# A Semi-Permanent Stuck-At Fault Analysis on AES Rijndael SBox

Priyanka Joshi
phd1801201001@iiti.ac.in
Indian Institute of Technology Indore
India

Bodhisatwa Mazumdar
bodhisatwa@iiti.ac.in
Indian Institute of Technology Indore
India

## Abstract

Fault attacks have gained particular attention in recent years as they present a severe threat to security in rapidly rising Internet-of-Things (IoT) devices. IoT devices are generally security-critical and resource-constrained. Therefore, any security protocol deployed in these devices has to satisfy several constraints such as small area footprint, low power, and memory consumption. Combinational circuit implementation of S-box is preferable over look-up table (LUT) in terms of memory consumption as the memory operations are usually the costliest part of lightweight cipher implementations. In this work, we analyze the S-box of AES against a novel fault analysis technique, Semi-Permanent Stuck-At (SPSA) fault analysis. We pinpoint hotspots in an optimized implementation of AES S-box that weaken the cryptographic properties of the S-box, leading to key recovery attacks. Our work investigates new vulnerabilities towards fault analysis in combinational circuit implementation.

***CCS Concepts:*** • **Security and Privacy → Security in hardware**; **Cryptography**; • **Computer systems organization** → *Embedded and cyber-physical systems*; Dependable and fault-tolerant systems and networks.

***Keywords:*** Fault attacks, AES, Lightweight ciphers, Combinational circuit, Semi-permanent, Stuck-at.

## 1 Introduction

Over the past two decades, fault analysis attacks have emerged as strong implementation-based attacks, especially, to embedded devices that run lightweight encryption and authentication operations. These devices, even though employing cryptographic primitives that exhibit provable security and are mathematically robust, are rendered weak to fault analysis attacks. The idea of fault attacks was first demonstrated on RSA cryptosystem in the seminal work of Boneh et al. [4]. Fault analysis involves malicious alteration of intermediate data/instruction in the normal execution of the cryptographic algorithm in embedded devices. This alteration, depending on spatio-temporal characteristics of the injected fault, and fault propagation characteristics of the algorithm implementation, leads to information leakage of the secret key embedded in the device through a set of faulty and fault-free ciphertexts.

The fault attacks on cryptographic primitives have been categorized in to many classes based on the attack principle, which comprise *differential fault analysis* [2], *fault sensitivity analysis* [12], *differential fault intensity analysis* (DFIA) [10], safe error attacks [3], and differential behavior analysis [17]. Unlike conventional fault analysis techniques, such as DFA and algebraic fault analysis (AFA) [6] that require both faulty and fault-free ciphertexts corresponding to the same set of input plaintexts, statistical fault analysis (SFA) [9] considers only faulty ciphertexts. SFA-based techniques gained popularity as these techniques are based on the ciphertext-only attack model, the weakest attack model.

In 2018, Dobraunig et al. [8] proposed another statistical approach called Statistical Ineffective Fault Analysis (SIFA) exploits information about the faults that do not corrupt the intermediate data, thus leaking information about the data values that do not participate in a specific operation during execution. Subsequently, Zhang et al. [18] proposed Persistent Fault Analysis (PFA), wherein a fault once injected persists in the system for a longer duration. PFA gained importance due to its effectiveness and relaxed fault model.

Each of the above-mentioned fault analysis techniques is based on a different fault model and attacker capabilities, and hence, requires varying attack efforts. In existing literature, majority of fault analysis techniques focus on fault injection in memory cells as well as registers [2, 3, 10, 12]. However, fault injection in the combinational circuits of cryptographic implementations and associated security vulnerabilities has not been investigated. It is imperative to investigate this aspect when compact and especially resource-constrained hardware implementations of crypto-primitives are increasingly adopted in IoT devices.

In this paper, we demonstrate a novel fault analysis technique, termed as *Semi-Permanent Stuck-at (SPSA)* fault analysis on combinational implementations of cryptographic primitives, such as Sboxes, in block ciphers. The contributions of this work are as follows,

- We investigate a *Semi-Permanent Stuck-at (SPSA)* fault analysis technique to evaluate the security of hardware implementations of cryptographic primitives against fault attacks.
- We pinpoint hotspots that result in crucial vulnerabilities in an optimized implementation of AES Sbox towards SPSA faults.

- We discuss sensitization model to quantify the amount of information an SPSA fault leaks when induced at a specified fault location.

## 2 Semi-permanent Stuck-at fault analysis

In this section, we describe the fault model, the threat model, and the attack procedure in the proposed semi-permanent stuck-at fault analysis (SPSA).

