Journal of Cryptographic Engineering (2021) 11:273–288
https://doi.org/10.1007/s13389-020-00249-0

REGULAR PAPER

Fault intensity map analysis with neural network key distinguisher

Keyvan Ramezanpour[1] · Paul Ampadu[1] · William Diehl[1]

Received: 14 April 2020 / Accepted: 7 November 2020 / Published online: 24 November 2020

Abstract
Physical cryptographic implementations are vulnerable to side-channel attacks, including fault attacks, which can be used to recover a secret key. Using a deep neural network (NN) with fault intensity map analysis (FIMA), we present a new highly efficient statistical fault injection analysis (FIA) technique called FIMA-NN. This technique employs a convolutional neural network to rank the key candidates based on multiple features in data distribution under fault with varying intensities and generalizes most existing statistical techniques including fault sensitivity analysis, differential fault intensity analysis, statistical ineffective fault analysis, and FIMA. As FIMA-NN does not rely on a single feature of data distribution, it is successful even in the presence of a wide variety of countermeasures against FIA. We introduce a generic statistical model for timing failure attacks using dynamic timing analysis of an AES S-box implemented in TSMC 65 nm technology with standard ASIC design flow. Using the simulated fault mechanism, we demonstrate that, in terms of required amount of collected ciphertexts, FIMA-NN is 12.6 times more efficient than statistical techniques using bias alone, when faulty and fault-free values are not filtered. Further, in the presence of error detection and infective countermeasures, FIMA-NN is 4.8 and 5 times more efficient than bias-alone techniques, respectively.

Keywords Convolutional neural network (CNN) · Dynamic timing analysis · Fault image · Fault injection analysis (FIA) · FIMA · AES

1 Introduction

Cryptography is an important component of robust end-to-end cybersecurity. Standardized cryptographic algorithms are generally secure against cryptanalysis or brute force attacks, but are subject to implementation vulnerabilities resulting from physical manifestation in hardware or software, called side-channel analysis (SCA).

Fault injection analysis (FIA) has emerged as a powerful active SCA technique used to compromise the security of many ciphers implemented in hardware or software [7,23,30,37]. Fault injection mechanisms may induce various properties in faulty or even fault-free data that can be exploited to recover a secret key. FIA techniques can be divided into the two broad categories of differential fault analysis (DFA) and statistical fault injection analysis (SFIA).

In differential fault analysis, fault-free and faulty outputs of the cipher for the same plaintext and initial state are used to calculate the error in an intermediate variable. Certain properties of the error are exploited to identify the correct key among all key candidates [19,21]. While DFA recovers the secret key with a small number of fault injections, as, for example, in [24], it assumes a strong fault model; fault manifestation must be precise.

Statistical fault injection analysis techniques relax the assumptions of fault model; however, more fault injections are required to recover the secret key. In SFIA, distinct statistical properties of data distribution under fault induction are exploited to recover the key. As a result, running multiple encryptions with the same input and initial conditions, necessary to calculate the error, might not be required. Further, noisy fault inductions, in which timing or location of the fault is not precise, can be tolerated. In particular, the authors in [26] introduce fault intensity map analysis (FIMA), which combines observables of fault bias, and the variation in fault distribution with fault intensity, to recover a secret key, with

✉ Keyvan Ramezanpour
rkeyvan8@vt.edu

Paul Ampadu
ampadu@vt.edu

William Diehl
wdiehl@vt.edu

[1] The Bradley Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA 24061, USA

imprecise fault models or in the presence of countermeasures against fault attack.

Deep learning techniques are being increasingly used to improve and evaluate information security. Applications such as intrusion detection systems (IDS) [11], anomaly detection [16], malware [14], and hardware Trojan (HT) detection [29] constitute a wide range of research exploiting the power of deep learning. Deep learning is also powerful in identifying features of data distribution in side-channel analysis. An example is [28] that employs neural networks for a non-profiled attack on a FPGA implementation of AES.

FIMA is augmented with a deep learning technique in [25], introduced as *fault intensity map analysis with neural network key distinguisher* (FIMA-NN), which is used to recover the secret key of the Advanced Encryption Standard (AES). While FIMA-NN demonstrated significant improvement in the efficiency of a fault attack, the success of a deep learning FIA (DL-FIA) technique relies on employing a proper training set. In this paper, we introduce a generic methodology for simulating timing failure attacks on hardware implementation of ciphers using standard electronic design automation (EDA) tools. The results of simulations are used to develop a generic statistical model for fault distribution under timing failure attacks which is employed to generate the proper training set for the neural network key distinguisher. We demonstrate that *dynamic timing analysis* (DTA) is necessary to simulate a fault injection analysis based on timing failure attacks. We implement the physical layout of AES S-box in TSMC 65 nm technology using standard ASIC design tools.

Our contributions in this work are as follows: (1) We introduce a generic fault model for simulating biased fault attacks based on timing failure of hardware implementations; (2) We demonstrate substantial quantitative improvements in the effectiveness of statistical fault injection analysis techniques; (3) We further stimulate the field of deep learning applied to cryptographic security, by introducing a convolutional neural network to learn the most efficient metric to evaluate different features of the data distribution to recover a secret key.

The rest of the paper is organized as follows: In Sect. 2, background and prior works on statistical fault analysis are presented. Section 3 describes our design and simulation flow for dynamic timing analysis of the AES S-box. Section 4 introduces the extracted timing-based fault model along with multiple features of data distribution under fault injection with varying intensities. In Sect. 5, the architecture of the convolutional neural network employed in FIMA-NN is presented, and important guidelines for training the network are discussed in Sect. 6. Section 7 presents the results, and the paper concludes in Sect. 8.

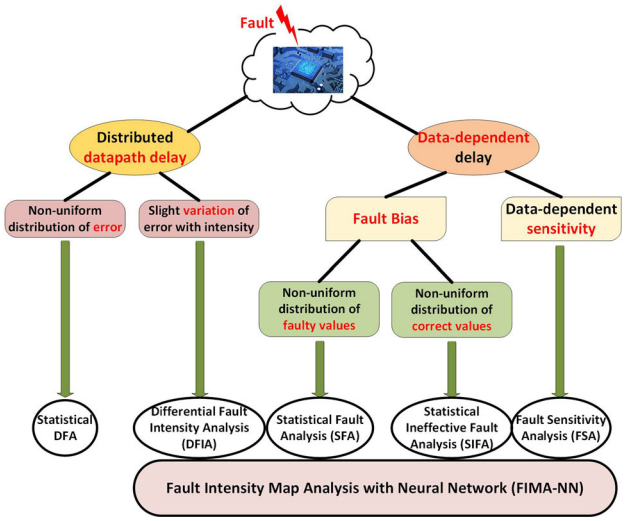


Fig. 1 Statistical properties of a cryptographic hardware implementation exploited by various statistical fault analysis techniques

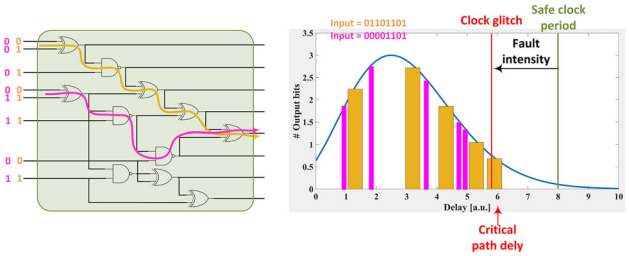


Fig. 2 Pictorial representation of the hardware implementation of a logic function with the corresponding data-dependent distribution of datapath delays

2 Background and prior work

2.1 Statistical fault analysis

The fundamental properties of hardware implementations exploited by most statistical fault analysis (SFA) techniques are data-dependent and intensity-dependent response of the hardware to fault injections. Various properties of an implementation employed as observables to identify the correct key in different SFA techniques are illustrated in Fig. 1. The intensity-dependent error distribution is exploited in techniques such as DFIA and statistical DFA while data-dependent error is the property used in biased fault techniques.

In the hardware implementation of a logic function with multiple input/outputs using standard CMOS gates, the delay of signal propagation from input to output is distributed which also depends on data. Most likely only a small number of output bits experience the longest delay which corresponds to the *critical path delay*, as illustrated schematically in Fig. 2. By injecting faults into the logic block in a clock/voltage glitch scenario, most likely, only the output

bit corresponding to the critical path delay experiences an error at low intensities. As a result, most errors have a low Hamming weight (HW). Hence, the distribution of the error values is non-uniform which also depends on the intensity. This property is exploited in the statistical DFA [17].

While statistical DFA still requires differential encryptions to capture the distribution of the error, it relaxes the assumptions on precise control of the attacker on fault injection. Differential fault intensity analysis (DFIA) further reduces the assumptions of fault model [10]. By slightly increasing fault intensity, only 1–2 more bits of the output experience an error (Fig. 2). DFIA exploits the fact that the HW of the error, hence the HW of faulty values, changes slightly with gradual increase in the fault intensity. Although DFIA is more efficient compared to biased-based fault attacks, it requires the faulty values for the same set of plaintexts and initial conditions which can be a limitation in nonce-based authenticated ciphers.

