A Biased Fault Attack on the Time Redundancy Countermeasure for AES

Sikhar Patranabis, Abhishek Chakraborty, Phuong Ha Nguyen and Debdeep Mukhopadhyay

Department of Computer Science and Engineering
IIT Kharagpur, India
sikharpatranabis@gmail.com, abhishek.chakraborty@cse.iitkgp.ernet.in,
phuongha@gmail.com, debdeep@cse.iitkgp.ernet.in

Abstract. In this paper we propose the first practical fault attack on the time redundancy countermeasure for AES using a biased fault model. We develop a scheme to show the effectiveness of a biased fault model in the analysis of the time redundancy countermeasure. Our attack requires only faulty ciphertexts and does not assume strong adversarial powers. We successfully demonstrate our attack on simulated data and 128-bit time redundant AES implemented on Xilinx Spartan-3A FPGA.

Keywords: Cryptanalysis, Time Redundancy, Biased Faults, AES

1 Introduction

Implementation attacks on secure embedded systems come in different flavors. One of these is the Side-Channel Analysis (SCA) such as *Differential Power Analysis* [8]. The other popular variety is the active Fault Analysis (FA) involving injection of faults into cryptographic systems and analysis under different fault models [2]. Attacks such as the Differential Fault Intensity Analysis (DFIA) [4] have in fact combined DPA with fault injection principles to obtain biased fault models. The advantage of a biased fault model lies in the ability of the adversary to derive an intermediate key-dependent state variable under several key hypotheses. The correct key hypothesis produces small changes to the faulty state while incorrect ones infer big, random changes.

This work attacks the time redundancy countermeasure using a biased fault model. The model is not as strict as some proposed earlier, such as stuck-at-zero or stuck-at-one faults [3]. The time redundancy technique is as an effective countermeasure, in which an encryption is followed by a redundant encryption, and in the event of a mismatch, *the faulty ciphertext is either suppressed or replaced by a random ciphertext*. Literature proposes time redundancy as a classical fault tolerance technique [11], [10] with the assumption of a uniform unbiased fault distribution. For a time redundant AES, in order to obtain the faulty ciphertext, the adversary must introduce exactly the same fault in both the actual

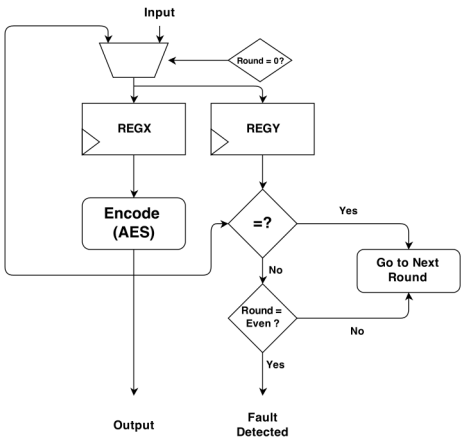


Fig. 1: Time redundancy

and redundant round cycles. When the fault distribution is unbiased (as classically assumed) the probability of occurrence of this event is very low. But a biased fault model augments this probability to the extent that it is feasible to obtain sufficient number of faulty ciphertexts to recover the key, while using a practical number of fault injections, even in the presence of the time redundancy countermeasure.

This work also assumes that we are operating only on faulty ciphertexts unlike traditional DFA which requires fault-free ciphertexts as well [1], [9], [6], [12], [14]. The proposed attack, like Differential Fault Intensity Analysis (DFIA) [4] targets an affected state variable by a biased fault injection methodology to retrieve the key. Our contributions are threefold: First, we develop a formulation for the degree of biasness in the fault distribution. Second, we propose fault models for biased faults and demonstrate actual fault attacks on a real life AES implementation with the time redundancy countermeasure. Finally, we establish through simulations and real life experiments that the number of fault injections required to defeat the time redundancy countermeasure is inversely proportional to the biasness of the fault induced.

2 Related Work

2.1 The Time Redundancy Countermeasure

Figure 1 illustrates the use of time redundancy in fault detection. Time redundancy is a fault tolerance technique that uses additional time to perform the functions of a system multiple times and compares the results to detect faults if any. A particular advantage of this approach is its low area overhead. The basic time redundancy technique has essentially three important aspects - repetition of function computation, storage of results of original and redundant computations and comparison of results for fault detection. In ciphers, time redundancy

is often used for concurrent error detection(CED) against DFA by repeating each round twice and comparing the results. Previous research has proposed some countermeasures to fault attacks using time redundancy. These include re-computation [11] as well as double data rate computation [10].

