Implementation and Comparative Analysis of the Fault Attacks on AES

Saleem Raza, Malik Najmus Saqib, Azfar Shakeel, and Majid Iqbal Khan
Department of Computer Science, COMSATS Institute of Information Technology, Pakistan

Abstract: *This research presents the survey, analysis, comparisons and implementation of the most threatening new kind of cryptographic attacks known as fault attacks or implementation attacks against Advanced Encryption Standard (AES) algorithm. AES algorithm is used in various applications and is considered the most secure against conventional cryptanalytic attacks which exploits the algebraic or mathematical weaknesses in the crypto-systems. Fault attacks are based on interrupting the execution of the algorithm in such a way that it produces faulty cipher output which can be analysed to break the algorithm. This research survey various fault attacks and provide implementation of three of them in detail for demonstration purposes. It mapped the complex mathematical analysis into programming algorithms for ease of implementation. At the end it compares various types of attacks based on our devised criteria of efficiency, flexibility and usability of the attack methods.*

Keywords: *Fault attack, AES, cryptanalysis.*

Received March 26, 2012; accepted May 22, 2012; published online August 5, 2012

1. Introduction

Cryptography is concerned with protection of private and confidential information. Cryptographic techniques evolve constantly because of constant threats being posed against them. Different kinds of attacks are always discovered with the advancement of technology. We are particularly interested in fault attacks due to two reasons. First, these attacks are totally different in their methodology and pose greater security threats than other attacks. Secondly, they have deep potential. Advanced Encryption Standard (AES) algorithm is chosen because nowadays it has been in most security processes. Consequently, cryptographic algorithms and their implementations have to be constantly improved in order to withstand contemporary attacks.

In 2000, NIST announced AES [11, 25] also, known as Rijndael (pronounced “Rhine-doll”) Algorithm developed by two Belgian cryptographers to protect digital information. Even some vendors (Checkpoint software Technologies Ltd, Cisco Systems Inc) planed to market AES based products firewalls before its announcement as a standard.

Rijndael is secure against all known conventional cryptanalysis attacks, which are successful on other algorithms. AES is symmetric block cipher, have simple design, highly efficient in term of space, time and also, much flexible (can run on various platforms and can operate on different key sizes 128bit, 192bits, and 256bits). The most common implemented version is AES128 because this key length is sufficient to provide security and requires less processing time than bigger key length versions. Various approaches in the key generation and expansion process of AES has been proposed in literature [1].

This research article is about comparisons and implementations of the new kind of attacks known as fault attacks on well known cryptographic algorithm AES. In such attack an adversary induces/injects faults i.e. erroneous data when AES is executing on a device and observes the reaction. This observation reveals secret information, Thus fault attacks have become serious security concerns.

2. Fault Based Attacks

Generally, fault based attacks lie under the category of active attacks. These attacks could further be divided into two categories when applied to cryptographic algorithms:

1. Simple Fault Attack (SFA).
2. Differential Fault Attack (DFA).

SFA was proposed in 1997, by Boneh *et al*. [9]. This attack was successfully applied against RSA cryptosystem. Several authors extended the idea and introduced various fault models.

After some time, a new kind of attack called DFA attack was introduced by Biham and Shamir [7], and it was successfully applied on secret key cryptosystem DES [9].

Attacking philosophy of both types of attacks is same. To attack disturbed the execution of cryptosystem by inducing errors in some particular time at some particular location. Then collect faulty cipher-texts which can be analysed and compared with correct cipher-text in order to get the secret information/key. Faults can be categorized in two ways:

1. Permanent.
2. Transient.

Permanent faults are also, called destructive faults. Here device circuit or memory cell is permanently changed/damaged (e.g., cutting data bus wire, freeing memory cell to constant value) while in transient fault (also, called provisional fault) device is just disturbed during its processing. It recovers back to its original behaviour after reset when fault simulation ceases. With transient faults, the attacker can use the device again and again and the attack can be repeated multiple times in order to collect sufficient information.

Faults induced on software are considered transient and such transient faults can be viewed from various angles.

1. *Error Type:* What type of error is being induced? It can be of two types:
   a. *Bit Error:* Single bit has been flipped via XOR operation in software or freezing any memory cell to 0/1 value in hardware.
   b. *Byte Error:* Whole byte has been changed by XOR operation the byte with some random data or freezing multiple cells.
2. *Error Location:* At which location, errors are being induced. Algorithm can contain many transformation and we can select any transformation for induction of wrong data in it. This location can be fixed or much flexible depending on the attack type.
3. *Time:* Some faults require to be induced at specific time during computations while other does not impose such restrictions.

3. Fault Based Cryptanalysis

Cryptanalysis is the study of methods to break cryptosystem but usually excludes methods e.g., exhaustive search of attack that do not primarily target weaknesses in the actual cryptosystems.

Most of the attacks and their cryptanalysis are based on the mathematical security of the cryptosystem. Modern attack take advantage of side channel information that can be derived from physical implementation of the cipher rather than exploiting some weakness of the cipher itself.