Statistical fault analysis (SFA) is based on the observation that the distribution of faulty values at the output of the logic function under attack is non-uniform, or biased [18]. SFA requires only the faulty outputs for random plaintexts to calculate the distribution of the intermediate variable under attack. The fault model of SFA is much simpler than DFA and DFIA. However, a wide range of countermeasures have been developed that either detect errors in the cipher operations and suppress the faulty values, or randomize the distribution of data under fault attack [3,12,31].

One of the first SFA techniques that obviated the need for the faulty values is fault sensitivity analysis (FSA) [20]. The property of faults exploited in FSA is data-dependent fault sensitivity, which is defined as the fault intensity at which errors start to appear. FSA is successful even in the presence of countermeasures suppressing faulty values; however, it requires a precise control over fault intensity to observe the correlation between intensity and data. Further, although FSA does not require faulty values for analysis, the attacker must still distinguish between the faulty and correct ciphertexts.

An alternative technique that does not require faulty values is statistical ineffective fault analysis (SIFA) [8]. Ineffective faults are defined as fault injections that do not induce an error into the intermediate variable. Due to data dependency of error, the distribution of correct values, under fault injection, is biased, which is exploited in SIFA. Depending on the fault model, i.e., location and injection mechanism, the bias of correct values can be less than the bias of faulty values. Further, the probability that the fault induces errors is higher than ineffective faults. Hence, SIFA requires more fault injections than other biased-based techniques to acquire a sufficient amount of data with meaningful bias. However, the simple fault model of SIFA results in a powerful technique; SIFA only requires correct outputs for random plaintexts and initial conditions.

Later, fault intensity map analysis (FIMA), a technique which builds on and generalizes DFIA and biased-based techniques, such as SIFA and SFA, was introduced in [26]. FIMA combines the biased distribution of correct ciphertexts under a correct key hypothesis with the variation in data distribution with fault intensity to reduce the number of fault experiments required to recover a secret key. The authors in [26] used FIMA with a classical metric measuring bias and variation in distribution with intensity based on the L_4-norm to recover the secret key of a software simulation of the Ascon cipher, but did not explore neural networks. FIMA, augmented with a neural network key distinguisher (FIMA-NN), as well as its application to the recovery of an AES secret key, is subsequently described in detail.

3 Timing-based fault model

3.1 Timing failure attacks

A wide range of FIA techniques exploit the timing failure of CMOS circuits as a result of fault injection mechanisms such as voltage/clock glitch and temperature variations [2,5,32]. Further, it has been shown that electromagnetic (EM) fault injections may result in timing failure of CMOS circuits [22]. Depending on how the timing failure is exploited to find the secret key, we divide FIA techniques into three categories as follows.

3.1.1 Classical differential fault analysis (DFA)

This group of techniques exploit the propagation of errors in cipher state due to diffusion operations of the cipher. Examples of DFA attacks on AES are [13,35]. In these attacks, an error in a particular location of the cipher state propagates through the state, after cipher operations, with deterministic relations between the errors. Rather than particular properties of the fault, classical DFA techniques rely on the deterministic flow of errors in the cipher structure to find the secret key. Hence, any type of error induced by a fault injection mechanism in a desired location of the state results in a successful attack. Major limitations of classical DFA techniques include the requirement for both correct and faulty values and the precise location of fault induction.

3.1.2 Statistical DFA

Rather than the flow of errors in the cipher state, statistical DFA techniques, including the attack in [17] and DFIA, exploit the properties of the error induced by fault injection. As illustrated in Fig. 2, timing failure as a result of fault

injection, with proper intensity, only affects one or a few output bits, corresponding to the critical path. Hence, the distribution of the induced error will be non-uniform, or biased. Statistical DFA relaxes the assumption of classical DFA on precise fault location; however, both correct and faulty values are still required for a successful attack. Since the distribution of error is biased, this class of techniques is often referred to as biased fault analysis in literature. However, in this work, we define a biased fault technique as follows.

3.1.3 Biased fault analysis

The class of techniques that exploit the biased distribution of an intermediate variable under fault attack, rather than the distribution of the error, is called biased fault analysis. As we will demonstrate in the next section, the distribution of a variable under attack is biased only if the probability of error, for a given fault intensity, depends on data. As illustrated in Fig. 2, the propagation delay of a CMOS logic circuit depends on data. Consequently, by injecting a fault with proper intensity, the output will be faulty only for a particular set of data. This results in the biased distribution of data under fault injection. Fault injection analysis techniques such as SFA and SIFA exploit the biased distribution of only faulty values and only correct values, respectively. As a result, these techniques do not require both faulty and correct values.

Standard timing analysis methodologies in VLSI circuit design flow can be used to detect vulnerability of the hardware implementation of a cipher to timing-based fault attacks. We employ timing analysis to develop a statistical fault model for training neural network key distinguishers. Before introducing the fault model, we describe proper models for the vulnerability of a hardware implementation of cipher operations to different classes of timing-based fault injection analysis.

3.1.4 Static timing analysis (STA)

STA is the dominant methodology for timing closure of VLSI circuits in the automated IC design flow. STA provides the propagation delay of all paths of a logic block at the circuit level. The delay distribution obtained by STA shows the worst case scenario and does not provide information about the data dependency of the delays. As a result, STA is not sufficient to provide information about the vulnerability of a cipher implementation to biased fault analysis. However, STA reveals the distribution of delays over different output bits of a logic block and thus is useful for modeling the response of the hardware to differential fault attacks.

3.1.5 Dynamic timing analysis (DTA)

The proper timing model of a hardware implementation for simulating the response to biased fault attack must include the data dependency of delays when the cipher operation under attack is executed in one clock cycle. Hence, DTA is necessary for simulating a biased fault analysis. We employ the transient time simulation on the transistor-level netlist of the hardware, in the analog domain, to capture the data-dependent timing.

3.1.6 Nonlinear model

In addition to data-dependent timing failure, a bias can also be induced in the distribution of values at the output of a nonlinear logic function if a fault is injected in the internal operations of the function. In this case, the error does not need to be data dependent; any random error can result in a biased data distribution. The bias is a result of non-bijective nonlinear components of the function. This type of fault is exploited in the SIFA attack of [7] on a software implementation of AES. Further, if a hardware implementation of a nonlinear function is executed in several clock cycles, a random fault at an intermediate cycle can result in a similar biased distribution, as shown in [27].

3.2 Timing analysis

Timing-based biased fault attacks target the nonlinear operation of ciphers as the fault location. In many block ciphers, including AES, *SubBytes* or S-box constitutes the nonlinear operation. In order to obtain a statistical model for data distribution under fault injections, we simulate the datapath of the S-box in AES as shown in Fig. 3. The state register contains the intermediate variable under attack. At each round of AES operation, the contents of the state are processed by the S-box and the result is loaded into the state at the next clock cycle. A multiplexer is included at the input of the register to enable loading of input data at the start of AES execution. The buffer at the output of the register simulates the loading effect of the circuits for cipher operations subsequent to S-box.

The physical layout of the circuit for the S-box datapath of Fig. 3 is implemented using standard electronic design automation (EDA) tools as shown in Fig. 4. In this work, the compact Canright S-box scheme [6] is implemented in TSMC 65 nm technology at nominal threshold voltage. The HDL code of the S-box datapath is synthesized using Synopsys compiler and TSMC standard cell library. The resulting netlist is used in Cadence SOC Encounter for place and route (P & R). The routed netlist of the design, along with the extracted parasitic components of the circuit, is the input to the Synopsys PrimeTime tool for STA. As mentioned earlier,

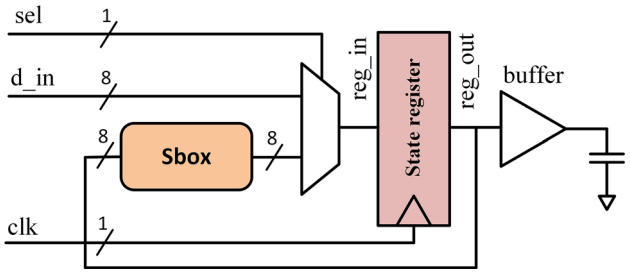


Fig. 3 Implemented datapath of AES S-box for simulating the effect of timing-based fault attacks

