (7), (8), and 4 more for equations (9), (10), (11), and (12). Each of these equations reduces the $2^{16}$ possible choices of right hand side (corresponding to two S-box output) to $2^8$ choices in the left hand side. Therefore, the reduction is given by $(\frac{1}{2^8})$. If there are $N$ differential equations then the reduction is given by $(\frac{1}{2^8})^N$. If $N$ equations contain $M$ unknown variables then the reduced search space is give by $(2^8)^{(M-N)}$. For more details on the analysis, one can refer to the paper [21]. We have 20 equations with 24 unknown variables; namely $m, n, o, p, w, x, y, z$ and 16 unknown key bytes. Therefore, we have $(2^8)^{(24-20)} = 2^{32}$ choices of final round key.

The time complexity of the attack is also $2^{32}$ as explained in the previous section.

**Note:**
In some cases the attacker may not have access to the plaintext. So, he can not perform brute-force search on the possible guessed keys. In that case the attacker need to uniquely determine the key. Our attack can be applied in such a situation with two faulty ciphertexts. Say the faulty ciphertexts are $(C_1^*, C_2^*)$. If we apply our proposed attack on these two faulty ciphertexts, we will have two sets of fault values $m_1, n_1, o_1, p_1, w_1, x_1, y_1, z_1$ and $m_2, n_2, o_2, p_2, w_2, x_2, y_2, z_2$. Each of these sets will produce corresponding 20 differential equations.

Now we can apply our attack on each of these sets in step by step fashion. In the first step we guess the values of $(w_1, m_1)$ and get $2^{16}$ choices of $K_{0,i}^{10}$ where $0 \leq i \leq 3$. For each of these choices we guess one value of $w_2$ and test equations (5) with the deduced key quartet $K_{0,i}^{10}$. If these two values satisfy the equations we accept them, else we discard them. There are 8 equations (two sets of equations (5) from two faulty ciphertexts ) and 8 unknown variables ( $m_1, m_2, w_1, w_2$ and $K_{0,i}^{10}$). This implies only one key candidate will satisfy the test.

In the second step we get the values of $z_1, z_2$ and guess the possible values of $p_1$, and $p_2$ each of which will produce $2^8$ choices $K_{3,i}^{10}$. Intersection of these two sets will uniquely determine $K_{3,i}^{10}$. Following the same technique in third and fourth steps we can uniquely identify rest of the two key quartets $K_{1,i}^{10}$ and $K_{2,i}^{10}$. So, finally we will have one choice of $K^{10}$. The time complexity of these attack is $2^{24}$ as initially we need to guess $w_1, w_2$, and $m_1$ to get the values $K_{0,i}^{10}$.

The attack analysis is quite obvious. One faulty ciphertext reduces the key space to $2^{32}$ from $2^{128}$. Therefore, two faulty ciphertexts will reduce the search space to $(\frac{2^{32}}{2^{128}})^2 \times 2^{128} = \frac{1}{2^{64}}$. This implies only the actual key will left and rest of the guessed keys will be discarded by the attack.

## 5 Experimental Results

In order validate our attack we performed extensive simulations. Some of the simulation results are presented in this section. We used 8 core Intel Xeon E5606 processor of 2.13 GHz speed running on Linux (Ubuntu 10.4). The attack code was written in C programming language and compiled with gcc-4.4.3 with O3 optimization. The simulation was performed over 100 times on different random keys. Some of the results are shown in Table 1. The attack takes less than 20 minutes to reveal the secret key. The first column of Table 1 represents the random 16-bytes keys which were attacked. The second column represents the number of possible key generated by the attack. The last column represents the total time taken by the attack which corresponds to generating possible keys and then performing brute-force search on them to get the master key.

**Table 1.** Experimental Results

| Random 128-bit Key | Number of Possible Keys | Running Time (Minutes) |
|---|---|---|
| aaf3100fdf183ef427464fbf4db85f7a | $3976380416 \approx 2^{31.88}$ | 17.65 |
| 5da4e5407bae5f94cc4a264bf694c0d2 | $8744409088 \approx 2^{33.025}$ | 18.533 |
| 19345421476b4e2b72191a845d30942a | $8424587264 \approx 2^{32.971}$ | 18.791 |
| 22615643211247530329 4a5bc2326a96 | $4018579456 \approx 2^{31.90}$ | 18.883 |
| 5da4e5407bae5f94cc4a264bf694c0d2 | $4223047680 \approx 2^{31.975}$ | 19 |

## 6 Comparison with the Previous Works

In this section we compare our attack with some of the existing attacks in these area. The first DFA on AES key schedule was proposed by Giraud [10]. Giraud's attack requires 250 faulty ciphertexts and five days execution time to retrieve the secret key. The attack proposed by Chen and Yen in [9], was an improvement over Giraud's attack where around 30 faulty ciphertexts were used. Peacham et. al. in [16] proposed an attack using 12 faulty ciphertext. DFA on AES key schedule using two faulty ciphertexts was first time proposed by Junko Takahashi et. al. [20], which reduced the AES key space to $2^{48}$ choices. Kim et. al. in [13], further improved the attack and reduced the key space to $2^{32}$ possible choices using two faulty ciphertexts.