2.2 Fault Attacks on AES

Recent research has focused on two broad categories of fault analysis of AES - attacks that require correct and faulty ciphertext pairs, and attacks that require faulty ciphertexts only. The first category principally includes Differential Fault Analysis(DFA). In DFA, the adversary compares the response of the cipher with and without fault injections [1], [12], [14], [6]. The other category of fault attacks on AES require only faulty ciphertexts to retrieve the key, as proposed by Fuhr et. al. [3]. The attack uses stuck-at fault models and depends on the degree of control the adversary has on the distribution of the injected fault. A very similar approach proposed by Ghalaty et. al. is the Differential Fault Intensity Analysis (DFIA) [4] that uses a biased but slightly less restrictive single byte fault model. Both these approaches make several key hypotheses on the affected state bytes in order to retrieve a hypothetical value whose distribution is strongly biased.

Our proposed attack uses a biased fault model to attack the time redundancy countermeasure for AES-128 using faulty ciphertexts only. The reason is that recovering the key using only faulty ciphertexts is widely believed to be more challenging. However, similar attack procedure using biased fault models can also be developed for the former scenario since we can always obtain fault free ciphertext as well. Biased fault models expose a significant vulnerability of the classical time redundancy countermeasure. Unlike in a uniform fault distribution, a biased fault distribution implies that the adversary can introduce the same fault in both the normal and the redundant computation cycles with high probability. This reduces the number of fault injections required per faulty ciphertext. As in DFIA, our attack achieves the desired fault distribution using clock glitches at various frequencies.

3 Fault Model and Fault Injection Set Up

In this section, we describe the fault model used for our attack and the fault injection set up employed to achieve this fault model.

3.1 Fault Model

Depending on the type and method of fault injection, different types of faults may occur with varying granularity such as single bit upsets, multi bit upsets, single and multi byte upsets, and diagonal upsets. Some previous works have considered *random effect on one byte*, where a single state byte may have changed to any random value [5], [1], [7], [14]. However, such a fault model has a uniform distribution. More recent work [4] has demonstrated that single-bit, two-bit,

Table 1: Fault Model Description

(a) The Fault Model

Symbol	Fault Model
FF	Fault Free
SBU	Single Bit Upset
SBDBU	Single Byte Double Bit Upset
SBTBU	Single Byte Triple Bit Upset
SBQBU	Single Byte Quadruple Bit Upset
OSB	Other Single Byte Faults
MB	Multiple Byte Faults

(b) Impact of fault location precision

Fault Model	Faults Possible(n) (Situation-1)	Faults Possible(n) (Situation-2)
SBU	8	128
SBDBU	28	448
SBTBU	56	896
SBQBU	70	1120
OSB	93	744

three-bit and four-bit upsets are achievable using clock glitches, and that one can control the granularity of fault injection by varying the fault intensity. We have ourselves verified that such faults can be achieved in hardware implementations of AES-128 via introduction of clock glitches at varying frequencies (refer 3.2).

For further discussions in this paper , we distinguish between major classes of faults that covers the entire possible fault state. Table 1a summarizes these categories. Our experiments have shown that SBU is the most suitable fault model for our attacks on time-redundant AES implementations. However, we also present results for SBDBU, SBTBU and SBQBU to show the impact of fault model granularity on the performance of our attacks. Note that the degree of control that the attacker has on the fault location impacts the fault models in terms of the number of possible fault (N) under that fault model. We distinguish between the following two situations - Situation-1 when the attacker has perfect control over the faulty byte and Situation-2 when the attacker does not have control over the faulty byte.

In the case of single byte faults, if k be the number of bit upsets in the target byte, then the number of possible faults in either scenario is different. In Situation-1, any k bits of the fixed target byte is affected, so number of possible faults is $\binom{8}{k}$. In Situation-2, however k bits of any target byte could be affected, so number of possible faults is $16\binom{8}{k}$, which is 16 times greater than in Situation-1.

Table 1b captures the number of possible faults under various fault models in both situations. Evidently, precision in terms of fault location restricts the set of possible faults under a fault model significantly. Note that n is the total number of faults possible under the fault model.

3.2 Fault Injection Set Up

Figure 2 describes our set up for fault injection in time redundant AES-128.

The set up consists of an FPGA (Spartan-3A XC3S400A), a PC and an external arbitrary function generator (Tektronix AFG3252). The FPGA has a DUT (Device Under Test) block, which is a time-redundant AES implementation. Faults are injected using clock glitches and the fault intensity is controlled by increasing/decreasing the glitch frequency. The system has two clock signals - clk_{slow} and clk_{fast}, derived from an external clock signal clk_{ext} via a Xilinx Digital Clock Manager (DCM) module. The clk_{ext} is generated by the external