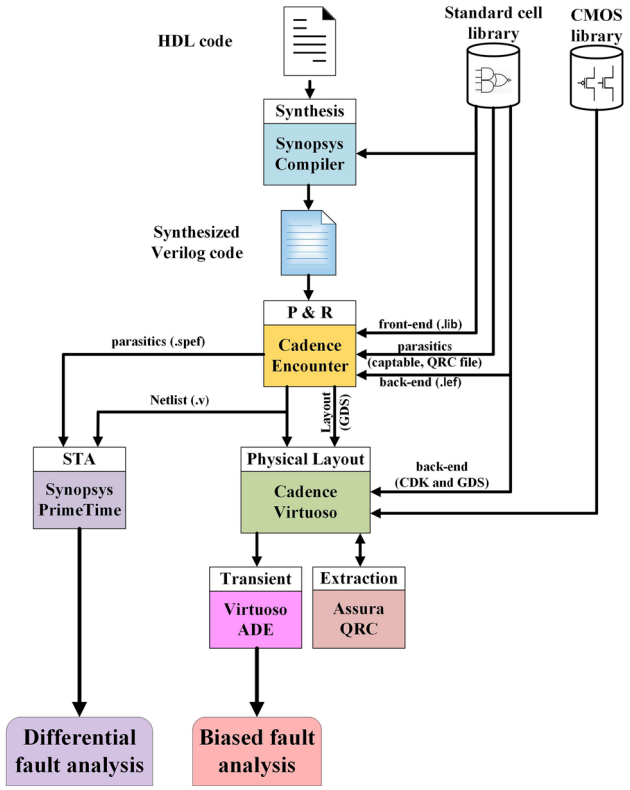


Fig. 4 Design and simulation flow using standard EDA tools for static and dynamic timing analysis of AES S-box for detecting vulnerability to differential and biased fault analysis based on timing failure

the results of STA can be used to model the response of the circuit to differential fault attacks.

The results of STA using Synopsys PrimeTime for the S-box circuit of Fig. 3 implemented in TSMC 65 nm technology at the nominal supply voltage of 1.2 V are shown in Table 1. The propagation delays in this table are measured from the input of the S-box to the output bits of the state register. The setup time and hold time of the corresponding registers are also reported in the table. It is observed that the output bit 3 experiences the longest delay which corresponds to the critical path of the circuit. Further, the difference between the longest and shortest delays is less than 38 ps. It implies that a fault injection mechanism, such as clock/voltage glitch, temperature variation or EM injection, should induce a timing failure with a resolution of higher than 38 ps. One solution to enable fault injections with lower intensity resolution, and thus simpler fault attacks, is reducing the supply voltage. At reduced voltage, the circuit experiences longer delays and the required resolution of fault intensity is lower. In the following, we will use the reduced supply voltage of 0.9 V for transient time simulations.

Table 1 Results of STA for AES S-box with nominal supply voltage of 1.2 V using Synopsys PrimeTime

Bit #	Delay (ps)	Setup time (ps)	Hold time (ps)
0	907.5	46.5	10.3
1	929.2	46.5	10.2
2	929.5	46.5	16.7
3	937.9	46.5	10.3
4	937.4	46.5	16.7
5	933.8	46.5	16.7
6	927.0	46.5	16.7
7	937.0	46.5	16.7

In order to simulate a timing-based biased fault attack, we conduct a transient time analysis in the analog domain. To obtain a transistor-level netlist of the design, we import the routed layout and the Verilog netlist of the design, generated by Encounter, to Cadence Virtuoso. The physical layout of the circuit is generated in Virtuoso, and the RC parasitic components are extracted using Assura QRC. The Virtuoso Analog Design Environment (ADE) is used for running a transient time simulation on the extracted netlist of the design. The time-domain waveforms are analyzed for the purpose of dynamic timing analysis. Examples of the time-domain waveforms for two different data values at the S-box input are shown in Fig. 5.

The propagation delay of every output bit for a given input to the S-box is obtained by analyzing the waveforms of Fig. 5. The delay is calculated as the time offset from the rising edge of the clock signal such that the output waveform is stabilized at the final value plus the *setup* and *hold* times of the registers as shown in Table 1. The output is considered stabilized when the waveform has no further variations larger than 10% of the supply voltage.

The data dependency of delays is observed in Fig. 5. When the input to the S-box is the value 0xFB, the longest delay is 954 ps, corresponding to output bit 2. When the S-box input takes the value 0xC5, the longest delay will be 1.22 ns corresponding to the output bit 1. In this case, the propagation delay of output bit 2 is around 1.16 ns, in contrast to the delay of 954 ps with the input 0xFB. When the (glitched) clock period is 1.2 ns, the S-box computation of 0xFB is carried out without any error while the computation of 0xC5 is faulty

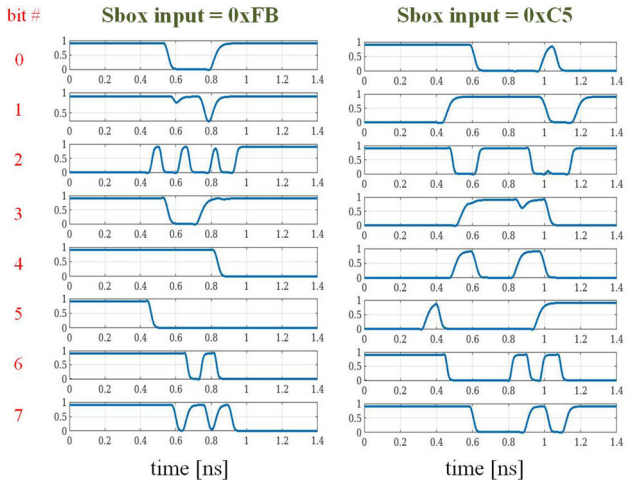
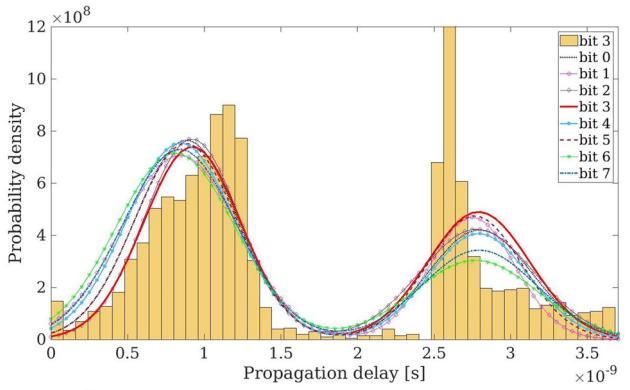


Fig. 5 Waveform of the output bits during S-box computations for two different inputs at supply voltage of 0.9 V; rising edge of the clock starts at time = 0



bit	0	1	2	3	4	5	6	7
μ_0 [ns]	0.88	0.84	0.92	0.92	0.85	0.88	0.81	0.84
μ_1 [ns]	2.80	2.75	2.78	2.79	2.79	2.76	2.77	2.79

Fig. 6 Propagation delay distribution for eight output bits of S-box obtained from transient time analysis at a supply voltage of 0.9 V with Monte Carlo simulation (mean of two components of bimodal GMM shown in the table)

with the error in output bit 1 and most likely bit 2. Hence, the distribution of correct values under a clock glitch with period 1.2 ns shows a high probability for the value 0xFB and low probability for 0xC5 which implies a biased distribution.

The distributions of delays for the output bits of the S-box over all input data, as extracted from transient time simulations, are shown in Fig. 6. These results are obtained from Monte Carlo simulation in Cadence ADE to account for process variations. For clarity, a histogram of delays only for output bit 3, with the longest delay, is plotted in the figure. The curves are the probability density functions (pdf) of a bimodal Gaussian mixture model (GMM). The mean of two components of the GMM distribution, i.e., μ_0 and μ_1, for all output bits is also shown in the figure.

The delay distributions of Fig. 6 reveal two significant timing properties of the S-box circuit of importance to biased fault analysis;

- The fact that the distributions are spread over a range of values reflects data dependency of delays. Hence, a biased fault analysis using timing failure can be conducted successfully on this design.
- The delay distributions for different output bits are centered over different values. Consequently, by changing fault intensity, error distribution also changes. This property implies that variations in fault intensity might reveal multiple features in data distribution that can be exploited to conduct a more efficient fault attack.

3.3 Data dependency bias

In the previous section, it was shown empirically that data-dependent timing failures (error induction) in a logic function result in fault bias. We define *fault bias* as the deviation of the distribution of an intermediate variable under fault injection from the uniform distribution. In this section, we demonstrate that data dependency of error probability is the necessary condition for fault bias. If the probability of an error is the same for all input data to a bijective function, such as an S-box, then data distribution under fault injection will be uniform.