Fault attacks are a kind of side channel attacks and they occur by injecting the faults in the implementation (hardware/software) of the cipher [4]. That is during execution of particular cipher, faults are induced. This disturbs the normal execution behaviour and results in creation of faulty output. The attacker can guess secret key after small number of the fault injections and analysing faulty cipher texts. These attacks work on both symmetric and asymmetric types of ciphers. They can also, break the system faster than any other kind of the physical attack. Fault attacks are being studied constantly since 1996 in smart card industry. Initially, most of the proposed attack models were theoretical however these days, theoretical faults attacks are being converted into practice due to advancement in technology. Firstly, in 2001, Skorobogatov and Anderson [28]. Performed a practical attack by flipping a bit in the memory cell via photoflash lamp.

Various symmetric as well as public key cryptosystems are under the fault attacks in different ways. Bao *et al*. [3], presented transient fault to break a public key cryptosystem.

Fiat-Shamir and Schnorr proposed first theoretical active fault attack focussing on RSA cryptosystem [9]. This attack exploits Chinese Remainder Theorem (CRT) and Montgomery Multiplication method. It is easy to deploy. It requires single fault induction at any position to completely break signature device. Due to widespread use of RSA, it was real threat however wide range of paper explains countermeasures. Then focus of attacks shifted towards the cryptosystems whose security is based on the Discrete Logarithm problem. Thus, Elgamal Signature Scheme, Schnorr Signature scheme and DSA scheme were the prime targets under this category.

The attack on Schnorr signature scheme was also, extended with modification of the identification scheme. Attacks on RSA montgomery were also, suggested employing them on the signing key instead of the message [23]. Joye and Quisqater [17], proposed the attacks in on Generalized RSA-type cryptosystem. After that most of the fault attacks proposed by Klima and Rosa. Successfully determined private key of RSA and secret key of DSA [18].

Klima and Rosa [19], presented another attack known as Bellcore attack on the Fiat-Shamir scheme is shown. However, the Fiat Shamir scheme defends against it. In [31], another attack proposed by Voyiatzis and Serpares is presented that succeeded against both classical and precautious Fiat-Shamir schemes.

Fault attacks focus on changes from public to symmetric key system with Biham and Shamir [6]. This attack is known as DFA. It was succeeded against DES using only 50-200 faulty cipher texts. This attack is also, presented on an unknown algorithm of cryptosystem, called Skip Jack. This attack was also, performed on DES via permanent fault induction instead of transient fault induction. Unknown cryptosystems described in [26], are also, targeted by DFA attacks.

Jacob *et al*. [16], employ the faults and attack obfuscated ciphers in software and then secret material is extracted by avoiding de-obfuscation of the program code.

Various cryptographic algorithms are based on complex mathematical problems like discrete logarithm and factoring calculations. Initially, it was thought that fault induction attacks can only succeed on such types of systems. Biehl *et al*. [5], performed DFA on Elliptic Curve Cryptosystems (ECC) and Zheng [32] attack modern random number generators with faults. Similar types of attacks are also, presented on ECC. One such attack is proposed by Biehl *et al*. [5], by extending the DFA attack on RSA cryptosystem.

Fault attacks falls into two main categories of SFA and DFA. This research describes these fault attacks with specific example to get clear understanding.

3.1. Simple Fault Attack on Secret and Public Cryptosystems

Bell-core attack was proposed for a public key cryptosystem. With this attack an implementation of RSA which is based on CRT can be broken using a single wrong signature. Algorithm 1 shown the working.

Algorithm 1. fault attack on DES.

```
Input: d M, P, q
Output: S=Mdmod p.q
dp=d mod (p-1),
dp=d mod (q-1)
Compute
Sp = Md mod p
Compute
Sq = Mdp modq
S = CRT (Sp, Sq)
Execute RSA-CRT Algorithm
Without Faults
Execute again and inject fault in computation of Sq
Deduce a parameter P using correct result S and faulty
result S'
S mod q ≠ S' mod q
S mod p = S' mod q
```

Now it is easy to know the factors of N by:

$$GCD\ (S\text{-}S',\ N) = p \tag{1}$$

RSA security is based on factorization. After successful factorization, System has been broken. A SFA against secret key cryptosystem is against DES which is described in [28], DES secret key is stored in EEPROM when it is implemented in smart card. This key is transferred to memory during encryption and decryption. The attacker resets an entire byte of the key when key is being transferred to memory. If attacker could do it successfully, then the secret key can be obtained. For this eight steps need to be performed in order to get secret key as shown in Table 1. In each step, known plain text is encrypted with different number of bytes of the key. Consider cipher text S7 in the table. Attacker can extract the first byte of the secret key by checking all possible values of the first byte until S7 is obtained. Similarly, based on S6 second byte of the key can be found. This process continues until the attacker gets all 8bytes of the key.

Table 1. Acquiring secret key.

DES Key	Output
K0 = xx xx xx xx xx xx xx xx	S0
K1 = xx xx xx xx xx xx xx 00	S1
K2 = xx xx xx xx xx xx 00 00	S2
K3 = xx xx xx xx xx 00 00 00	S3
K4 = xx xx xx xx 00 00 00 00	S4
K5 = xx xx xx 00 00 00 00 00	S5
K6 = xx xx 00 00 00 00 00 00	S6
K7 = xx 00 00 00 00 00 00 00	S7

3.2. Differential Fault Attack

DFA attacks come after simple fault attacks. The idea was originated in 1996, when Boneh and Lipton from Bellcore conducted a new kind of cryptographic attack (DFA Attack) against public key cryptosystems. Biham and Shamir [6], showed that this attack can be extended to break almost any secret key algorithm.