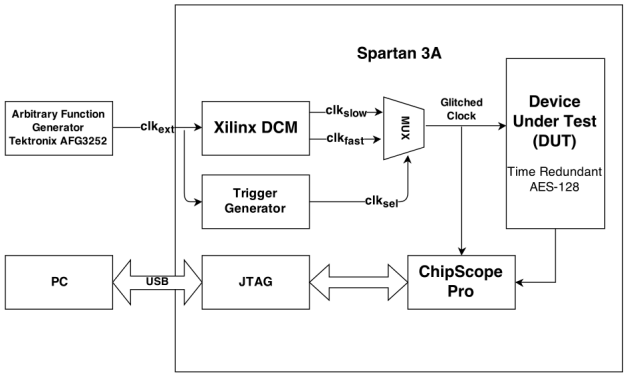


Fig. 2: Fault Injection Setup

function generator and can take frequency values up to 120 MHz. The clk_{slow} signal has the same frequency as clk_{ext} and is used for fault-free operation of the DUT. The clk_{fast} signal has a frequency equal to twice the frequency of clk_{ext} and is used to create the glitches for fault injection. The appropriate signal is fed to the DUT via a MUX. The select line of the MUX is the clk_{sel} signal which is output by the trigger generator and is set to high when clk_{fast} is to be fed to the DUT. The faulty states of the registers were monitored using Chipscope Pro 12.3 analyzer.

We injected faults in both the original and redundant rounds of time-redundant AES-128 by varying the clk_{ext} over a wide range of frequencies. Since the Chipscope pro 12.3 Analyzer limits the number of observable samples at a given frequency to 1024, we observed 512 samples for the original round and 512 samples for the redundant round. Tables 2a and 2b summarize the fault patterns obtained in either round. Table 3 summarizes the common frequency ranges between either round where each type of fault model is predominant.

4 Effectiveness of the Biased Fault Model

In this section, we demonstrate the effectiveness of the biased fault model in our attack. We quantify the biasness of a given fault model using the variance of the fault probability distribution. We assume that the set of faults that can occur under the fault model is given by $\mathcal{F} = \{f_1, \ldots, f_i, \ldots, f_n\}$, where n is the total number of faults possible under the fault model. Let F be a random variable that denotes the outcome of random occurrence of a single fault under this fault model. So the probability of occurrence of fault f_i is given by $p_i = Pr[F = f_i]$. Evidently, the fault model follows the probability distribution $\mathcal{P} = \{p_1, \ldots, p_i, \ldots, p_n\}$.

Table 2: Fault Distribution

(a) Fault Distribution Pattern - Original Round

Fast Clock Frequency (MHz)	FF	SBU	SBDBU	SBTBU	SBQBU	OSB	MB
125.0	512	0	0	0	0	0	0
125.1	503	9	0	0	0	0	0
125.2	489	22	1	0	0	0	0
125.3	456	50	6	0	0	0	0
125.4	425	59	22	6	0	0	0
125.5	396	45	43	28	0	0	0
125.6	354	34	112	32	0	0	0
125.7	303	23	101	85	0	0	0
125.8	260	11	55	86	0	0	0
125.9	208	5	46	147	6	0	0
126.0	176	1	39	228	68	0	0
126.1	143	0	18	211	136	4	0
126.2	115	0	10	94	178	15	0
126.3	101	0	8	95	251	49	8
126.4	65	0	9	45	232	141	20
126.5	32	0	5	16	131	187	141
126.6	13	0	3	8	98	101	289
126.7	5	0	1	4	32	112	358
126.8	0	0	1	2	5	105	399
126.9	0	0	1	2	3	88	421
127.0	0	0	0	1	2	33	476
127.1	0	0	0	0	1	12	499
127.2	0	0	0	0	0	0	512
127.3	0	0	0	0	0	0	512
127.4	0	0	0	0	0	0	512
127.5	0	0	0	0	0	0	512

(b) Fault Distribution Pattern - Redundant round

Fast Clock Frequency (MHz)	FF	SBU	SBDBU	SBTBU	SBQBU	OSB	MB
125.0	512	0	0	0	0	0	0
125.1	512	0	0	0	0	0	0
125.2	507	5	0	0	0	0	0
125.3	479	32	1	0	0	0	0
125.4	456	50	8	4	0	0	0
125.5	416	63	29	4	0	0	0
125.6	375	41	67	29	0	0	0
125.7	345	29	120	32	0	0	0
125.8	303	23	158	28	0	0	0
125.9	255	11	121	123	2	0	0
126.0	215	3	51	251	2	0	0
126.1	192	1	39	214	66	0	0
126.2	131	0	11	187	177	25	0
126.3	105	0	10	104	278	15	0
126.4	87	0	8	64	231	98	24
126.5	50	0	8	46	157	162	90
126.6	27	0	5	16	113	125	226
126.7	21	0	4	10	98	118	261
126.8	13	0	3	6	50	103	337
126.9	7	0	3	5	21	107	369
127.0	3	0	3	2	12	91	401
127.1	2	0	1	1	8	44	456
127.2	0	0	0	1	7	17	487
127.3	0	0	0	0	3	8	501
127.4	0	0	0	0	1	3	508
127.5	0	0	0	0	0	0	512

Table 3: Fault Models and Corresponding Frequency Ranges

Fault Model	Frequency Range (Original and Redundant Rounds) (MHz)
FF	< 125.3
SBU	125.3-125.4
SBDBU	125.6-125.7
SBTBU	126.0-126.1
SBQBU	126.3-126.4
OSB	126.5
MB	> 127.2