Assume the intermediate variable X, i.e., the output of the S-box under attack in AES, takes the values X' as a result of fault injection. The distribution of X, without fault injection, in a cipher with random input plaintext is uniform. Denoting the error induced by the fault injection as e, the faulty intermediate value is $X' = X \oplus e$. The distribution of the faulty value can be obtained as

$$\begin{aligned} p_{X'}(x') &= \sum_x p_E(X \oplus e | X = x) p_X(x) \\ &= \frac{1}{2^8} \sum_x p_E(X \oplus e | X = x) \end{aligned} \tag{1}$$

in which the 8-bit variable X at the output of a S-box is uniformly distributed. From this equation, we observe that if the error is independent of the data X, we have $p_E(X \oplus e | X = x) = p_E(x \oplus e)$. As a result, the distribution of the intermediate variable under attack, X', is uniform. Hence, when a fault injection mechanism induces an error independent of data, no bias is observed and the fault attack is unsuccessful. In other words, for a successful biased fault analysis, the induced error must be data dependent. This implies that the error distribution must depend on data.

A simple model for the fault distribution in the literature of biased fault analysis is random-AND. In this model, the faulty values are represented by $X' = X \odot u$, in which the

error mask u, as well as the intermediate variable X, is uniformly distributed. The random-AND model is a simplified model of data-dependent error; it is equivalent to the special case of bit-flip probabilities $p_{1\to0} = 0.5$ and $p_{0\to1} = 0$. This is what the AND operation models; the AND operation does not change a bit 0 ($p_{0\to1} = 0$). However, for a bit 1, when the error mask u is uniformly distributed, the bit flips to 0 with a probability $p_{1\to0} = 0.5$.

As discussed in Sect. 3.1, a random fault in the internal operations of a nonlinear function might also result in biased data distribution. This effect is again due to data-dependent error probability. Take the simple example of an AND gate $c = a \odot b$. If $a = 0$, the output of the gate is always the correct value $c = 0$, irrespective of the value of b. In this case, even if the value of b is faulty, the probability of error under fault injection is 0. With the input $a = 1$, the output c is faulty if b is faulty. In this case, if b experiences a random fault, the output c is erroneous with a probability 0.5. Hence, the probability of error depends on the input data.

In this paper, we focus on the data-dependent timing failure of hardware implementations to conduct a biased fault analysis. In the next section, we develop a generic model for fault distribution that reflects both data- and intensity-dependent properties of fault injections. We point out that the model is used for training the neural network key distinguishers in FIMA-NN attacks, while the actual simulated faults to recover the secret key are modeled using the waveforms of transient time simulations.

4 Timing-based fault distribution

Most existing biased fault analysis techniques employ only a few features of data distribution to identify the correct key. An attack approach exploiting multiple features can bypass most countermeasures while still providing reasonable efficiency. In this section, we demonstrate that data distribution under varying fault intensities reveals multiple features of a correct key that can be exploited to achieve an efficient attack with a simple adversary model.

To develop a data-dependent fault model, we fit a Generalized Extreme Value (GEV) distribution on delays of the 8 bits at the output of the S-box for every input data. The cumulative distribution function (CDF) of GEV distribution is [9]

$$F(x;\mu,\sigma,\xi) = \exp\left\{-\left[1+\xi\left(\frac{x-\mu}{\sigma}\right)\right]^{-\frac{1}{\xi}}\right\} \tag{2}$$

in which $\mu \in \mathbb{R}$ and $\sigma > 0$ are location and scale parameters, respectively, and $\xi \in \mathbb{R}$ is the tail index which determines the shape of the function.

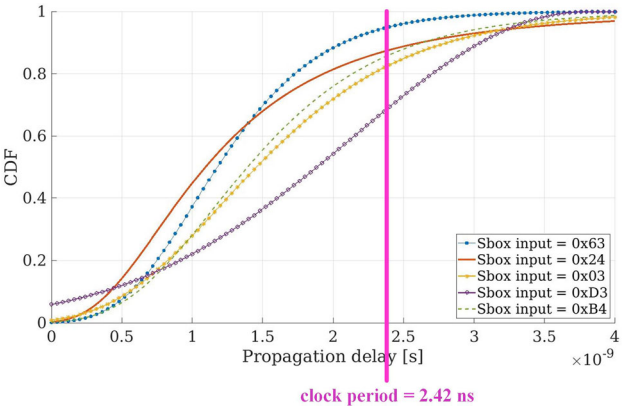


Fig. 7 Data-dependent fault distribution for different input values at S-box input

For every input $d = 0, 1, \ldots, 255$ to the S-box, we obtain a probability distribution function $F(x;\mu_d,\sigma_d,\xi_d)$ for the propagation delays, with the function $F()$ defined in (2). The fitted distributions for a few different data values are shown in Fig. 7. The value $p_d = F(T^*;\mu_d,\sigma_d,\xi_d)$ is the probability that the delay of all output bits of the S-box is less than the (glitched) clock period of T^* when the S-box input is the value d. It is observed in Fig. 7 that when the clock period is 2.42 ns, S-box computation for input data 0x63 is error-free with a probability of more than 95%. When the S-box input is 0x03 and 0xD3, the probability that the propagation delays are within one clock period is less than 83% and 70%, respectively. The probability of error-free output for S-box inputs 0x24 and 0xB4 is around 88%.

To simulate a biased fault using the extracted distributions, we first sort the S-box output bits in the order of descending delays, according to the delays extracted in transient time simulations (Fig. 6), denoted by $\mathbf{s} = (s_0, s_1, s_2, \ldots, s_7)$. For a clock period of T^* and input data d, we calculate the probability $P_d = 1 - F(T^*;\mu_d,\sigma_d,\xi_d)$ that the output will be faulty. Hence, P_d percent of the output bits, corresponding to the largest delays, will be faulty. Considering that the value $\mathbf{s}$ is the correct result of computations, the bits of faulty output are obtained as

$$\begin{aligned} s_i^* &= r_i, \quad i \le \lfloor 8\cdot P_d \rfloor \\ s_i^* &= s_i, \quad i > \lfloor 8\cdot P_d \rfloor + 1 \\ s^*_{\lfloor 8\cdot P_d\rfloor+1} &= \begin{cases} r, & \text{with prob. } \ 8\cdot P_d - \lfloor 8\cdot P_d \rfloor \\ s_{\lfloor 8\cdot P_d\rfloor+1}, & \text{otherwise} \end{cases} \end{aligned} \tag{3}$$

in which r and r_i's are uniformly distributed binary random values.

We point out that when the result of an 8-bit S-box computation is faulty under a timing failure, with a probability P_d, it implies that a number of $8 \cdot P_d$ output bits with the largest delay will be erroneous. This is formulated in Eq. (3).

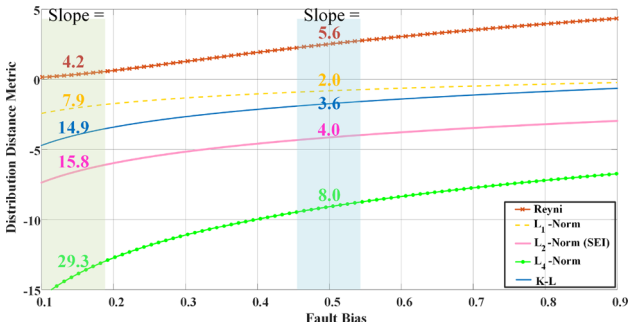


Fig. 8 Sensitivity of data distribution to fault intensity variations; fault intensity variations introduce larger variations to data distribution with lower bias (incorrect key)

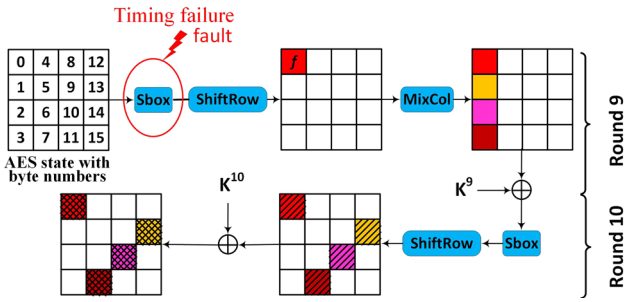


Fig. 9 Fault location on AES at the beginning of round 9; AND gate at the output of S-box simulates a biased fault

When a timing failure occurs, at low fault intensity, the setup and hold time of the registers are violated. Hence, the registers fall into a *meta-stability* state in which case the sampled output of the registers can be a random value. Further, output bits of the S-box might change value multiple times until becoming stable, according to the waveforms of Fig. 5. Thus, at higher fault intensities, the sampled output bit might take a random value. This is the reason why faulty values of the output bits in Eq. (3) are considered random binary values.

4.1 Variation in distribution with intensity

The metric to measure the smooth variation in errors with fault intensity exploited in DFIA is the Hamming Weight of the faulty values which is valid only if the faulty values are acquired for the same plaintext and initial conditions. FIMA generalizes this idea by observing that the variation in data distribution under fault injection is also smooth with fault intensity for the correct key. In this section, we measure the variations in data distribution with fault intensity by evaluating the *sensitivity* of the distance between data distribution and uniform distribution, as a reference, to fault bias variations.