Adi Shamir implemented this attack on a personal computer and single bit faults were induced in all rounds randomly. Complete last sub key (48bits) was found in less than 200 cipher texts. To recover the complete key (56bit), a simple way is that to guess via exhaustive search missing 8bits in all 2^8=256 ways. This attack works as follows: 1). A single bit error is induced during the computation, suppose in round 16 of DES. So, one bit in the right half of the data is flipped from 0 to 1 or vice versa. Now let the cipher text collected before the final permutation of DES. This will be faulty cipher text due to one bit error. Assuming we can also, collect correct cipher text (at the same point) without inducing error, only one bit in the right half of the cipher text will differ (before final permutation) between both of the cipher texts. 2). In final permutation, this single bit will enter in S-Box affecting the left half of the cipher text generated from one or two S-Boxes. Now if we see different distribution tables of these S-Boxes, the difference must be related to none zero entries. 3). At the end, the six key bit of each S-Box are guessed and any value disagreeing with the expected difference of these S-Boxes is discarded. Due to this phenomenon, this attack is referred to as differential cryptanalysis.

4. Fault Attacks on AES

Rijndael [11, 12], was selected as the AES algorithm and it replaced the DES. AES was designed after introduction of the crypt analysis on secret key cryptosystems. So, AES was made resistant against all known cryptanalysis. As an example, the original proposal prevents all timing attacks. It was known that careless implementation of the AES may result in successful timing attacks and the secret key can be derived easily [2]. This key can be derived by 3000 samples per key byte. The final proposal of the algorithm was aware of this and the attack was immune easily but fault attacks were proved successful even against AES.

Fault attacks can be induced permanently or temporarily in smart card. Examples of permanent fault include cutting data bus wire or freezing memory cells etc. Temporary faults include introducing abnormal voltages during AES execution. Faults can also, be induced at specific location (memory cell) or at specific time. Some of the attacks have been applied to smart cards in practice successfully.

Further any round can be disturbed via induction of faulty data. For example, if one byte is changed in sub-bytes operation, then this effect will cascade to all next three operations in the round. It will affect one to four bytes in output of that particular round resulting ultimately in faulty cipher-text. Faulty and fault free cipher texts can be collected and employ mathematical cryptanalysis to discover keys or portion of the keys. Some attacks also, belong to key expansion operation of Rijndael.

There are various ways to induce faults in devices implementing AES. Some of them are voltage spikes, clock glitches, overheating via excessive temperature, exposing to radiations and light. Interested reader should consult [8, 15, 21]. The focus of this article is on fault attacks induced in the software implementation of AES.

4.1. DFA Attacks on AES

New types of faults attacks known as DFA succeeded against AES even it was considered to combat all known attack when proposed. Even though DFA attacks target initially was feistal structure like DES but Dusart *et al*. [13], applied it successfully against AES, even though it does not have such a structure. Blomer and Seifert [8], presented additional fault based attack on AES.

Stricter model derives 128bit secret key by collecting 128bits faulty cipher texts, but require exact synchronization in time and space for errors injection. This model assumed that one can force the value of any chosen bit to Zero. It is shown by Biham and Shamir that applying this technique upon memory cells storing the key would make it trivial to get the key and the same is true for any other algorithm. The other less-stricter model extracts keys after collecting 256 faulty cipher texts. This second attack is implementation dependent and its principle is to turn timing attack on AES suggested by Koeune and Quisqater [20]. Into a fault based cryptanalysis.

In [14], Giraud discussed two fault attacks on AES. In the first one, assumption is that the faults can be induced on one bit before the last diffusion layer. About 50 cipher texts are needed to retrieve the full key. A second attack proposed by Giraud is on the whole byte and is more realistic to perform. Here fault is induced at different places including the key schedule and in 250 cipher texts. It would become possible to extract the key. Here it is assumed that attacker can choose the stage of the byte where the fault occurs but not exact location. The calculation to determine the exact location takes 5 days.

Dusart *et al*. [13], presented the four fault models and attack succeeded for all key sizes (128, 192, 256bits). They showed that 128bit key can be created with 10 pairs of faulty/correct messages. These attacks take advantages of byte fault occurring after shift row layer of 9th round. Authors show that five well located faults are necessary to retrieve the four key bytes.

There is also, another DFA attack against AES by Chien-Ning and Surg-Ming [10], that is very interesting and it covers all of the errors under its two fault models. For the first model they claimed that with only six faulty cipher texts, whole 128bit key can be found. However, the other fault model needs 1500 cipher texts to cover all faults. When two models are combined, all faults can be covered those occur in 9th round of encryption algorithm of the AES cryptosystem. It is generalized method as it does not assume any specific location or values of the occurred faults.

Chien-Ning and Sung-Ming focused on the fault attacks against AES key schedule by inducing single byte fault in round key instead of inducing faults in encryption or decryption algorithm. However, their fault model is based on the Giraud fault model. But with extension they were able to get keys in 22 faulty cipher texts and with less computational complexity. Also, they focus on countermeasures of their proposed attacks [22].