### 2.1 Fault Model

A fault model is a representation that denotes the alteration in the normal construction or normal operation of an electrical device or an algorithm. These representations enable predicting the consequences of a given fault. In general, fault models are used in the detection of manufacturing defects in integrated circuits (ICs). However, in fault attacks, defects are introduced maliciously in the device executing a cryptographic operation to obtain secret information. A fault model includes fault type (bit-flip/ stuck-at/ set-reset, etc.),required control on fault injection (precise/ relaxed), duration of effect of fault (transient/ permanent/ persistent/ intermittent/ semi-permanent), location of fault injection (memory cell/ register/ logic circuit), and assumptions about the cipher implementation (serial/ round rolled/ software/ hardware). In this section, we present a novel fault model termed as *Semi-Permanent-Stuck-At* (SPSA) fault model. The proposed SPSA fault model is based on two properties of injected faults: 1) A *Stuck-at 0 (1)* type of fault needs to be introduced in the specific locations in the Sbox circuit. 2) The fault should persist in the circuit for the desired duration, i.e., between fault injection and fault release, called *Semi-permanent* fault. As a semi-permanent fault is introduced with the help of two events this fault can be referred to as Double Event Transient (DET) [14].

Owing to the Total Ionizing Dose (TID) effect [16] in electronic devices, when *nano-focused X-ray* beam is injected at a specified location in the targeted circuit for a longer duration, it induces a semi-permanent fault. Once the fault is induced it remains in the circuit until a heat annealing treatment is performed to release the fault injected earlier. To induce a stuck-at-0(1) fault in the circuit at the input/output of a specified logic gate, the adversary makes a selected NMOS transistor conductive and PMOS transistor blocked [1]. The respective CMOS cell can be stuck at a logic value 0 or 1 based on the implemented functionality. Fluorescence mapping can be used to locate the targeted transistors, whereas the desired precision in beam triggering time can be achieved using a digital oscilloscope.

### 2.2 Threat Model

We assume that the adversary is aware of the implementation details of the Sbox in the targeted cipher and capable of introducing Semi-permanent stuck-at faults at desired locations in the Sbox circuit. After fault injection, the adversary waits for the faulty device to perform encryptions to collect sufficient number of faulty ciphertexts to be used in the analysis to recover the secret key.

### 2.3 Attack procedure

An adversary performs the following five steps to mount an attack on the targeted cipher implementation in the proposed SPSA fault analysis method.

**2.3.1 Identify hotspots :** In this phase, an adversary analyzes the fault propagation behavior of each fault point in the combinational circuit implementation of targeted Sbox against SPSA-0 and SPSA-1 faults, thereby locate the hotspots. Hotspots are fault locations that, when subjected to a particular fault, either alter the probability distribution of Sbox outputs from uniform to non-uniform or affect the targeted element's cryptographic properties, such as nonlinearity, algebraic degree, etc. This analysis can be done through simulations.

Consider a $2 \times 2$ toy Sbox shown in Figure 1, and its corresponding logic circuit as depicted in Figure 2. The input $X_0$ and $Y_0$ are the most significant input and output bits (MSBs) of the Sbox, respectively. The circuit has three intermediate wires, referred to as *fault points*, where an SPSA fault can be injected. Each fault point can take three possible values: semi-permanent stuck-at-0 denoted as SPSA-0, semi-permanent stuck-at-1 denoted as SPSA-1, and fault-free. A fault point is fault-free if a fault is not injected at that point. In Figure 2, $f_1$,

| $x$ | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| $S(x)$ | 3 | 0 | 1 | 2 |

**Figure 1.** Example Sbox S

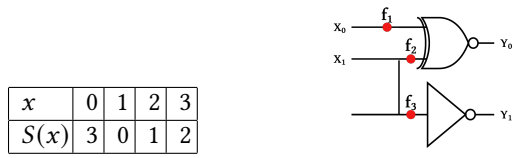


**Figure 2.** Combinational circuit

$f_2$, and $f_3$, represent fault-points that can acquire any of the three possible values, thus, resulting in $3^3 = 27$ fault combinations including a non-faulty output or correct output. Figure 4 and Figure 6 illustrate effects of two different fault combinations: SPSA-0 at $f_1$ and SPSA-1 at $f_2$, respectively. The faulty Sboxes, $S_1(x)$ and $S_2(x)$ corresponding each fault combination are shown in Figure 4 and Figure 6. Entries in the red color text in faulty Sboxes indicate faulty values generated due to the presence of respective SPSA faults. $S_1$ is a biased mapping exhibiting a non-uniform distribution of output values that an attacker can exploit in statistical analysis to retrieve the secret key. Whereas $S_2$ is a *linear bijection* resulting in an effortless key recovery attack.