In order to get a faulty ciphertext in time redundant AES, the same fault f_i must occur in both the original and redundant rounds of computation. Let F_{org} and F_{red} be the random variables denoting the outcome of fault injections in the original and redundant rounds respectively. Since the fault injection in the original and redundant rounds are independent, we have $Pr[F_{org} = f_i, F_{red} = f_j] = p_i p_j$. We focus on the event where $F_{org} = F_{red}$. Let the probability of this event be denoted by $\tilde{p}$.

$$\tilde{p} = \sum_{i=1}^{n} Pr[F_{org} = f_i, F_{red} = f_i] = \sum_{i=1}^{n} {p_i}^2. \tag{1}$$

Evidently, this is also the probability of leakage of faulty ciphertexts. Our objective is to find if there is a correlation between the biased nature of the fault

Table 4: Notations Used

P	Plaintext
C	Fault-free ciphertext
f_i	A specific fault instance
n	The number of possible faults under the fault model
N_C	The total number of faulty ciphertexts obtained (excluding random ciphertexts generated by the countermeasure)
N_F	The total number of fault injections
C'_{f_i}	The faulty ciphertext under fault f_i
r	A round of AES
k	A key hypothesis
K	The correct key
$S^r{}_K$	The fault free cipher state in round r for key K
$S'^r{}_{k,f_i}$	A guess for the faulty cipher state before the SubBytes of round r under fault f_i and key hypothesis k

distribution and this probability of fault co-occurrence. Given the fault model $\mathcal{F}$ and the corresponding probability distribution $\mathcal{P}$, let Var denote that variance of $\mathcal{P}$. From the standard definition of variance of a probability distribution is given by $Var = \frac{\sum_{i=1}^{n} {p_i}^2}{n} - \frac{1}{n^2}$.

Note that the value of Var is 0 for a uniform fault distribution and increases with increase in non-uniformity. This justifies using the variance of the fault probability distribution as a measure for quantifying the biasness of the fault model. Finally, we have the following relation.

$$\tilde{p} = nVar + \frac{1}{n} \tag{2}$$

Thus, by using a biased fault model, one could greatly enhance the probability of occurrence of identical faults in consecutive rounds of computation in a time redundant circuit. The significance of this is as follows:

- If the countermeasure suppresses the ciphertext on fault detection, a biased fault model will warrant much fewer fault injections to get a faulty ciphertext.
- If the countermeasure produces a random ciphertext on fault detection that does not contribute to hypothesis testing, a biased fault model will require fewer ciphertexts and hence fewer fault injections.

In either scenario, the countermeasure is weakened.

5 Description of the Attack

In this section, we describe the detailed procedure of the performed attacks on a time redundant version of AES. The attack procedure introduces the fault into either round 8 or round 9 of AES, and exploits the biased nature of the introduced fault to decipher the key.

Please refer to Table 4 for the notations used for describing the attack procedure. Note that our fault model for the attack only comprises SBU, SBDBU, SBTBU and SBQBU (refer Table 1a), i.e, all the fault models are *single byte fault models*.

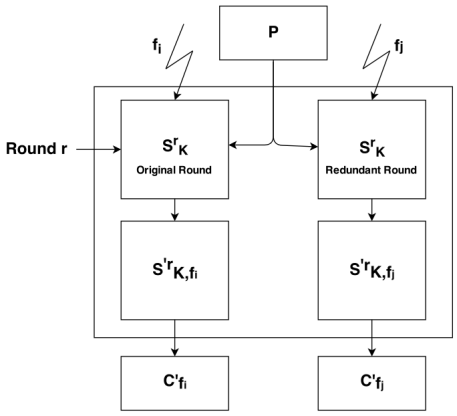


Fig. 3: Attack Steps

5.1 General Attack Procedure

We now present the general steps of the attack, irrespective of the round in which the fault is introduced. A more round-specific treatment of the attack is presented following the general discussion. Table 4 summarizes the notations used in describing the attack procedure. The steps are also elucidated in Figure 3.

Step 1: In this step the adversary induces faults f_i and f_j in both the normal and redundant computation of the target round r. However, the adversary can get the desired faulty ciphertext C'_{f_i} only if f_i and f_j are identical; otherwise the ciphertext is suppressed. *Note that alternatively, if the countermeasure produces random ciphertexts on fault detection instead of suppressing, the attack procedure does not change. The random ciphertext cannot distinguish between correct and incorrect key hypotheses and so, does not contribute to key hypothesis testing. This only increases the number of fault injections required to recover the key, as in the case of suppression.* For the purpose of a general treatment that encompasses both the scenarios, we consider N_C to be the *number of non-random faulty ciphertexts* and N_F to be the *overall number of fault injections.*

Step 2: Once the adversary collects the value of faulty ciphertext C'_{f_i}, he can compute the value of faulty state S'^r_{k,f_i} under key hypothesis k. He computes this value for every possible key hypothesis k.(Note that it is sufficient to hypothesize only those bytes of k that affect the faulty byte of S'^r_{k,f_i} since our fault model allows only single byte faults). After doing this for several collected ciphertexts, the adversary uses a distinguisher to identify the correct key hypothesis.