Another similar attacks focussing on key scheduling is presented in [27]. Totally new approach is adopted in [30], by devising some rules to get 80 bits of the secret key with just two pairs of correct and faulty cipher text. They claim that following these rules make the process efficient because we get rid of the solving complex simultaneous equations. They extended the attack and would be able to get extra 8bit of secret key just combining with previous attack, they are able to get 88bit of secret keys effectively [29]. Remaining 40bits are obtained by brute-force approach.

5. Design of Algorithm

The process explained by Dusart *et al*. [13], is mathematical and described in very complex way to find more difficulties for a computer science student to convert it into some proper procedure so that it can be implemented in easily in any programming language.

5.1. Algorithm 1: From Dusart *et al*. [13]

- *First Step:* The first step is to generate a set called *E1* described as proposition 1 in [13]. This set depends on the mapping $E1=x^2+x$. The way this set is computed is as follows:

1. Select each value from $x=1$ to 255.
2. Compute x^2+x.
3. Look to see if a new unique value is obtained. If yes store in an array $E1$.

- *Second Step:* Now, next step is to generate a set called $S_{(c,\varepsilon')} \rightarrow S_{cep}$. It is also, defined in [13], as proposition 1. The way this set is computed is as follows:
  1. First generate a set for each of the "'c" value and the corresponding "ε'" value.
     a. "c" value refers to the {2, 1, 1, 3} for the first Mix Columns.
     b. "ε'" refers to the "error" found between the no-error ciphertext and faulted ciphertext.
  2. There are 4 sets above- for each pair of "c" and "ε'".
  3. Now find a set y that is an intersection of all 4 sets.
  4. Set y represents all potential faults.
- *Third Step:* Now-for each value in the set $S_{(c,\varepsilon')} \rightarrow S_{cep}$ (called θ in proposition 5 [13]), we want to find a solution to the equations:

$$\theta = t^2+t \qquad (2)$$

  These solutions are called α and β. They follow the relationship $\beta=\alpha+1$.
- *Fourth Step:* Now we are ready to generate potential values of K_{10} (key of 10th round) using the two solutions found above as α and β. one value of K_{10} is=Sub_Byte(c*e*α). The other value is=Sub_Byte(c*e*β). Finally, store the potential K_{10} keys and sort to find only the unique ones. This is now our list of potential K_{10} values. This process can be repeated with multiple attacks until only one value left that will be the correct key value.
- *Full Key recovery:* Once a correct single K_{10} is generated it is possible to recover the entire key by reversing the key expansion algorithm. This is shown by Dusart *et al.* [13].

5.2. Algorithm 2: From Giraud [14]

The main idea of the algorithm is a follows:

1. Induce a bit fault in the final round of encryption.
2. A bit fault means error on a single bit.
3. Then analyse the results of final ciphertext and compare to fault free ciphertext.

The algorithm proceeds as follows:

1. Do an exhaustive search of possible M_9 bytes (0-255) and all possible bit faults (8total), where M_9 is cipher after 9th round.
2. Find the error value that is the difference between clean ciphertext (called C) and faulty ciphertext (called D).
3. Store the list of potential values in an array.

This is now repeated for each byte of 9th round cipher text. Then find the common value (intersection) of all possible values. That is:

1. Calculate error value for each fault value.
2. Find the "intersection" of potential M_9 values.
3. Once we have a single value of M_9.
4. Calculate K_{10} from equation below:

$$C = ShiftRows(SubBytes(M^{9}))\mathring{A}K^{10} \qquad (3)$$

The process can be repeated for each byte of K_{10}. Then finally the entire key is recovered.

5.3. Algorithm 3: Moradi [24]

The algorithm proceeds as follows:

1. *First Step:*
   a. Generate the set called S1. This is the set where at least one of the faults bytes is zero.

$$S1 = \{ \varepsilon' : (e_1, e_2, e_3, e_4) \mid \forall e'_i \neq 0 ; (1 \leq i \leq 4),$$
$$\exists \varepsilon : (e_1, e_2, e_3, e_4)\ \varepsilon\ FM1;$$
$$MixColumn(\varepsilon) = (\varepsilon')\}$$

   b. Using the possible fault bytes generate all possible first column vectors (column vector).

   If only byte fault then there are 256 vectors (0 to 255 total byte values, each vector for each byte).

2. *Second Step:*
   a. The most important part of the algorithm is the generation of the set called "I".
   b. Since fault in the first column affects only the first column after the MixColumn step. It is possible to generate a complete set of potential values of M_9.
   c. This is generated using the set $S1$ above.
   d. Do an exhaustive search of possible M_9 columns.
   e. Once we have a single column we can compute K_{10}.
   f. Find the error value that is the difference between clean ciphertext (called C) and faulty ciphertext (called D).
   g. Store this list of potential values in an array.

   This is now repeated the above steps for each column. Then we find the common value (intersection) of all possible values of K_{10}. Once we have K_{10} we use the key recovery from K_{10} to get the original key.