Various distance metrics between probability distributions are defined in the literature of information theory and statistics. The most common metric, used in statistical fault analysis, is the Norm-2 distance which is commonly referred to as Square Euclidean Imbalance (SEI) in the literature. The common metrics in information theory include Kullback–Leibler (K–L) divergence, or relative entropy, and Rényi divergence. Other metrics include Norm-1 distance, called total variation distance, and Norm-4 distance. It is shown that relative entropy is a special case of Rényi divergence [36]. In addition, total variation distance is also related to the Rényi divergence. The variation in data distribution with fault intensity is shown in Fig. 8 with different distance metrics plotted in logarithmic scale.

It is observed in Fig. 8 that depending on the distance metric used for evaluating the variations in data distribution, the sensitivity can be different; higher-order norm distance metrics exhibit larger sensitivity to fault intensity variations. When data distribution is close to uniform, Rényi divergence shows the lowest sensitivity. Further, the SEI metric and K–L divergence exhibit similar sensitivity of the distribution. Hence, to exploit the variation in data distribution to fault intensity changes as a feature for identifying the correct key with classical statistics, it is critical to employ the proper distance metric to measure the variations. In our work, a neural network is employed to learn the proper metric to evaluate different features of the data distribution.

4.2 Fault image of AES

AES is a U.S. federal and principal worldwide standard for secret key cryptography outlined in [1]. In particular, AES-128 uses a 128-bit key and consists of 10 rounds and a composition of four transformations: *SubBytes*, *ShiftRows*, *MixColumns*, and *AddRoundKey*. In this section, we inspect the *fault image* of AES to identify the distinct features of data calculated with a correct key candidate.

The distribution of an intermediate variable under fault injection with varying intensities reveals multiple features of fault injection, which are exploited in various statistical techniques. Fault image, as defined in [26], is the 2-dimensional map of data distribution versus fault intensity which constitutes the data representation analyzed in FIMA-NN for identifying the correct key.

By injecting faults into the operation of S-box 0 at the beginning of round 9 of AES, as shown in Fig. 9, byte 0 of the state assumes a biased distribution. A fault image is constructed for this state byte as the intermediate variable. To calculate this intermediate variable from the output ciphertext, we need bytes (0, 7, 10, 13) of the ciphertext and 10th round key (K^{10}). With a guess for these 4 bytes of the key, the intermediate variable X can be calculated using the inverse operations of the cipher. It is observed in Fig. 9 that to calculate the output of a S-box at round 9, K^9 is also required.

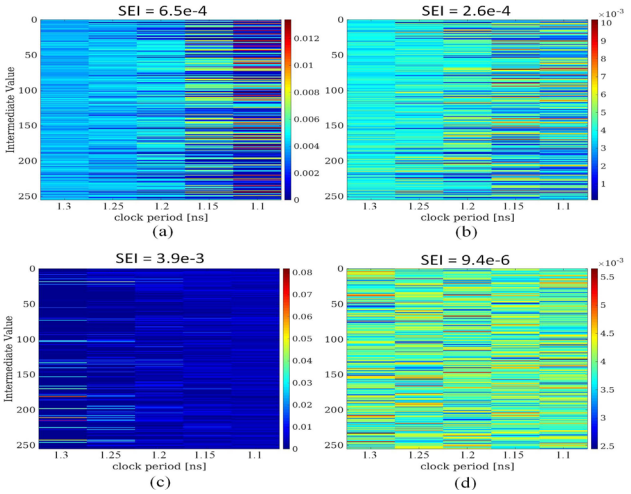


Fig. 10 Fault image of one state byte of AES with timing failure at the beginning of round 9; **a** distribution of fault-free intermediate values calculated with correct key, **b** distribution of all intermediate values with correct key, **c** distribution of faulty intermediate values calculated with correct key, **d** distribution of intermediate values calculated with incorrect key; color map represents probability

However, K^9 effectively adds a constant to the value of X. By denoting the state byte 0 as S_0, the effect of K^9 can be expressed as

$$S_0 = X \oplus \left(14 \cdot K_0^9 \oplus 11 \cdot K_1^9 \oplus 13 \cdot K_2^9 \oplus 9 \cdot K_3^9\right) \quad (4)$$

in which multiplications by constants are performed over Galois Field $GF(2^8)$ modulo the polynomial $x^8 + x^4 + x^3 + x + 1$. The term in parenthesis is an 8-bit constant. Addition of this term is equivalent to a certain permutation of the probability mass function of X. Hence, most statistical features of X and S_0 are the same except for the specific values of the variables that take a particular probability value.

By calculating the intermediate variable X with a guess for 4 bytes of the key during multiple faulted encryptions, a fault image can be constructed as the distribution of X at different fault intensities. We denote the fault image as $p_X(x, I; K)$ which represents the probability that for a fault intensity I, the variable X calculated with a key guess K takes the value x.

The fault image of AES for one byte of the state under attack is shown in Fig. 10. The distributions of values calculated with all output ciphertexts, only correct outputs and only faulty outputs, using the correct key are shown in parts (a), (b) and (c), respectively. Part (d) shows the distribution of data calculated with an incorrect key candidate. It is observed that the bias of only faulty values under a timing failure attack is highest, with the bias decreasing with fault intensity. It implies that an SFA technique which exploits only faulty values achieves higher efficiency at lower fault intensities. On the other hand, the bias of only correct values increases with faulty intensity. Hence, in SIFA attacks, higher intensities improve the efficiency. The distribution of all values under timing failure is also biased with the lowest bias value.

The distinct features of the fault image corresponding to a correct key candidate can be summarized as follows:

1. *Fault Bias* The most pronounced feature of the correct fault image in Fig. 10 is the fault bias; particular values of the intermediate variable take much higher probabilities. However, different hardware implementations exhibit different levels of bias. Countermeasures can also reduce the observed bias significantly.
2. *Fault Sensitivity* We generalize the definition of fault sensitivity, as defined in [20], to the fault intensity at which the probability of a particular value of the intermediate variable deviates from the uniform distribution. It is observed in Fig. 10 that fault sensitivity is data dependent.
3. *Smooth Variations* As shown in Fig. 8, the distance between data distributions at close fault intensities is small under the correct key assumption. However, it should be noted that a proper distance metric is required to distinguish the variations under the correct and incorrect key candidates, especially when the fault bias is small.
4. *Intensity-dependent Distribution* Although the data distribution is biased at different fault intensities, the set of values with the highest/lowest probabilities changes at different intensity values. This property originates from intensity-dependent distribution of errors.

Although existing techniques exploit one or two of the above features in data distribution, we introduce FIMA-NN that employs deep learning to extract all statistical features. As a result, the efficiency of the attack is improved and the attack is successful even if countermeasures suppress part of the features.

5 FIMA with neural network

5.1 Fault model for attack on AES

In the fault model of a FIMA-NN attack on AES, we assume that the attacker is able to inject faults into the AES state, at the beginning of round 9, with varying intensities. The input plaintexts and initial conditions for multiple encryptions can be random. As a result of fault injection, any one of the following features may appear in the distribution of the variable under attack: (1) The distribution of data is biased for a range of fault intensities; (2) The data distribution is dependent on the intensity; (3) The variation in the distribution with gradual change of intensity is smooth; (4) The fault sensitivity is data dependent.

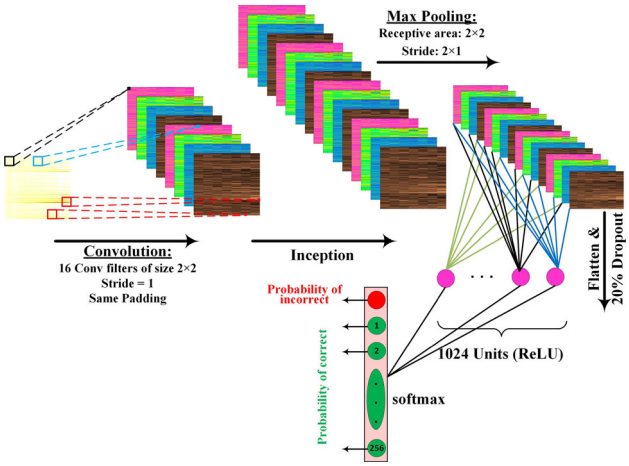


Fig. 11 Architecture of CNN, employed as the key distinguisher, consisting of a convolutional layer, an inception and max pooling, fully connected layer, and multi-class softmax output

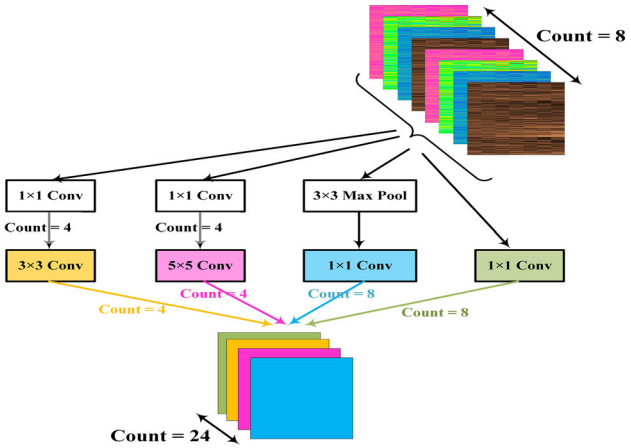


Fig. 12 Inception module as the second layer of CNN key distinguisher showing the number of feature maps generated by each component

The attacker has prior knowledge of which of the faulty or correct values are biased, and can filter out the proper values. If all values are biased, FIMA-NN uses all data and the attacker does not require differential encryptions to identify faulty values. FIMA-NN can exploit multiple statistical features to improve the efficiency. Countermeasures that detect errors in the cipher operation cannot protect the implementation against FIMA-NN, but can reduce the attack efficiency. The attacker does not need precise control over fault injections. Since the input plaintexts and initial conditions need not to be known, FIMA-NN can successfully attack various block cipher modes of operation.