Step-3: The adversary chooses the key hypothesis k that minimizes/maximizes the appropriate distinguisher function for the chosen fault model. A detailed description of the distinguisher functions is presented in 5.2. If no satisfactory

key guess can be made, N_C is to be increased and the test repeated. Note that in time redundant AES with suppression, the number of fault injections N_F is greater than N_C as not all fault injections yield a faulty ciphertext.

5.2 Distinguisher Functions

Distinguisher functions are used by the adversary to decide on the correct key byte(s) by selecting the key hypothesis that corresponds to the expected bias in the faulty state. For our attacks, we use two well known distinguisher functions - *Hamming Distance* [4] and *Squared Euclidean Imbalance* [3, 13]. Equations 3 and 4 describe these functions, with k as the key hypothesis and b as the affected byte of the AES state.

$$H(k) = \sum_{i=1}^{N_C} \sum_{j=1}^{i-1} HD({S'^r}_{k,f_i}, {S'^r}_{k,f_j}) \tag{3}$$

$$S(k) = \sum_{\delta=1}^{255} (\frac{\#\{b \mid {S'^r}_{k,f_i}[b] = \delta\}}{N_C} - \frac{1}{256})^2 \tag{4}$$

5.3 The Attack on Time Redundant AES-128

We describe the fault attack procedure where the faults are introduced in rounds 8 and 9 of AES, and the choice of distinguisher function is made accordingly.

Attack on the 8th round

Fault Location: The fault f_i is injected just after the ante-penultimate AddRoundKey operation of the AES, modifying a random byte b of ${S^8}_K$ [3]. The injection occurs in both the original and redundant rounds of computation.

Attack Procedure: Equation 6 summarizes the relation between the faulty ciphertext and the faulty state. The adversary can hypothesize on 4 bytes of K_{10} and one byte of K_9 to get the corresponding states and then use the SEI distinguisher to identify the correct key hypothesis, because the Hamming Distance is found to require more faulty ciphertexts in this case to arrive at the key hypothesis.

Attack Complexity: The attack requires 2^{32} key hypotheses for recovering 4 bytes of the key [3], and a total of 4 such sets for recovering the entire key, leading to an overall requirement of $4 \times 2^{32} = 2^{34}$ hypotheses. Once again, time redundancy demands that the actual number of attacks be greater than the required number of faulty ciphertexts.

$${S'^9}_{K,f_i} = SB^{-1}(SR^{-1}({C'}_{f_i} \oplus K_{10})) \tag{5}$$

$${S'^8}_{K,f_i} = SB^{-1}(SR^{-1}((MC^{-1}((SB^{-1}(SR^{-1}({C'}_{f_i} \oplus K_{10})) \oplus K_9)))) \tag{6}$$

Attack on the 9th round

Fault Location: The fault f_i is injected just after the penultimate AddRoundKey operation of the AES, modifying a random byte b of ${S^9}_K$ [3]. The injection occurs in both the original and redundant rounds of computation.

Attack Procedure: Since the last round involves no MixColumns operation, we have Equation 7. The adversary collects several faulty ciphertexts ${C'}_1, \ldots,$ ${C'}_N$ on the same P and hypothesizes on one byte of the key to obtain 256 guesses of the faulty state ${{S'}^9}_{k,f_i}$ - one for each key hypothesis k. This is followed by the computation of $H(k)$ to identify the correct key hypothesis. It should be noted that the SEI distinguisher is useless in this context, as the distance to the uniform distribution will be the same for each hypothesis [3].

Attack Complexity: The attack requires 256 key hypotheses for recovering each byte of the key.

$$ {{S'}^9}_{K,f_i} = SB^{-1}(SR^{-1}({C'}_{f_i} \oplus K_{10})) \tag{7} $$

6 Simulated Results

In this section, we present results of simulations of attacks on AES-128 with time-redundancy countermeasure. The attack simulations were carried out on a software implementation of the time-redundant AES-128.

We divide the simulation into two major halves. In the first half, we assume the same fault for the original and redundant rounds so that each fault injection gives us a faulty ciphertext, i.e., N_C is same as N_F . Our aim here is to estimate the number of faulty ciphertexts required to recover the full key under different fault models. In the second half, we vary the probability distribution for each fault model to confirm the correlation of the bias with the number of fault injections required per faulty ciphertext, as described by Equation 2. Here, N_C is less than N_F as the suppressions are simulated.

6.1 Simulation: Part-1

In this part of the simulation, we assume identical faults in both the original and redundant computation rounds and aim to estimate the average number of faulty ciphertexts required to recover the entire key.