Compared to these attacks our attack requires only one faulty ciphertext. The required brute-force search for our attack is 32-bit which is same as in Kim et. al.'s attack [13]. Therefore, the proposed attack required minimal faulty ciphertexts to mount an attack on AES key schedule. Table 2 shows the comparison.

**Table 2.** Comparison with existing attack on AES-128 key schedule

| Reference | Fault Model | Number of Faults | Exhaustive Search |
|---|---|---|---|
| [9] | Single byte fault | 22 to 44 | 1 |
| [16] | Multi byte fault | 12 | 1 |
| [20] | Multi byte fault | 2 | $2^{48}$ |
| [13] | Multi byte fault | 2 | $2^{32}$ |
| Our Attack | Multi byte fault | 1 | $2^{32}$ |

## 7 Conclusions

We proposed an improved attack on AES-128 key schedule. The attack require only one pair of fault-free and faulty ciphertexts. The proposed attack reduces

the AES-128 key space to 32-bit. The time complexity of the attack is $2^{32}$. In order to validate the attack we have provided extensive simulation results. The simulated attack retrieves the secret key on less than 20 minutes on an 8 core Intel Xeon E5606 processor at 2.13 GHz speed. This shows that the attack is indeed practical.

**Acknowledgment.** The authors would like to thank the anonymous reviewers for their comments and suggestions. The authors also extend their gratitude to the Department of Information Technology, India for funding this work.

## References

1. National Institute of Standards and Technology, Advanced Encryption Standard, NIST FIPS PUB 197 (2001)
2. Ali, S.S., Mukhopadhyay, D.: Acceleration of Differential Fault Analysis of the Advanced Encryption Standard Using Single Fault. Cryptology ePrint Archive, Report 2010/451 (2010), `http://eprint.iacr.org/`
3. Ali, S.S., Mukhopadhyay, D., Tunstall, M.: Differential Fault Analysis of AES using a Single Multiple-Byte Fault. Cryptology ePrint Archive, Report 2010/636 (2010), `http://eprint.iacr.org/`
4. Barenghi, A., Bertoni, G., Parrinello, E., Pelosi, G.: Low Voltage Fault Attacks on the RSA Cryptosystem. In: Breveglieri, et al. (eds.) [8], pp. 23–31
5. 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)
6. Blömer, J., Seifert, J.-P.: Fault Based Cryptanalysis of the Advanced Encryption Standard (AES). In: Wright, R.N. (ed.) FC 2003. LNCS, vol. 2742, pp. 162–181. Springer, Heidelberg (2003)
7. 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)
8. 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)
9. Chen, C.-N., Yen, S.-M.: Differential Fault Analysis on AES Key Schedule and Some Coutnermeasures. In: Safavi-Naini, R., Seberry, J. (eds.) ACISP 2003. LNCS, vol. 2727, pp. 118–129. Springer, Heidelberg (2003)
10. Giraud, C.: DFA on AES. In: Dobbertin, H., Rijmen, V., Sowa, A. (eds.) AES 2005. LNCS, vol. 3373, pp. 27–41. Springer, Heidelberg (2005)
11. Dusart, P., Letourneux, G., Vivolo, O.: Differential Fault Analysis on A.E.S. Cryptology ePrint Archive, Report 2003/010 (2003), `http://eprint.iacr.org/`
12. Fukunaga, T., Takahashi, J.: Practical Fault Attack on a Cryptographic LSI with ISO/IEC 18033-3 Block Ciphers. In: Breveglieri, et al. (eds.) [8], pp. 84–92
13. Kim, C.H., Quisquater, J.-J.: New Differential Fault Analysis on AES Key Schedule: Two Faults Are Enough. In: Grimaud, G., Standaert, F.-X. (eds.) CARDIS 2008. LNCS, vol. 5189, pp. 48–60. Springer, Heidelberg (2008)
14. 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)

15. 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)
16. Peacham, D., Thomas, B.: A DFA attack against the AES key schedule. SiVenture White Paper 001, October 26 (2006)
17. 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)
18. Selmane, N., Guilley, S., Danger, J.-L.: Practical Setup Time Violation Attacks on AES. In: EDCC, pp. 91–96 (2008)
19. 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)
20. Takahashi, J., Fukunaga, T., Yamakoshi, K.: DFA Mechanism on the AES Key Schedule. In: Breveglieri, L., Gueron, S., Koren, I., Naccache, D., Seifert, J.-P. (eds.) FDTC, pp. 62–74. IEEE Computer Society (2007)
21. Tunstall, M., Mukhopadhyay, D.: Differential Fault Analysis of the Advanced Encryption Standard using a Single Fault. Cryptology ePrint Archive, Report 2009/575 (2009), http://eprint.iacr.org/
22. 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)