5.2 CNN key distinguisher

An important consideration in developing a neural network (NN) key distinguisher is the different response of hardware implementations to fault injections. Various hardware platforms exhibit different levels of data and intensity dependency of errors. As a result, the distinct features of data distribution may appear at different locations of the fault image. Hence, the *spatial invariance* of the neural network is an important characteristic to identify the correct key.

We employ a convolutional neural network (CNN) to extract the proper features of the fault image. The prominent properties of CNNs that make them an interesting architecture for deep neural networks (DNN) in various applications are the parameter sharing and localized feature detection. Each layer of a CNN consists of a set of small size kernel functions that process the entire input data. Since the same set of kernels process different locations of the input to the layer, a CNN can learn the same feature at different locations of the input data. This is due to the *equivariance* property of CNNs [15] which is desirable to detect the features of fault images from various hardware implementations.

The architecture of the CNN, employed in this work as the key distinguisher, is shown in Fig. 11. It consists of one convolutional layer, an inception module followed by a max pooling, a fully connected layer, and a softmax output. The convolutional layer comprises 16 kernels of size 2×2. The stride of the kernels is 1, and the input image is processed with *same* padding. The activation function of each convolution is the rectified linear unit (ReLU).

The size of kernels at each layer of CNN is important in detecting features of different sizes in an image. We refrain from using a fixed kernel size in the second layer of the CNN key distinguisher. Instead, the second layer consists of an *inception* module. GoogLeNet, often called Inception network, first introduced inception modules to use network parameters much more efficiently, hence, resulting in a deeper network [34].

The special property of the inception module, desirable for processing fault images, is a construction of kernels with different sizes processing the input in parallel. As a result, the inception layer is capable of learning more complex features of different sizes in the fault images. The employed inception module is shown in Fig. 12. All kernels use *same* padding and ReLU activation function. Two sets of 1×1 kernels, each with a size of 8 kernels, are used: One set processes the input maps directly while the other set processes the maps after a 3×3 max pooling layer. In addition, 4 kernels of sizes 3×3 and 5×5 are used. The total number of feature maps at the output of the inception layer is equal to 24.

The inception module is followed by a max pooling layer, which is commonly used in CNNs and subsamples the feature maps. This results in more efficient parameter usage in the subsequent layers of the network. The max pooling layer in the key distinguisher of Fig. 11 processes the input in blocks

of 2×2 with a stride of 2 in the vertical direction (values of the intermediate variable) and a stride of 1 in the horizontal direction (intensity values). The type of padding is *valid*; as a result, the feature maps shrink by a factor of 2 in the vertical direction and decreases by 1 unit in the horizontal direction.

The final layer of the key distinguisher, similar to most CNNs, is a fully connected (FC) layer consisting of 1024 neurons with ReLU activation. In order to prevent overfitting on a specific data distribution, especially when the size of available training data is limited, we use dropout regularization with a rate of 20% [33]. Overfitting can substantially reduce the efficiency of FIMA-NN. If the training data do not include the typical fault behavior of a particular hardware platform, an overfitted network requires an unnecessarily large number of data samples to identify the correct key.

The output layer of the NN key distinguisher is a softmax function with 257 output classes. The first class represents the incorrect key and the remaining 256 classes are assigned to 256 possible permutations of the fault image for the correct key, as explained in the next section. The softmax output provides the probabilities that a given fault image belongs to an incorrect key candidate or a given permutation of the fault image calculated with the correct key. The loss function of the network is the cross-entropy, and the Adam algorithm is employed for optimizing the network parameters.

While the NN key distinguisher provides the probabilities of correct/incorrect key candidates, one approach in identifying the correct key is comparing the probabilities with a threshold. The efficiency of this approach highly depends on the training data. If the fault image from a given hardware implementation has a large *similarity* with a subset of training data; then, the probability of the correct key can increase beyond the threshold which will unnecessarily increase the required number of fault injections to achieve a high probability. Instead, we use the *relative* probabilities of all key candidates to identify the correct key.

6 Training set generation

The training data set has a large impact on the efficiency of FIMA-NN. If the NN key distinguisher is overfitted on fault images constructed with a large size of data samples, its ability to distinguish the correct key with a limited data size is degraded. In this section, we provide guidelines to generate a proper training set for FIMA-NN. Before proceeding, we discuss a challenge in training the key distinguisher for learning the features of the fault image in AES.

As shown in Eq. (4), the effect of K^9 in the fault images appears as a particular permutation of the rows in the image. An immediate solution to incorporate the effect of K^9 is to include all 256 possible permutations of the correct fault image and label them as the correct key. However, this approach does not result in a well-trained network. The effect of K^9 is not a simple spatial translation of the features in the fault image. Instead, the value of the K^9 scrambles the fault image such that it would be difficult for the network to learn all possible features, at least in a network with few layers.

A simple solution to the problem of unknown K^9 in AES is proposed in [25], in which a CNN key distinguisher classifies fault images into two classes of correct/incorrect key, and all 256 permutations of a given fault image are tested by the CNN. In this work, we employ a CNN with 257 output classes, as shown in Fig. 11, in which one class corresponds to the fault image of an incorrect key and 256 classes represent different permutations of a fault image for the correct key. The key candidate that has a fault image with the lowest probability of classifying as incorrect is selected as the correct key.

The proper fault images for training the CNN of Fig. 11 are generated using the statistical model introduced in Sect. 4. According to Figs. 6 and 7, errors start to appear at the output of the S-box when the clock period (timing stress) starts from around 3.5 ns. When the clock period is reduced to around 2.5 ns, at least one output bit of the S-box, thus the result of S-box computation, will be faulty for every input data. Hence, in generating the training fault images, we change the fault intensity from 3.5 to 2.5 ns with equidistant intensity values.

We point out that further reducing the clock period beyond 2.5 ns does not favor FIMA-NN as it (indirectly) inspects the variations in the probability that an error appears at the output; it is shown in Sect. 3.3 that fault bias is a result of data-dependent probability of error. However, differential fault attacks, such as DFIA, might benefit from reducing the fault intensity further: Although the probability of error does not change by reducing the intensity beyond 2.5 ns, the distribution of the error (specific values of the errors) changes. Hence, differential fault attacks which inspect the values of the errors can exploit the variations in the error distribution as an observable to find the correct key. Biased techniques, such as SIFA and FIMA-NN, inspect the values of the intermediate variable under fault injection rather than the values of the errors.

The number of fault intensities in the range of low (3.5 ns) to high (2.5 ns) values is another parameter of FIMA-NN to be selected by the attacker. More intensity values provide a finer sampling of fault distribution. However, two important limitations for larger number of fault intensities include: (1) the attacker requires a fine control over fault intensity and (2) larger number of faulted encryptions are required since sufficient data should be collected at every intensity to obtain a decent estimate of data distribution. We select 5 equidistant fault intensities in the range [2.5, 3.5] ns, with a resolution of 200 ps. It is observed in Fig. 7 that changing the intensity by 200 ps results in less than 10% change in the probability of

error for the steepest curve. Hence, 5 intensity values provide a decent sampling of fault distribution.

In generating the training fault images, we also add clock jitter in injecting fault into the S-box operation. This is similar to *data augmentation* in which, given a limited size of training set, an extra data set is generated by adding noise to the available data. Data augmentation helps prevent overfitting in deep neural networks [4]. Considering that for a fault injection mechanism with a timing resolution of 200 ps, as explained above, the clock jitter is much less than 200 ps. To prevent overfitting, we select a pessimistic estimate of 100 ps for the clock jitter in generating the training images.