In the simulation, a byte of the state at the desired attack point is chosen at random and then fault is introduced into a certain number of bits belonging to that byte, varying from 1 to 4. Note that these bits are also chosen at random.We simulate the attacks in rounds 8 and 9 respectively. In each case, the appropriate distinguisher function is used to choose the key hypothesis. Table 5 summarizes the number of faulty ciphertexts required for each fault model to guess the entire 128-bit key with 99% accuracy for the attacks on rounds 8 and 9.

Table 5: Number Of Faulty Ciphertexts Required To Guess the Entire Key With 99% Probability

Round	Fault Model	N_C
8	SBU	320-340
	SBDBU	580-600
	SBTBU	1000-1040
	SBQBU	1900-2000
9	SBU	288-320
	SBDBU	608-640
	SBTBU	832-880
	SBQBU	1360-1440

6.2 Simulation: Part-2

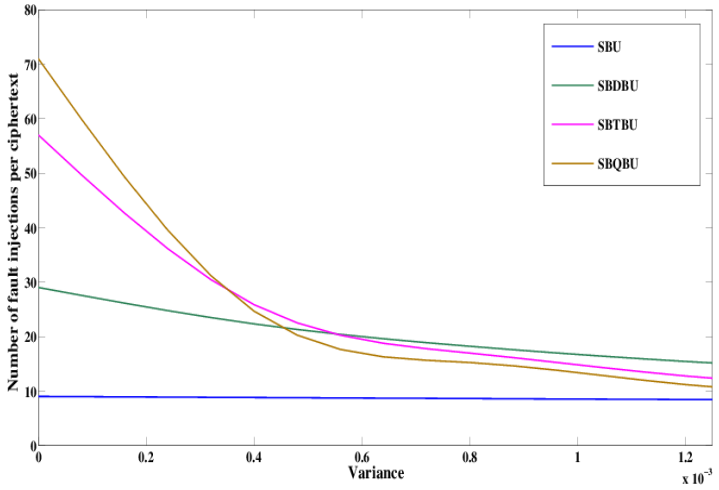
In the second half of the simulation, we varied the degree of bias for each fault model by controlling the variance of the fault probability distribution for each model and observed the average number of fault injections required per faulty ciphertext, computed over a set of 100 ciphertexts. In this experiment, the assumption was that the countermeasure suppresses the ciphertext on fault injection. Our experiment considered two distinct scenarios, in which the adversary has perfect and no control respectively over the target byte in which the fault is to be induced. For the first scenario, the fault was injected only in the fixed target byte, while in second scenario, the target byte was randomly chosen. In either scenario, we simulated the fault probability variance using a normal distribution with mean $1/n$ and the desired variance, where n is the total number of faults achievable under the corresponding fault model. Figures 4a and 4b summarize the simulation observations over a wide range of fault distribution variances, in both scenarios. These observations show that with increase in bias of the fault distribution, the number of fault injections that are required per faulty ciphertext drops rapidly. Thus, using a fault model with high variance indeed weakens the time redundancy countermeasure.

We also simulated another experiment with the same fault model, but with the assumption that the countermeasure produces a random ciphertext instead of suppressing it. Figure 4c shows, for the attack on round 9 where the adversary has perfect control over the target byte, how the required number of fault injections varies with the variance of the probability distribution. Clearly, even in this scenario, the total number of fault injections decreases with increase in the variance of the fault probability distribution.

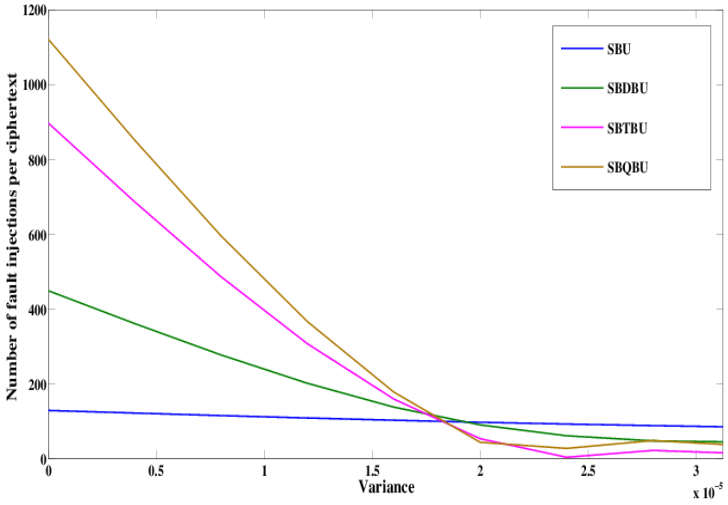
7 Experimental Results

In this experiment, we evaluate the proposed attack on a time-redundant hardware implementation of AES on Spartan-3A FPGA . The implementation is a register-transfer level Verilog definition of AES with each round duplicated by a re-computation round that helps achieve time redundancy. Thus a total of 20 rounds of computation are necessary. The plaintext and key are randomly