6. Implementation

We have implemented the following algorithms to demonstrate that fault attacks really works. AES implementation is done before the implementation of below algorithms:

1. Fault attack defined by Dusart *et al*. [13].
2. Fault attack defined by Giraud in [14].
3. Key recovery from bit fault attack defined by Giraud [14].
4. Key recovery defined by Moradi *et al*. [24].
5. Full Key recovery (Dusart) by assuming that partial key (round key) recovered in Dusart.

This section briefly explains the implementation aspects for 1, 2, 3, and 4 and rest is based on same aspects. MATLAB is used in the implement of these algorithms. These algorithms manipulate the data in form of matrices and MATLAB is most suitable for this kind of work due to its matrix-orientation.

1. *Fault Attack Implementation (K9 Attack) Defined by Dusart:* How to induce fault into any state byte is defined theoretically in section 2.1.1 of this report. For implementation, we set mode bit in aes_main file to 1. Fault induction is done in this way:

K9byte=bin2dec('00011110') (4)

After checking that round is 9 then we add this into the first byte of state at position (1, 1) via this change of code

```
if (mode==1)
   if (i_round == 9),
      state(1,1) = bitxor( state(1,1), k9byte);
   end
end
```

2. *Fault Attack Implementation Defined by Giraud:* How to induce just one bit fault in any location is tricky process. How to change one bit? It can be done via XOR operation that bit with 1. Let suppose we want to change a bit in a byte at location bfa_i and bfa_j. The code snippet is given below:

```
bfa=bin2dec('10000000');
when we want to change any bit in any byte, we will
bitxor with  this value of bfa.
mode variable will be set to  2 for this attack.
% If BFA is done -- Bit fault attack
if mode==2,
   state(bfa_i,bfa_j) = bitxor( state(bfa_i,bfa_j), bfa);
end
```

3. *Key Recovery from Bit Fault Attack defined by Giraud:* The main file from Giraud attack recovery is Get_K10_BFA.m

C10 is cipher output, D10 is the faulty output. If any byte is faulty, we can find it by XOR of C10 and D10. In our code, we simply take byte (1, 1) and induce three error cases for it. Thus faulty bytes in D matrix are D10_1, D10_2, and D10_3. We find the error (Err) for each of them by XOR operation corresponding bytes. This is done using following code:

```
% First compute the Err vector
nerr = 3;
Err = 0*ones(nerr,1);
Err(1) = bitxor(C10(1,1),D10_1(1,1));
Err(2) = bitxor(C10(1,1),D10_2(1,1));
Err(3) = bitxor(C10(1,1),D10_3(1,1));
disp('Error bytes for the attacks are ...');
disp(' ');
disphex10('Error bytes =  ',Err);
disp(' ');
```

'disphex' converts integer values into hexadecimal format. Errors values are calculated via 'bitxor' of correct and faulty bytes and stored in three locations of the array. 'ebit_hex' is an array representing 8-possibilities of 1 bit error in a byte.

```
ebit_hex = {'01' '02' '04' '08' '10' '20' '40' '80' };
ebit = hex2dec(ebit_hex);
```

We iterate through all possible values of M9 byte (0…255) and through all possibilities of bit error (1…8). Then for all possibilities, we have calculated their substitution bytes via subbyte () and store the results in B1. Similarly, for each inducing of error bits in M9 byte, we determined subbytes values and store the results into B2. Then 'ecal' determines error value by XOR operation between correct and faulty B1 and B2 arrays as shown below:

```
for k=1:nerr,
// loop over all possibilities
  for i=1:2^8,
  // i loops over all possible values of M9 (1..255)
    for j=1:8,
      // j loops over all possible values of bit err
     //so let M9_pos be the potential M9 byte
    M9_pos= i-1;
     // the corresponding "C" byte (i.e. without fault)
     B1=sub_bytes( M9_pos, s_box);
    // the corresponding D byte (i.e., with faults)
    B2=sub_bytes( bitxor(M9_pos, ebit(j)), s_box);
    //so the error with this M9_pos and this
    // bit error is ecal
    ecal=bitxor( B1, B2);
```

Also, we check for each value if the calculated error in the error stored in 'Err' array. If this is the case, store the error value along with its position and then determine potential M9 bytes (for each error induced, their will be many values as potential candidates for M9 byte). This is done via adding these lines to previous code snippet

```
// if the calculated error IS the error then we have
potential M9
// jth position is being stored as well along with potential
bytes
      if (ecal == Err(k)),
          nf(k) = nf(k)+1;
          M9list(nf(k),k) = M9_pos;
          Elist(nf(k),k) = ebit(j);
      end
```

At end we intersect all sets of potential candidates and common value of M9 will be the actual byte (because all errors are originating from M9 byte) as shown below:

```
// Now find the intersection of the potential M9 byte and find the intersection of the sets
xx = intersect(M9list(1:nf(1),1),M9list(1:nf(2),2));
M9_found = intersect(xx,M9list(1:nf(3),3));
```

Now we can find one byte of key K10 via applying the equation given in paper like this:

```
K10_1_1 = bitxor( sub_bytes( M9_found, s_box), C10(1,1));
```

After we find K10 (1, 1), it shows that this is correct value. In the same way, we can find other key bytes by applying bit errors in other locations. Once we find the complete K10 (Last round key), we can get initial secret key by implementing the pseudo-code given at last page of Dusart's paper.