In this work, the training set for the CNN key distinguisher is generated according to the above considerations. In order to prevent overfitting on a specific data distribution and improve the success of the network even with a limited data size, the following considerations are important in generating the training set:

1. In order to learn all features of the fault distribution, the training data should include fault images constructed with a sufficient amount of data. Also, to prevent overfitting on the most obvious features of fault image, the training set should also include fault images with a limited size of data samples.
2. The training set should include fault images representing the distribution of only correct values in addition to those images that include unfiltered correct/faulty values. The trained neural network can be used to analyze a given fault image without having a prior knowledge on whether faulty and correct values are filtered.
3. The fault images in the presence of infective countermeasures should be included in the training set. Infective countermeasures usually replace the faulty values with random numbers. This type of fault image can also model the effect of noisy fault injections.
4. As discussed earlier in this section, different permutations of a fault image for the correct key are assigned to different classes. Hence, the training set must include all 256 permutations corresponding to different values of K^9.

We emphasize that the statistical model of Sect. 4 is only used to generate fault images for training the CNN key distinguisher. The actual FIMA-NN attack is deployed using the analog waveforms from transient time simulations, as explained in the next section. We randomly choose one iteration of a Monte Carlo simulation as an actual timing model for a particular ASIC implementation. For every clock period, the output of the S-box under attack is read from the analog values of the waveforms.

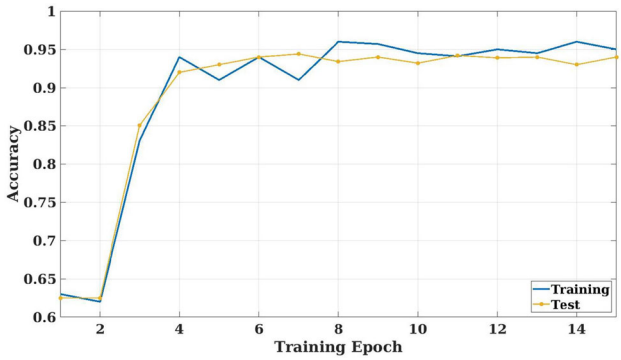


Fig. 13 Accuracy of the CNN key distinguisher during training (last batch accuracy) and test

7 Results

We use a physical layout of an AES S-box, after RC parasitic extraction, implemented in TSMC 65 nm technology with nominal threshold voltage, and Cadence Virtuoso ADE transient time analysis for simulating a timing failure fault injection. We use FIMA-NN to recover K^{10} when the simulated fault injection occurs during S-box computations at the beginning of round 9 of AES. The neural network key distinguisher is trained using the statistical fault model described in Sect. 4.

To find output bits of the S-box under timing failure attack, we read the value of the corresponding analog voltage obtained with transient time simulations in Virtuoso ADE. If the voltage value is stable at a 0/1 logic value, during the time window around the clock edge that satisfies the hold and setup time conditions of the registers, we take the value 0/1 as the S-box output bit; otherwise, we take a random binary value. The voltage value is stable at 0/1 if it shows variations less than 10% of the supply voltage. To simulate an infective countermeasure, faulty ciphertexts under attack are replaced by random numbers.

The CNN key distinguisher was implemented in TensorFlow, and executed on a PC with Intel Core-i7 CPU, 16 GB RAM, and Nvidia GeForce GTX 1080 GPU. Processing the fault images with the CNN key distinguisher for all key candidates required 5.2 hours.

7.1 Training CNN key distinguisher

The CNN key distinguisher in Fig. 11 is trained with a simulated training set generated according to the guideline presented in Sect. 6. The size of generated data is 32,000, of which 10% are chosen randomly for test. The fault images in the training set are generated for a random secret key for 5 equidistant (timing) intensity values in $T^* \in [2.5, 3.5]$ ns.

The CNN is trained using the adaptive moment estimation (Adam) algorithm and batch normalization. The batches

have a size of 64 samples and are chosen randomly from the training set. The training and test accuracy of the neural network at different epochs of training is shown in Fig. 13. We observe that test accuracy is saturated at a level of around 95%. We stop the training after 10 training epochs. Since the relative probabilities of the key distinguisher are used as the metric to rank the key candidates, the accuracy of Fig. 13 is sufficient to achieve a highly efficient attack.

7.2 Key recovery using all data

Faulty values of the intermediate variable under timing failure attack exhibit the highest bias, as shown in Fig. 10. Hence, if all outputs are available, in an unprotected cipher, the most efficient attack is obtained using only faulty values. However, filtering the faulty values is a limitation as the attacker requires a prior knowledge on the expected correct output of the cipher. We demonstrate that even if the faulty and correct values are not separated, FIMA-NN is a highly efficient attack.

The rank of the correct key candidate under a timing fault injection using all output values is compared in Fig. 14 with the maximum rank of incorrect keys versus the size of data samples. The data are collected at 5 different equidistant timing intensity values in the range $T^* \in [2.5, 3.5]$ ns with a simulated clock jitter equal to 100 ps. We define timing intensity as the period of time available for the S-box output bits to stabilize at their correct value. For comparison, the rank of key candidates in part (a) is calculated according to the classical SEI metric while part (b) shows the probabilities assigned to them by the CNN key distinguisher.

We observe in Fig. 14 that the classical SEI metric requires at least 12,150 ciphertexts to detect the correct key. However, the CNN distinguisher assigns the highest probability to the correct key after collecting only 960 output ciphertexts. Hence, FIMA-NN is 12.6 times more efficient than classical biased-based techniques using all faulty/fault-free ciphertexts. Additionally, although the probability assigned to the correct key by the CNN is relatively small, i.e., 40% at higher intensities, all other incorrect key candidates are assigned with smaller probabilities.

One consideration in comparing FIMA-NN with bias-alone fault analysis techniques is that the collected data experience faults with different intensities. Higher intensity values favor classical biased-based techniques, as data at low intensities exhibit smaller bias. As observed in Fig. 10, the distribution of intermediate variable using all output ciphertexts under attack has the lowest bias compared to only correct or faulty values. However, with a higher fault bias, e.g., by increasing fault intensity or using data values with larger bias, the efficiency of bias-alone techniques could improve.

We point out that in a real-world attack, the fault intensity to achieve a given bias in data distribution is unknown. All existing fault injection analysis techniques, including SFA and SIFA, *assume* that the proper intensity to obtain a pronounced feature is known. The amount of data required in a profiling step to identify the proper intensity is typically not reported in the literature to evaluate the efficiency of the attack. However, FIMA-NN has a great advantage in that such a profiling step is not required. All data at different intensities are exploited efficiently by FIMA-NN to identify the correct key.

7.3 Key recovery using correct data

In a cipher protected with error detection countermeasures that suppress faulty outputs, the efficiency of bias-alone fault analysis techniques can be substantially less than techniques that exploit only faulty values, as the bias of correct values under ineffective fault induction is smaller than faulty values (Fig. 10). The rank of key candidates using only correct values is shown in Fig. 15 for fault intensities in $T^* \in [1.0, 1.3]$ ns and a clock jitter of 100 ps. The amount of correct ciphertexts required to identify the correct key using the SEI metric, shown in part (a), is at least 4810, which is smaller than the value of 12,150 in Fig. 14 when all values are also used to measure the bias.

As observed in Fig. 10, the bias of correct values increases with fault intensity. Error detection countermeasures can make the attack even more difficult since the probability of ineffective faults is small, especially at higher intensities. While at low intensities, most of the outputs are correct, the data distribution exhibits small bias. By increasing the intensity, the bias of correct values increases; however, most outputs are faulty. Thus, in order to collect a sufficient amount of correct ciphertexts, a much larger number of faulted encryptions are required. FIMA-NN does not rely on the bias alone. As observed in Fig. 15b, the CNN key distinguisher can identify the correct key with only 1000 correct ciphertexts, i.e., an improvement in efficiency by a factor of 4.8 compared to bias-alone techniques.