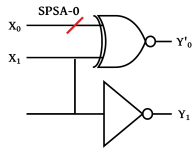


**Figure 3.** Fault combination-I

| $x$ | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| $S_1(x)$ | 3 | 0 | 3 | 0 |

**Figure 4.** Faulty Sbox $S_1$

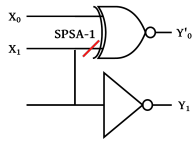


**Figure 5.** Fault combination-II

| $x$ | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| $S_2(x)$ | 1 | 0 | 3 | 2 |

**Figure 6.** Faulty Sbox $S_2$

**2.3.2 Select most promising fault-points :** An adversary chooses the optimal fault-combination for fault injection among the identified vulnerable hotspots in step (i). The choice of fault-combination depends on several factors such as attackers capability, analysis method for key recovery, residual key search space after analysis, etc.

**2.3.3 Inject desired SPSA fault(s):** An adversary injects chosen SPSA faults in the S-box circuit. Due to the semi-permanent characteristic the SPSA can be injected much before the encryption starts and thus does not need precise time synchronization.

**2.3.4 Data Collection :** Adversary collects faulty ciphertexts to be used in analysis for key recovery. Once the sufficient number of faulty ciphertexts are obtained, the adversary releases the fault by heat annealing.

**2.3.5 Key recovery from collected ciphertexts :** In this phase, attacker recovers the secret key from collected faulty ciphertexts of unknown plaintexts or chosen plaintexts depending on the method of analysis. In order to recover the secret key, the proposed SPSA fault model can be coupled with multiple cryptanalysis techniques depending on the most vulnerable hotspot in the circuit. For example, 1) An SPSA fault at some fault-point may result in non-uniform S-box mapping and hence a good candidate for statistical analysis techniques such as elimination of impossible key candidates based on non-uniform distribution of S-box output values, Square Euclidean Imbalance (SEI) distinguisher, etc. 2) A different SPSA fault at a different fault-point may negatively affect cryptographic properties of the S-box such as *nonlinearity*, *differential uniformity*, or *correlation immunity*, thereby resulting in weakened resilience against *linear cryptanalysis*, *differential cryptanalysis*, or *correlation or subset attacks*, respectively.

# 3 Analyzing AES S-box with SPSA faults

In this section, we demonstrate the proposed SPSA fault analysis on AES Rijndael Sbox.

## 3.1 AES

In October 2000, Rijndael [7] cipher was chosen as the Advanced Encryption Standard, commonly known as AES. AES is a substitution permutation (SPN) based cipher that runs in $N$ rounds where $N$ is 10 for AES-128 and 14 for AES-256. The input data block of 128 bits is considered as a $4 \times 4$ matrix of bytes, called state matrix. The state matrix is updated through $N$ rounds. Each round except the last performs four operations, namely *SubBytes*, *ShiftRows*, *MixColumns*, and *AddRoundKey*. A key schedule is used to generate $N + 1$ round keys to be used in the AddRoundKey operation. The round operations are described briefly:

(1) SubBytes (SB) is the substitution step where an $8 \times 8$ non-linear function called Sbox is applied on each byte in the state matrix.
(2) ShiftRows (SR) performs a cyclic rotation on each row of the state.
(3) MixColumns (MC) linearly combines the elements in each column using a constant matrix. The last round does not execute this operation.
(4) AddRoundKey operation performs an XOR between the intermediate state matrix and round key.

We present the analysis on combinational circuit implementation of an optimized S-box of AES towards the proposed SPSA fault vulnerability. AES uses a 8-bit bijective Sbox, which has a large fault space. Several optimized implementations of AES Sbox have been proposed in the literature [5, 11, 13, 15]. The implementation proposed by Boyar et al. in [11] is optimal in *gate count* and *circuit depth* for both reverse and forward directions. Hence, we choose this implementation for our analysis. The chosen implementation requires 125 gates, and its circuit depth in both forward and reverse directions is 16.