4. *Key Recovery Defined by Moradi:* Moradi algorithm is based on generalized attack i.e., any fault attack that changes at most three bytes in first column of mixcolumns. Each byte in input to mixcolumn affects four bytes in its output.

Moradi does not define any specific fault attack because it is general model. It just says that if fault in effecting first column of mixcolumn in 9th round, then the key recovery procedure explained in paper can determine the correct key bytes. So, we based the implementation by taking fault model described in Dusart's paper. We took all relevant correct faulty bytes given in Dusart's paper. So, code snippet is:

```
// Faulted bytes from the encryption algorithm in dusart
fb_hex={'de' '3b' 'c2' '62'};
fb=hex2dec(fb_hex); // a column vector
nfb=max(size(fb));
// Fault free bytes -- from the encryptoin Appendix B in standard
fc_hex= {'39' '6a' '85' 'fb'};
fc= hex2dec(fc_hex); % a column vector
nfc = max(size(fc));
// Faults – also, from the encryption algorithm (called "epp" in Moradi)
epp_hex= {'e7' '51' '47' '99'};
epp= hex2dec(epp_hex);      % a column vector
nepp=max(size(epp));

// The correct "B" vector bytes –first column at start of round 10 encryption algorithm
B10_hex={'eb' '40' 'f2' '1e'};
B10 = hex2dec(B10_hex);
```

So, the focus is on key recovery assuming faulted bytes and error primes are known. This algorithm works on exhaustive search methodology. We calculate all values of applying mixcoumn operation on all possible bytes for the first coulumn of input to MixColumn via genS1 function. The important code snippet is:

```
% for each potential byte error compute a column of S1,j is index into cmat array
%cmat contain the 4 constant values used in MixColumn operation for mixing the first column of input state matrix
for i=1:255,
   for j=1:4,
      S1(j,i) = poly_mult( i, cmat(j), mod_pol);
   end
end
```

Now we need to generate I matrix shown in equation 21 [14], with the help of equation 20 [14]. Then generates the set E1 and helps to create I matrix. Now it is iterated through all possible values of I matrix and check if we can find correct B vector values (for single column of M9). Once the correct B vector is found, we can apply equation 22 [14], and equation 23 [14] to determine Key byte. The process is repeated until the key for 10th round (K10) is found. Once K10 is found, we trace back the key expansion algorithm to find the original key.

7. Comparisons of Algorithms

The implemented fault attack algorithms are compared based on the following criteria.

7.1. Flexibility

It is defined as an ability to change the components of the attacks (i.e., encryption, decryption and key scheduling), their location (i.e., function or round) and timings.

The attacks presented by Dusrat *el al*. [13], cover both components these are encryption/decryption and key scheduling. Their second attack actually presents three sub models to cover more fault models. One sub model applies on key expansion and the other two applies on plain text at round 8 and round 9 respectively.

The attacks presented by A. Moradi *et al*. [24], claims to cover all attacks in its two general models. All attacks up to MixColumns of 9th rounds are covered under their generalized models. It is the most flexible attacks of all kind because it presents the attacks under generalized models. The models cover all faults that can occur in 9th round of encryption. Giraud's presented attack in [14], is not much general as the above two and cover simple model of static attack but it includes other merits.

Flexibility can be expanded to include AES three modes (128, 192 and 256bits). Giraud's attack works for all modes while Moradi *et al*. attack has been implemented just to recover 128bit key.

7.2. Efficiency

The efficiency could be space-wise and time-wise. Space wise efficiency is less focused when attacks are demonstrated but time is much critical. Some attack models can recover secret information faster than other models. This is usually measured by considering that how many cipher or plaintext pairs are needed to get secret key information from any model.

Attacks presented by Moradi *et al*. [24], are generalized First model require just six faulty texts to get secret key information for the last round key. The time to get this is less than 10 seconds. However, the second model requires 1495 faulty texts. Comparing this with Giraud's presented attacks later is the most efficient as for its two models. It just requires 50 and less than 250 cipher texts respectively to determine secret key information.

Note that we are considering external efficiency instead of internal. It would be possible to run the key recovery algorithm for long time but we would need to input just two cipher text in it to get required key. It would be assumed to be more efficient but there might be contrary results if algorithms spends considerable amount of time internally to perform all operations.

7.3. Feasibility

Feasibility is the determination that a procedure can be successfully accomplished in the required time frame. Almost all attacks are feasible that have been performed but we would need to define less or more feasible.

Giraud *et a*l., defines two attack models. First attack is Bit Fault Attack (BFA) that is less feasible as very sophisticated hardware is needed to induce just one bit fault at proper location while in the same paper the second attacks model was actually put into practice on smart cards. There is also, an attack by Dusart *et al*. that is less feasible in hardware and takes 5 days to get exact location of error induced. But if we could find the location, then it is much flexible after that and can recover keys within minutes.

8. Conclusions

Although, AES is prominent crypto standard widely used nowadays in various applications and is secure against all conventional cryptanalytic attacks, yet fault based cryptanalysis poses serious threats against it. By implementing various attacks and comparing different fault models, it is clear that such kind of attacks is practical. Most of the attack models allow us to obtain full AES-128bit key.