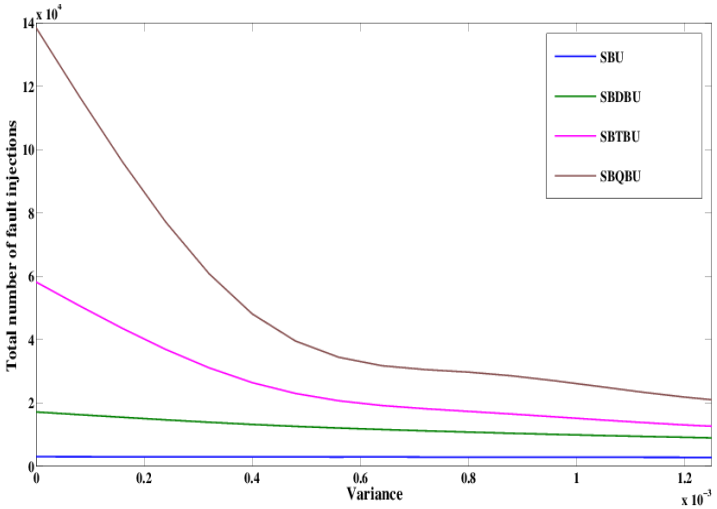
Fig. 4: Number of Fault Attacks per Faulty Ciphertext vs Variance of Fault Probability Distribution



(a) Adversary has perfect control over target byte



(b) Adversary has no control over target byte



(c) Countermeasure produces random ciphertexts

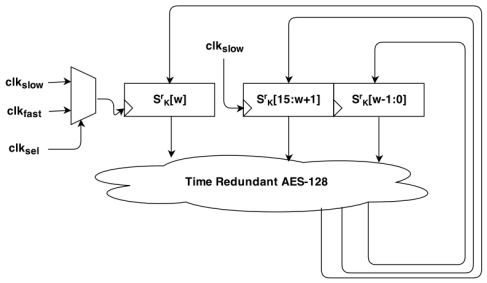


Fig. 5: Modified Fault Injection Setup: adversary has control over affected byte

chosen 128 bit values. If the output of original and redundant round of computations is different, i.e., if a fault is detected by the countermeasure, the output is immediately suppressed.

7.1 Experimental Procedure

Attack on Round-8: A total of 4 bytes of the AES state were affected one by one after the anti-penultimate AddRoundKey operation, since each byte of the faulty state can be guessed by hypothesizing 4 bytes of the Round 10 key K_{10}. Again, the external clk_{fast} was increased gradually from 125.3 MHz to 126.4 MHz to achieve the for different fault models. Once sufficient number of faulty ciphertexts had been collected for each of the 4 bytes, the entire key was deciphered using the appropriate Squared Euclidean Imbalance computation for each byte for all the key hypotheses.

Attack on Round-9: Each of the 16 bytes of the AES state were affected one by one after the penultimate AddRoundKey operation to guess the 16 bytes of the Round 10 key K_{10}. The external clk_{fast} was increased gradually from 125.3 MHz to 126.4 MHz to achieve the four different fault models. Once sufficient number of faulty ciphertexts had been collected for each byte, the entire key was deciphered using the appropriate Hamming Weight computation for each byte for all the key hypotheses.

7.2 Fault Location Precision

We performed 2 types of attacks - Type-1 in which the adversary has perfect control over the byte in which the fault is to be introduced and Type-2 in which the adversary only knows that the fault injected is a single byte fault without any knowledge of the byte affected. The second type of experiments demands much lesser control over the actual fault injection, but is weaker as observed in the experimental results, and demands a significantly larger number of fault injections. For the first type, only the target byte should be affected by the clock

glitch while in the second, the entire AES state should be subjected to the clock glitch. We describe the set up changes to be made for either scenario in greater detail. Suppose that the adversary wishes to affect only byte w of the AES state. She can achieve this precision by modifying the fault injection set up slightly to allow clk_{fast} to affect only byte w while all other bytes are driven by clk_{slow}. This ensures that in the event of a clock glitch, only byte w is affected. This is illustrated in Figure 5. Type-2 is the normal fault injection scenario where all bytes are allowed to be affected by clk_{fast}.

For each scenario, we repeated the experiment 100 times, with the same randomly chosen key and the randomly chosen plaintext and took the average values for the number of faulty ciphertexts as well as the number of fault injections required to recover the key as well. Table 6 demonstrates the number of faulty ciphertexts and the number of fault attacks required for recovering the entire key under the attack on rounds 8 and 9, for both the scenarios where the adversary has and does not have control over the fault location. The variance of fault distribution presented for each model was experimentally observed. In both tables, we compare the experimentally required number of fault injections with the expected number of fault injections according to the simulation. It is evident that the experimentally obtained data corroborates the simulation results very well, thus confirming the hypothesis that with more bias, our proposed fault attack can break the time redundancy countermeasure with very less number of fault injections, as compared to unbiased faults.