## 3.2 SPSA Analysis and Observations on AES Sbox

Boyar et al. [11] presented an optimized implementation of AES Sbox as shown in Fig. 7. We evaluated the same implementation in our analysis. The circuit has 125 gates, and there are 250 fault-points. For analysis, we consider only 125 fault points, i.e., each gate's output is a fault-point. If a faulty output is fed as input to multiple gates, then all those inputs become faulty. If multiple fault-points can be faulty at the same time, then there is a total ${}^{125}C_1 + {}^{125}C_2 + \ldots {}^{125}C_{125} = 2^{125} - 1$ fault combinations. Since, each fault-point can be either *stuck-at-0* or *stuck-at-1*, this results in $(3^{125} - 1)$ possible fault combinations. This is a large fault space which is infeasible for an exhaustive test. Therefore, for analysis, we choose fault combinations that contain atmost two fault-points, i.e., $(2 \times {}^{125}C_1) + (2^2 \times {}^{125}C_2) = 31250$

```
T1 = U6 + U4
T2 = U3 + U0
T3 = U1 + U2
T4 = U7 + T3
T5 = T1 + T2
T6 = U1 + U5
T7 = U0 + U6
T8 = T1 + T6
T9 = U6 + T4
T10 = U3 + T4
T11 = U7 + T5
T12 = T5 + T6
T13 = U2 + U5
T14 = T3 + T5
T15 = U5 + T7
T16 = U0 + U5
T17 = U7 + T8
T18 = U6 + U5
T19 = T2 +T18

T20 = T4 + T15
T21 = T1 + T13
T22 = U0 + T4
T39 = T21 + T5
T40 = T21 + T7
T41 = T7 + T19
T42 = T16 + T14
T43 = T22 + T17
T44 = T19 x T5
T45 = T20 x T11
T46 = T12 + T44
T47 = T10 x U7
T48 = T47 + T44
T49 = T7 x T21
T50 = T9 x T4
T51 = T40 + T49
T52 = T22 x T17
T53 = T52 + T49
T54 = T2 x T8

T55 = T41 x T39
T56 = T55 + T54
T57 = T16 x T14
T58 = T57 + T54
T59 = T46 + T45
T60 = T48 + T42
T61 = T51 + T50
T62 = T53 + T58
T63 = T59 + T56
T64 = T60 + T58
T65 = T61 + T56
T66 = T62 + T43
T67 = T65 + T66
T68 = T65 x T63
T69 = T64 + T68
T70 = T63 + T64
T71 = T66 + T68
T72 = T71 x T70
T73 = T69 x T67

T74 = T63 x T66
T75 = T70 x T74
T76 = T70 + T68
T77 = T64 x T65
T78 = T67 x T77
T79 = T67 + T68
T80 = T64 + T72
T81 = T75 + T76
T82 = T66 + T73
T83 = T78 + T79
T84 = T81 + T83
T85 = T80 + T82
T86 = T80 + T81
T87 = T82 + T83
T88 = T85 + T84
T89 = T87 x T5
T90 = T83 x T11
T91 = T82 x U7
T92 = T86 x T21

T93 = T81 x T4
T94 = T80 x T17
T95 = T85 x T8
T96 = T88 x T39
T97 = T84 x T14
T98 = T87 x T19
T99 = T83 x T20
T100 = T82 x T10
T101 = T86 x T7
T102 = T81 x T9
T103 = T80 x T22
T104 = T85 x T2
T105 = T88 x T41
T106 = T84 x T16
T107 = T104 + T105
T108 = T93 + T99
T109 = T96 + T107
T110 = T98 + T108
T111 = T91 + T101

T112 = T89 + T92
T113 = T107 + T112
T114 = T90 + T110
T115 = T89 + T95
T116 = T94 + T102
T117 = T97 + T103
T118 = T91 + T114
T119 = T111 + T117
T120 = T100 + T108
T121 = T92 + T95
T122 = T110 + T121
T123 = T106 + T119
T124 = T104 + T115
T125 = T111 + T116
S0 = T109 + T122
S2 = T123 # T124
T128 = T94 + T107
S3 = T113 + T114
S4 = T118 + T128

T131 = T93 + T101
T132 = T112 + T120
S7 = T113 # T125
T134 = T97 + T116
T135 = T131 + T134
T136 = T93 + T115
S6 = T109 # T135
T138 = T119 + T132
S5 = T109 + T138
T140 = T114 + T136
S1 = T109 # T140
```

**Figure 7.** Optimized AES Sbox circuit [11]. Inputs to the Sbox are denoted as $U0, \ldots, U7$ and outputs are denoted as $S0, \ldots, S7$. Symbols +, ×, and # represent *XOR*, *AND*, and *XNOR* operations, respectively.

fault combinations. We observe that the residual key-space for last round key of AES can be reduced by upto $34^{16} \approx 2^{81}$. However, we could not test the entire fault-space of $3^{125} - 1$ fault combinations in AES. As a result, there may exist an optimal set of fault combinations that may further reduce the residual key-space. We consider optimizing this residual key space as a future scope of this work.