Although, we have observed that most of the models are difficult to implement and in theory they are analysed to show that these attacks work. We have observed that most of such work is evolved theoretically. In this work we have implemented some fault attacks, which are theoretical in literature, to check their practicability. With rapid evolution of technologies, it would be possible in future to convert those theoretical models into practical ones so it is important that counter work should be done side by side.

Numerous countermeasures have been proposed to avoid, prevent or detect fault attacks. Malkin *et al*. [22], has provided a good comparison of various fault attack countermeasure. It also, provides a thorough overview of known strategies, with an emphasis on AES, to thwart fault attacks. But, attacks models outweigh counter attack methods and with rapid development of technology, there will be more threats than we face today.

9. Future Work

The attacks always improve and new approaches are coming into existence. This work can be extended to propose countermeasures against fault attacks.

Another work that can be done in future is to propose fault models and countermeasures against AES192, and AES256 versions. Currently, AES128 is sufficient to use due to its strong security but in future when more powerful processors would be available, there would be definitely a need to shift to other long-key length versions of the AES. Also, when we combine various facets of approaches, we can device some efficient approach to perform attack. Various faults attacks methods can be combined to get the secret information in more effective ways. Some work has also, been started in this direction where researchers are combining collision and fault attack and tries to get the secret information without obtaining faulty cipher text.

However, whatever fault model is proposed, due to sophisticated fault injection equipment and complexity of the process, it does not put AES aside. If there can be any method that just analyse various cipher texts produced without injection of fault and can give some clues about the secret information then we think it would completely cease AES. Similarly, rigorous side channel analysis techniques are being combined with mathematical analysis techniques to uncover secret information.

References

[1] Ajlouni N., El-Sheikh A., and Rashed A., "A New Approach in Key Generation and Expension in Rijndael Algorithm," *International Arab Journal of Information Technology*, vol. 3, no. 1 pp. 35-41, 2006.

[2] Anderson R. and Kuhn M., "Low Cost Attacks Attacks on Tamper Resistant Devices," *in Proceedings of Security Protocols Workshop, Springer Lecture Notes in Computer Science*, France, vol. 1361, pp. 125-136, 1997.

[3] Bao F., Deng H., Han Y., Jeng B., Narasimhalu D., and Ngair T., "Breaking Public Key Cryptosystems on Tamper Resistant Devices In

The Presence of Transient Faults," *in Proceedings of the 5th International Workshop on Security Protocols*, France, pp. 115-124. 1998.

[4] Bar-El H., Choukri H., Naccache D., Tunstall M., and Whelan C., "The Sorcerer's Apprentice Guide to Fault Attacks," *in Proceedings of IEEE*, vol. 94, no. 2, pp 370-382, 2006.

[5] Biehl I., Meyer B., and Muller V., "Differential Fault Attacks on Elliptic Curve Cryptosystems," *in Proceedings of the 20th Annual International Cryptology Conference on Advances in Cryptology*, USA , pp. 131-146, 2000.

[6] Biham E. and Shamir A., "Differential Fault Analysis of Secret Key Cryptosystems," *in Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology, Lectures Notes in Computer Science*, USA, vol. 1294, pp. 513-525, 1997.

[7] Biham E. and. Shamir A., "A New Cryptanalytic Attack on DES: Differential Fault Analysis," avalible at: http:// cryptome.org/jya/dfa.htm, last visited 1996.

[8] Blomer J. and Seifert P., "Fault Based Cryptanalysis of the Advanced Encryption Standard," *in Proceedings of Computer Aided Verification the 15th International Conference*, USA, vol. 2742, pp. 162-181, 2003.

[9] Boneh D., Demillo A., and Lipton J., "on the Importance of Checking Cryptographic Protocols for Faults," *in Proceedings of the 16th annual international conference on Theory and application of cryptographic techniques*, Berlin, vol. 1233, pp. 37-51, 1997.

[10] Chien-Ning C. and Sung-Ming Y., "Differential Fault Analysis on AES Key Schedule and Some Countermeasures," *in Proceedings of the 8th Australasian conference on Information security and privacy*, Australia, pp. 118-129, 2003.

[11] Daemen J. and Rijman V., " The Block Cipher Rijndael," *in Proceedings of Smart Card Research and Applications, Lecture Notes in Computer Science*, Belgium, pp. 288-296, 2000.

[12] Daemen J. and Rijmen V., "AES Proposal Rijndael, the First Advanced Encryption Standard," *Candidate Conference*, NIST, 1998.

[13] Dusart P., Letourneus G., and Vivolo O., "Differential Fault Analysis on AES," *in Proceedings of the 1st International Conference on Applied Cryptography and Network Security, Lecture Notes in Computer Science*, China, vol. 2846, pp. 293-306, 2003.

[14] Giraud C., "DFA on AES," *in Proceedings of the 4th International conference on Advanced Encryption Standard*, Germany, pp. 27-41, 2004.

[15] Gutmann P., "Data Remanence in Semiconductor Devices," *in Proceedings of the 10th Conference on USENIX Security Symposium*, USA, vol. 10, pp. 4, 2001.