Table 6: Experimental Results

Round	Fault Model	Fault Variance		N_C	N_F(simulation)		N_F(experimental)	
		Type-1	Type-2		Type-1	Type-2	Type-1	Type-2
8	SBU	9.5×10^{-2}	3.6×10^{-3}	304.75	340.48	647.52	387.67	687.91
	SBDBU	1.4×10^{-2}	9.2×10^{-4}	625.12	1456.25	1506.25	1448.45	1652.30
	SBTBU	9.7×10^{-3}	4.9×10^{-4}	1020.49	1815.60	2315.40	1974.86	2395.83
	SBQBU	3.2×10^{-3}	5.9×10^{-5}	1878.55	7868.82	28038.54	8003.14	30201.41
9	SBU	9.2×10^{-2}	3.5×10^{-3}	304.24	385.88	603.11	387.98	632.71
	SBDBU	8.8×10^{-2}	7.9×10^{-4}	624.65	641.18	1487.36	647.82	1556.69
	SBTBU	8.1×10^{-2}	6.7×10^{-4}	832.32	873.56	2054.00	878.23	2489.25
	SBQBU	7.5×10^{-2}	3.5×10^{-5}	1328.22	1788.84	17239.10	1809.25	20145.66

8 Conclusions

This paper presents the first successful practical fault attack on the time redundancy countermeasure for AES-128 using biased fault models. The proposed attack requires neither precise fault injection techniques nor strong adversarial powers. The attack involves fault injection in either round 8 or round 9 of time redundant AES-128 using clock glitches. Our attack has been successfully demonstrated on simulated data as well as on 128-bit time redundant AES implemented on Xilinx Spartan-3A FPGA. The paper also develops a scheme to

show the effectiveness of a biased fault model in the analysis of the time redundancy countermeasure. We conclude that the usage of the countermeasures in secure systems based on uniform fault distribution should be reconsidered in the presence of biased fault models. Our future work is to apply our proposed attack to the hardware redundancy countermeasure.

References

1. Biham, E., Shamir, A.: Differential fault analysis of secret key cryptosystems. In: Advances in Cryptology-CRYPTO'97, pp. 513–525. Springer (1997)
2. Boneh, D., DeMillo, R.A., Lipton, R.J.: On the importance of checking cryptographic protocols for faults. In: Advances in Cryptology-EUROCRYPT'97. pp. 37–51. Springer (1997)
3. Fuhr, T., Jaulmes, E., Lomné, V., Thillard, A.: Fault attacks on aes with faulty ciphertexts only. In: Fault Diagnosis and Tolerance in Cryptography (FDTC), 2013 Workshop on. pp. 108–118. IEEE (2013)
4. Ghalaty, N.F., Yuce, B., Taha, M., Schaumont, P.: Differential fault intensity analysis
5. Hemme, L.: A differential fault attack against early rounds of (triple-) des. In: Cryptographic Hardware and Embedded Systems-CHES 2004, pp. 254–267. Springer (2004)
6. Kim, C.H.: Differential fault analysis against aes-192 and aes-256 with minimal faults. In: Fault Diagnosis and Tolerance in Cryptography (FDTC), 2010 Workshop on. pp. 3–9. IEEE (2010)
7. Kim, C.H.: Improved differential fault analysis on aes key schedule. Information Forensics and Security, IEEE Transactions on 7(1), 41–50 (2012)
8. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Advances in Cryptology-CRYPTO'99. pp. 388–397. Springer (1999)
9. Li, Y., Sakiyama, K., Gomisawa, S., Fukunaga, T., Takahashi, J., Ohta, K.: Fault sensitivity analysis. In: Cryptographic Hardware and Embedded Systems-CHES 2010, pp. 320–334. Springer (2010)
10. Maistri, P., Leveugle, R.: Double-data-rate computation as a countermeasure against fault analysis. IEEE Transactions on Computers 57(11), 1528–1539 (2008)
11. Malkin, T.G., Standaert, F.X., Yung, M.: A comparative cost/security analysis of fault attack countermeasures. In: Fault Diagnosis and Tolerance in Cryptography, pp. 159–172. Springer (2006)
12. Piret, G., Quisquater, J.J.: A differential fault attack technique against spn structures, with application to the aes and khazad. In: Cryptographic Hardware and Embedded Systems-CHES 2003, pp. 77–88. Springer (2003)
13. Rivain, M.: Differential fault analysis on des middle rounds. In: Cryptographic Hardware and Embedded Systems-CHES 2009, pp. 457–469. Springer (2009)
14. Tunstall, M., Mukhopadhyay, D., Ali, S.: Differential fault analysis of the advanced encryption standard using a single fault. In: Information Security Theory and Practice. Security and Privacy of Mobile Devices in Wireless Communication, pp. 224–233. Springer (2011)