The observations from our experiments show that majority of fault-points when subjected to SPSA faults, severely disrupt the distribution of Sbox output values, i.e., the faulty Sbox outputs are non-uniformly distributed. For instance, when an SPSA-0 fault is injected at fault-point $T1$, 128 elements among 256 possible output elements occur with *zero* probability, i.e., absent from S-box output. These absent elements at faulty Sbox output affect the *balancedness* property of the Sbox. Therefore, based on this observation, we define *Imbalance Factor* as follows.

$$N_{ae} = \{Count(y) | y \in S(x) \wedge y \notin S^*(x)\} \tag{1}$$

$$Imbalance\ Factor = \frac{N_{ae}}{2^n} \tag{2}$$

where $S(x)$, $S^*(x)$ represent original and faulty $n$-bit Sboxes, respectively and $N_{ae}$ denote number of non-occurring elements at the output of faulty Sbox $S^*(x)$. Hence, if in a $8 \times 8$ faulty Sbox $S^*(x)$, 50 elements among 256 possible elements do not occur at the output then *Imbalance Factor* of $S^*(x)$ is 0.195. The value of Imbalance Factor of an Sbox $S$ can be between 0 and 1 i.e. $0 \leqslant Imbalance\ Factor(S) \leqslant 1$. A *Zero* value of the Imbalance Factor indicates that the Sbox is balanced. In contrast, a higher value of the Imbalance Factor indicates an extremely imbalanced Sbox, which is vulnerable to several attacks such as Subset cryptanalysis, statistical distinguishers, etc.

Table 1 presents the cryptographic properties of faulty Sboxes corresponding to specific SPSA fault/s at chosen fault-point/s in the AES Sbox implementation provided in Fig. 7. It presents possible attack threats due to the weaken cryptographic properties. First row of the table shows properties of original non-faulty Sbox.

## 4 Sensitization Model in SPSA Fault Analysis

In this section, we model the sensitization of a fault injected at a node to the output of Sbox. We consider that an injected SPSA-0 or an SPSA-1 fault when sensitized and propagates the fault to the output, it eases the attacker's effort to recover the key. An Sbox $F : \{0,1\}^n \mapsto \{0,1\}^n$ comprises of $n$ coordinate functions defined as, $F(x) : (f_{n-1}(x), f_{n-2}(x), \ldots, f_0(x))$ for all $x \in \{0,1\}^n$. From this definition, $f_i$, $0 \leq i \leq n-1$, is called a *coordinate function* of $F$. Suppose under SPSA fault model, Sbox $F(x) : (f_{n-1}(x), \ldots, f_0(x))$ is changed to $F^*(x) : (f^*_{n-1}(x), \ldots, f^*_0(x))$.

For an input, $x \in \{0,1\}^n$, we say that an SPSA-0 (SPSA-1) fault gets sensitized to the output of faulty Sbox $F^*$ if there exists at least a pair of coordinate functions, $(f_i, f_i^*)$, $0 \leq i \leq n-1$, such that for input $x \in \{0,1\}^n$, $f_i(x) \oplus f_i^*(x) = 1$. If, for input $x$, $f_i(x) \oplus f_i^*(x) = 0$, we consider that the respective fault does not get sensitized to the output of the coordinate function. The sensitization of an SPSA fault to the output of a coordinate function $f_i$, over the entire input domain, $x \in \{0,1\}^n$ is captured by the correlation function, $C_{f_i,f_i^*}$, between the fault-free coordinate function and its respective faulty coordinate function as,

$$C_{f_i,f_i^*} = \sum_{x \in \{0,1\}^n} (-1)^{f_i(x) \oplus f_i^*(x)} \tag{3}$$

The correlation function, $C_{f_i,f_i^*}$, is the difference of the number of input values $x$ for which $f_i(x) \oplus f_i^*(x) = 0$ and $f_i(x) \oplus f_i^*(x) = 1$. The sensitization of an SPSA-0 or SPSA-1 fault corresponding to the output of the coordinate function