[16] Jacob M., Boneh D., and Felten E., "Attacking an Obfuscated Cipher by Injecting Faults" *in Proceedings of ACM workshop on Digital Rights Management*, USA, pp. 16-31, 2002.

[17] Joye M. and Quisquater J., "Attacks on Systems using Chinese Remaindering," *Technical Report CG 1996/9*, Belgium, 1996.

[18] Klima V. and Rosa T., "Further Results and Considerations on Side Channel Attacks on RSA," *in Proceedings of the 4th International Workshop Redwood Shores Cryptographic Hardware and Embedded Systems-CHES*, USA, pp. 244-259, 2002.

[19] Klima V. and Rosa T., "Attack on Private Signature Keys of the Open PGP Format," available at: http://eprint.iacr.org/2002/076. pdf, last visited 2004.

[20] Koeune F. and Quisquater J., "A Timing Attack Against Rijndael," *Technical Report CG-1999/1*, Universite Catolique de Louvain, 1999.

[21] Maher P., "Fault Induction Attacks, Tamper Resistance, and Hostile Reverse Engineering in Perspective," *in Proceedings of the 1st International Conference on Financial Cryptography, Lectures Notes in Computer Science*, British West Indies, vol. 1318, pp. 109-121, 1997.

[22] Malkin G., Standaert X., and Yung M., "A Comparative Cost/Security Analysis of Fault Attack Countermeasures," *in Proceedings of the 2nd Workshop on Fault Detection and Tolerance in Cryptography Edinburgh*, UK, pp. 109-123, 2005.

[23] Marc J. and Jean-Jacques Q., "Faulty RSA Encryption," *Technical Report CG-1997/8*, UCL Crypto Group, 1997.

[24] Moradi A., Mohammad T., Manzuri S., and Mahmoud S., "A Generalized Method of Differential Fault Attack Against AES Cryptosystem," *in Proceedings of the 8th International Workshop Cryptographic Hardware and Embedded Systems-CHES*, Japan, pp. 91-100, 2006.

[25] NIST, "Advanced Encryption Standard," *Federal Information Processing Standards Publication FIPS-1997*, 2001.

[26] Paillier P., "Evaluating Differential Fault Analysis of Unknown Cryptosystems," *in Proceedings of the 2nd International Workshop on Practice and Theory in Public Key Cryptography*, Japan, pp. 235-244, 1999.

[27] Peacham D. and Thomas B., "A DFA Attack Against the AES Key Schedule," Available at http://www.siventure.com/pdfs/AES_KeySchedule_DFA_whitepaper.pdf, last visited 2011.

[28] Skorobogatov P. and Anderson R., "Optical Fault Induction Attack," *in Proceedings of the 4th International Workshop Redwood Shores*

Cryptographic Hardware and Embedded Systems-CHES , vol. 2523, pp. 13-15, 2002.

[29] Takahashi J. and Fukunaga T., "Differential Fault Analysis on the AES Key Schedule," available at: http://eprint.iacr.org/2007/480. pdf, last visited 2007.

[30] Takahashi J., Fukunaga, T., and Yamakoshi K., "DFA Mechanism on the AES Key Schedule," *in Proceedings of Workshop on Fault Diagnosis and Tolerance in Cryptography*, Vienna, pp. 62-74, 2007.

[31] Voyiatzis G. and Serpanos N., "A Fault Injection Attack on Fiat-Shamir Cryptosystems," *in Proceedings of the 24th International Conference on Distributed Computing Systems Workshops*, pp. 618-621, 2004.

[32] Zheng Y., "Breaking Real World Implementations of Cryptosystems by Manipulating Their Random Number Generation," *in Proceedings of the 29th Symposium on Cryptography and Information Security*, Japan, pp. 1-7, 1997.

Saleem Raza received his MSc degree in computer and network security from Middlesex University London. He also, did PGCert in Engineering-Advanced computing from University of the Bristol, UK. He has been teaching for four years at various national and international institutions and colleges in UK and Pakistan. His research interests include network security and cryptography. Also, interested in enhancing learning of university students, improvements in curriculum, and databases.

Malik Najmus Saqib received his MS degree in software engineering from COMSATS Institute of IT Pakistan, in 2003. He completed his PhD studies from Vienna University of Technology Austria in 2009. His PhD work was on security issues in mobile/multi agent system. He also worked as a research in University of California at Davis, USA. He is in technical program committee of international conferences and invited speakers in various workshops. His research focuses information security, security issues in wireless sensor network.

Azfar Shakeel is working as Lecturer in Computer Sciences Department, COMSATS Institute of Information Technology, Pakistan. He has done MS-IT from NUST, SEECS, MCS from University of Arid Agriculture Rawalpindi and MBA from International Islamic University Islamabad. He has several years of university level teaching experience and has research interests in information security, cryptography and cellular automata and its applications.

Majid Iqbal Khan received his BSc degree with majors in Mathematics and Physics in 2001 and MS in software engineering in 2004. He obtained his PhD in wireless sensor networks from University of Vienna, Austria, in 2009. He has been enjoying a distinguished carrier in both research and academics. He is author of several research articles. He is a member editorial board of reputed international conferences. His research interest includes network optimization, network security and internet of things.