| *Fault point* | *Fault type* | *Imbalance Factor* | *Nonlinearity* | *Differential Uniformity* | *Vulnerabilities* |
|---|---|---|---|---|---|
| No-fault | No-fault | 0 | 112 | 4 | – |
| $\{T1\}$ | {SPSA-0} | 0.511 | 96 | 8 | Statistical distinguishers |
| $\{T1\}$ | {SPSA-1} | 0.511 | 98 | 4 | Statistical distinguishers |
| $\{T12\}$ | {SPSA-0} | 0.278 | 92 | 12 | Linear cryptanalysis |
| $\{T12\}$ | {SPSA-1} | 0.391 | 96 | 8 | Statistical distinguishers Linear cryptanalysis |
| $\{T62\}$ | {SPSA-0} | 0.254 | 92 | 14 | Linear cryptanalysis |
| $\{T62\}$ | {SPSA-1} | 0.391 | 92 | 10 | Statistical distinguishers Linear cryptanalysis |
| $\{T75\}$ | {SPSA-0} | 0.133 | 100 | 10 | PFA[18] like analysis |
| $\{T75\}$ | {SPSA-1} | 0.465 | 100 | 10 | Statistical distinguishers |
| $\{T78\}$ | {SPSA-0} | 0.133 | 100 | 10 | PFA[18] like analysis |
| $\{T78\}$ | {SPSA-1} | 0.465 | 100 | 10 | Statistical distinguishers |
| $\{T83\}$ | {SPSA-0} | 0.332 | 96 | 12 | Statistical distinguishers Linear cryptanalysis |
| $\{T83\}$ | {SPSA-1} | 0.332 | 96 | 12 | Statistical distinguishers Linear cryptanalysis |
| $\{T128\}$ | {SPSA-0} | 0.281 | 86 | 6 | Linear cryptanalysis |
| $\{T128\}$ | {SPSA-1} | 0.281 | 86 | 6 | Linear cryptanalysis |
| $\{T1, T2\}$ | {SPSA-0} | 0.641 | 76 | 80 | Vulnerable to most of the cryptanalytic attacks |
| $\{T1, T2\}$ | {SPSA-1} | 0.636 | 76 | 64 | Vulnerable to most of the cryptanalytic attacks |
| $\{T1, T49\}$ | {SPSA-0} | 0.651 | 80 | 16 | Linear, Algebraic attacks Statistical distinguishers |
| $\{T1, T50\}$ | {SPSA-1} | 0.644 | 82 | 16 | Linear, Algebraic attacks Statistical distinguishers |
| $\{T1, T51\}$ | {SPSA-0, SPSA-1} | 0.681 | 82 | 16 | Linear, Algebraic attacks Statistical distinguishers |

**Table 1.** Hotspots for proposed *SPSA* analysis in AES Sbox optimized implementation as shown in Figure 7. Node outputs $\{T_1, T_2, \cdots, T_{140}\}$ are considered as fault-points.

$f_i$ is hence defined as,

$$S_{SPSA}^{f_i,f_i^*} = \frac{1}{2} - \frac{C_{f_i,f_i^*}}{2^{n+1}} = \frac{1}{2} - \frac{\sum_{x\in\{0,1\}^n} (-1)^{f_i(x)\oplus f_i^*(x)}}{2^{n+1}} \tag{4}$$

If an SPSA-0 or SPSA-1 fault injected at any fault location in Sbox do not propagate to the output of a coordinate function $f_i$ of the Sbox $F$, the corresponding sensitization, $S_{SPSA}^{f_i,f_i^*} = 0$ implies that the SPSA fault does not propagate to the output of $f_i$. In other words, the SPSA fault does not leak information in the output ciphertext pattern as there are no faulty ciphertexts. However, if SPSA-0 or SPSA-1 fault propagates to the output of coordinate function $f_i$ of Sbox $F$ or all input patterns, $x \in \{0,1\}^n$, then $S_{SPSA}^{f_i,f_i^*} = 1$. In other words, for coordinate function $f_i$, its corresponding sensitization, $S_{SPSA}^{f_i,f_i^*}$ defines the vulnerability or the amount of information that a fault location leaks to the output ciphertext pattern.

The sensitization of an SPSA-0 or SPSA-1 fault when injected at a fault location in an Sbox can affect one or multiple coordinate functions of an Sbox. As a result, we define the sensitization of an SPSA-0 or SPSA-1 fault to the output of Sbox $F$ in terms of its coordinate functions, $f_i, 0 \le i \le n-1$, as a tuple, $S_{SPSA}^F = (S_{SPSA}^{f_{n-1},f_{(n-1)}^*}, \ldots, S_{SPSA}^{f_0,f_0^*})$. If an SPSA-0 (SPSA-1) fault does not leak any fault information to the output of $F$, then for all $i$, $0 \le i \le n-1$, $S_{SPSA}^{f_i,f_i^*} = 0$. If there exists an $i$, $0 \le i \le n-1$, for which $S_{SPSA}^{f_i,f_i^*} \neq 0$, the corresponding SPSA-0 (SPSA-1) leaks fault information to the output $F$, and hence may be exploitable by the fault analysis adversary. We demonstrate the sensitization of SPSA-0 and SPSA-1 faults at some fault points for the coordinate functions in AES Sbox implementation in Figure 8 and Figure 9. Figure 8 correspond to fault points correspond to $T_{75}$ and $T_{78}$, and Figure 9 correspond to fault points $T_1$, $T_{12}$, and $T_{83}$, respectively. In Figure 9, as larger number of Sbox output values are not occurring, the sensitization values at these fault points for all corresponding Sboxes are significantly larger than that of Figure 8. The sensitization values thus depict the strength of fault points in leaking the fault information to the Sbox values, and hence to the ciphertext values as well.

## 5 Conclusion

In this work, we propose semi-permanent stuck-at (SPSA) fault analysis on the Sboxes of SPN block ciphers, which results in key recovery from the collected faulty ciphertexts. We propose parameters, imbalance factor and sensitization of the coordinate functions of AES Sbox, that capture the SPSA vulnerability of the different fault points in the Sbox architecture. We investigated vulnerable hotspots on an optimized implementation of AES Sbox. We tested 31250 fault combinations and observed several instances of single-point SPSA faults that, when injected, restrict between 34 to 168 elements to occur at the output of substitution layer among 256 possible values. We evaluated essential cryptographic properties of these faulty Sboxes and observed that several single-point SPSA faults when injected can leave the cipher

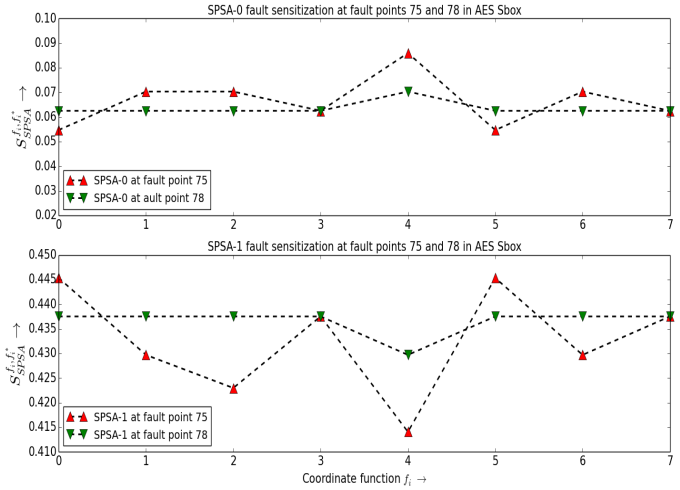


**Figure 8.** SPSA-0 and SPSA-1 fault sensitization values for all coordinate functions of AES Sbox for fault points $T_{75}$ and $T_{78}$.

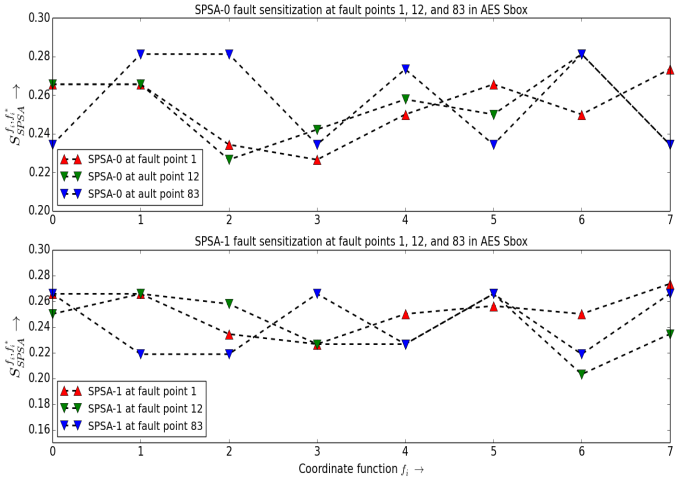


**Figure 9.** SPSA-0 and SPSA-1 fault sensitization values for all coordinate functions of AES Sbox for fault points $T_1$, $T_{12}$ and $T_{83}$.

implementation to be vulnerable to multiple cryptanalytic attacks. We believe that our work will contribute to a more comprehensive understanding of the security vulnerabilities in the light of semi-permanent stuck-at faults.

## References


[1] Stéphanie Anceau, Pierre Bleuet, Jessy Clédière, Laurent Maingault, Jean-Luc Rainard, and Rémi Tucoulou. 2017. Nanofocused X-Ray Beam to Reprogram Secure Circuits. In *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings (Lecture Notes in Computer Science, Vol. 10529)*, Wieland Fischer and Naofumi Homma (Eds.). Springer, 175–188.

[2] Eli Biham and Adi Shamir. 1997. Differential Fault Analysis of Secret Key Cryptosystems. In *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings (Lecture Notes in Computer Science, Vol. 1294)*, Burton S. Kaliski Jr. (Ed.). Springer, 513–525.

[3] Johannes Blömer and Jean-Pierre Seifert. 2003. Fault based cryptanalysis of the advanced encryption standard (AES). In *International Conference on Financial Cryptography*. Springer, 162–181.

[4] Dan Boneh, Richard A DeMillo, and Richard J Lipton. 1997. On the importance of checking cryptographic protocols for faults. In *International conference on the theory and applications of cryptographic techniques*. Springer, 37–51.

[5] D. Canright. 2005. A Very Compact S-Box for AES. In *Proceedings of the 7th International Conference on Cryptographic Hardware and Embedded Systems* (Edinburgh, UK) *(CHES'05)*. Springer-Verlag, Berlin, Heidelberg, 441–455.

[6] Nicolas Courtois, David Ware, and Keith Jackson. 2010. Fault-Algebraic Attacks on Inner Rounds of DES. *Proceedings of the eSmart 2010, pp. 22–24, 2010* (01 2010).

[7] Joan Daemen and Vincent Rijmen. 1998. The Block Cipher Rijndael. In *Smart Card Research and Applications, This International Conference, CARDIS '98, Louvain-la-Neuve, Belgium, September 14-16, 1998, Proceedings (Lecture Notes in Computer Science, Vol. 1820)*, Jean-Jacques Quisquater and Bruce Schneier (Eds.). Springer, 277–284.

[8] Christoph Dobraunig, Maria Eichlseder, Hannes Groß, Stefan Mangard, Florian Mendel, and Robert Primas. 2018. Statistical Ineffective Fault Attacks on Masked AES with Fault Countermeasures. In *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 11273)*. Springer, 315–342.

[9] Thomas Fuhr, Éliane Jaulmes, Victor Lomné, and Adrian Thillard. 2013. Fault Attacks on AES with Faulty Ciphertexts Only. In *2013 Workshop on Fault Diagnosis and Tolerance in Cryptography, Los Alamitos, CA, USA, August 20, 2013*. 108–118.

[10] Nahid Farhady Ghalaty, Bilgiday Yuce, Mostafa Taha, and Patrick Schaumont. 2014. Differential fault intensity analysis. In *2014 Workshop on Fault Diagnosis and Tolerance in Cryptography*. IEEE, 49–58.

[11] J.Boyar and René Peralta. 2012. A Small Depth-16 Circuit for the AES S-Box. In *Information Security and Privacy Research - 27th IFIP TC 11 Information Security and Privacy Conference, SEC 2012, Heraklion, Crete, Greece, June 4-6, 2012. (IFIP Advances in Information and Communication Technology, Vol. 376)*. Springer, 287–298.

[12] Yang Li, Kazuo Sakiyama, Shigeto Gomisawa, Toshinori Fukunaga, Junko Takahashi, and Kazuo Ohta. 2010. Fault sensitivity analysis. In *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 320–334.

[13] Sumio Morioka and Akashi Satoh. 2002. An Optimized S-Box Circuit Architecture for Low Power AES Design. In *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers (Lecture Notes in Computer Science, Vol. 2523)*, Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar (Eds.). Springer, 172–186.

[14] Debdeep Mukhopadhyay. 2020. Faultless to a Fault? The Case of Threshold Implementations of Crypto-systems vs Fault Template Attacks. In *IEEE/ACM International Conference On Computer Aided Design, ICCAD 2020, San Diego, CA, USA, November 2-5, 2020*. IEEE, 66:1–66:9.

[15] Yasuyuki Nogami, Kenta Nekado, Tetsumi Toyota, Naoto Hongo, and Yoshitaka Morikawa. 2011. Mixed Bases for Efficient Inversion in $F((2^2)^2)^2$ and Conversion Matrices of SubBytes of AES. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* 94-A, 6 (2011), 1318–1327.

[16] T.R. Oldham and F.B. McLean. 2003. Total ionizing dose effects in MOS oxides and devices. *IEEE Transactions on Nuclear Science* 50, 3 (2003), 483–499.

[17] Bruno Robisson and Pascal Manet. 2007. Differential behavioral analysis. In *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 413–426.

[18] Fan Zhang, Xiaoxuan Lou, Xinjie Zhao, Shivam Bhasin, Wei He, Ruyi Ding, Samiya Qureshi, and Kui Ren. 2018. Persistent Fault Analysis on Block Ciphers. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2018, 3 (2018), 150–172.