7.4 Key recovery with infective countermeasures

Infective countermeasures detect errors in the cipher operation and replace the faulty state with a random value. Similar to error detection countermeasures, the bias of faulty values are diminished, and the observed bias in data distribution is only due to the correct values under ineffective fault inductions. The advantage of infective countermeasures over error detection is that the attacker requires differential encryptions to identify whether the fault induced an error into the cipher operations. The use of differential encryptions is a limitation in ciphers using certain modes of operation (e.g., nonces)

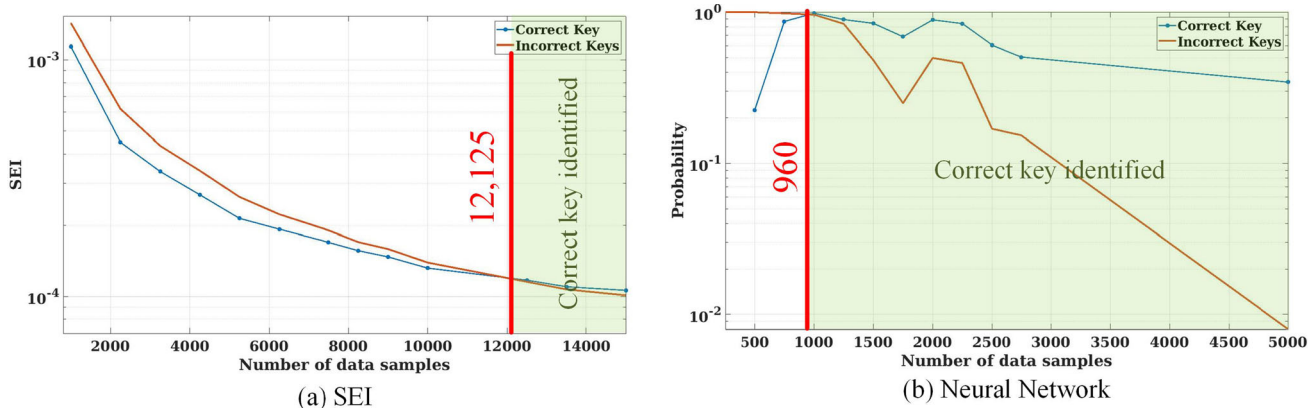


Fig. 14 Rank of key candidates for fault injections with 5 equidistant clock period in the range $T^* \in [2.5, 3.5]$ ns, and using all faulty/fault-free values; **a** classical SEI metric, **b** neural network

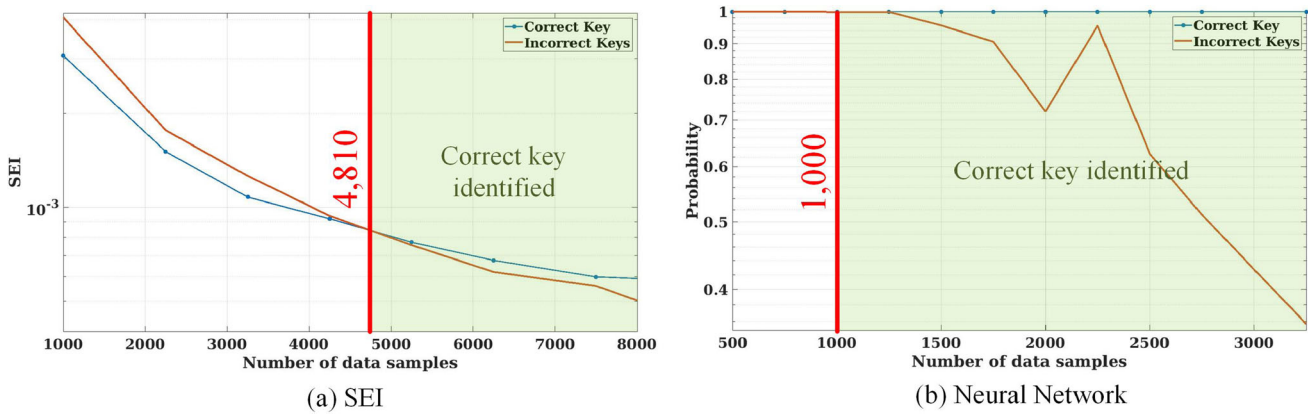


Fig. 15 Rank of key candidates for 5 equidistant clock period in the range $T^* \in [2.5, 3.5]$ ns, and using only correct values; **a** classical SEI metric, **b** neural network

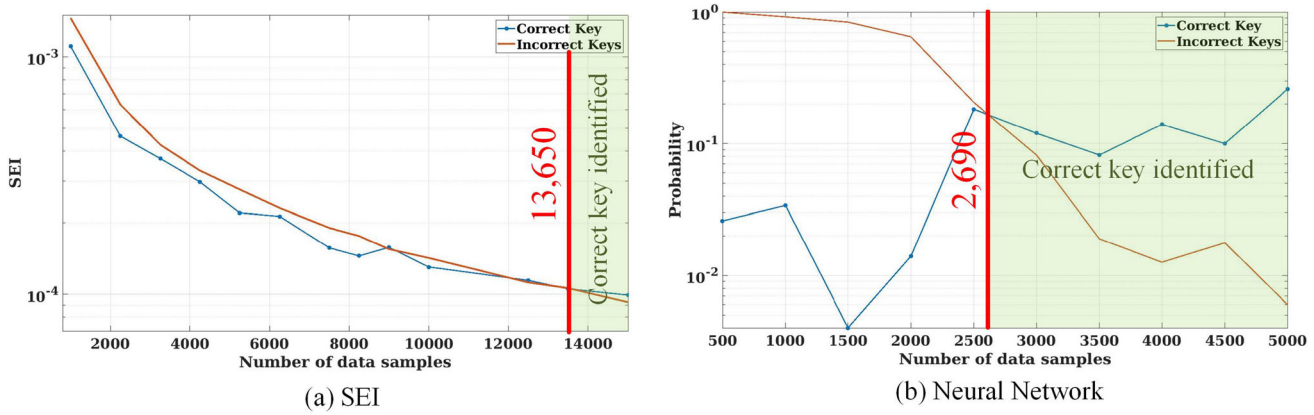


Fig. 16 Rank of key candidates for 5 equidistant clock period in the range $T^* \in [2.5, 3.5]$ ns, with infective countermeasure; **a** classical SEI metric, **b** neural network

and requires a profiling step to identify the correct ciphertext corresponding to every input plaintext.

The rank of key candidates in the presence of infective countermeasures is shown in Fig. 16. In this figure, all data, including the randomized faulty ciphertexts, are used to calculate the distribution of the intermediate variable. As shown in part (a) of the figure, with the SEI metric, at least 13,650 ciphertexts are required to identify the correct key candidate. However, as part (b) shows, the CNN key distinguisher identifies the correct key with only 2,690 faulted encryptions.

The efficiency of FIMA-NN is thus around 5 times better than bias-alone techniques.

8 Conclusions

We developed a generic fault model based on timing failure attacks that can reveal the vulnerability of hardware implementations using simulations with standard EDA tools. We augmented fault intensity map analysis (FIMA) with a convolutional neural network, called FIMA-NN, with a training set generated from the proposed fault model. We demonstrated that FIMA-NN is successful even in the presence of most existing countermeasures, although some countermeasures can degrade the efficiency of the attack. Using transient time analysis on a physical layout of an AES S-box in TSMC 65 nm technology for simulating timing failure attacks, we showed that fault injection analysis techniques which rely on bias alone require 12.6× more data than FIMA-NN to recover the 10th round key K^{10} using unfiltered output ciphertexts. Using only fault-free output ciphertexts, FIMA-NN is 4.8× more efficient than the biased-based techniques. Also, in the presence of infective countermeasures, FIMA-NN is 5× more efficient.

Acknowledgements This research was supported by NIST Award 70NANB18H219 for Lightweight Cryptography in Hardware and Embedded Systems.

References

1. 197, F.I.P.S.P.: Advanced Encryption Standard (AES) (2001)
2. Agoyan, M., Dutertre, J.M., Naccache, D., Robisson, B., Tria, A.: When clocks fail: On critical paths and clock faults. In: International Conference on Smart Card Research and Advanced Applications, pp. 182–193. Springer (2010)
3. Azzi, S., Barras, B., Christofi, M., Vigilant, D.: Using linear codes as a fault countermeasure for nonlinear operations: application to AES and formal verification. J. Cryptogr. Eng. **7**(1), 75–85 (2017)
4. Bae, H.J., Kim, C.W., Kim, N., Park, B., Kim, N., Seo, J.B., Lee, S.M.: A perlin noise-based augmentation strategy for deep learning with small data samples of hrct images. Sci. Rep. **8**(1), 1–7 (2018)
5. Barenghi, A., Bertoni, G.M., Breveglieri, L., Pelosi, G.: A fault induction technique based on voltage underfeeding with application to attacks against aes and rsa. J. Syst. Softw. **86**(7), 1864–1878 (2013)
6. Canright, D.: A Very Compact Rijndael S-box. Defense Technical Information Center (2005). https://apps.dtic.mil/docs/citations/ADA434781
7. Dobraunig, C., Eichlseder, M., Groß, H., Mangard, S., Mendel, F., Primas, R.: Statistical ineffective fault attacks on masked AES with fault countermeasures. In: International Conference on the Theory and Application of Cryptology and Information Security, pp. 315–342. Springer (2018)
8. Dobraunig, C., Eichlseder, M., Korak, T., Mangard, S., Mendel, F., Primas, R.: SIFA: Exploiting ineffective fault inductions on symmetric cryptography. IACR Trans. Cryptogr. Hardw. Embedded Syst. 547–572 (2018)
9. Faranda, D., Lucarini, V., Turchetti, G., Vaienti, S.: Numerical convergence of the block-maxima approach to the generalized extreme value distribution. J. Stat. Phys. **145**(5), 1156–1180 (2011)
10. Ghalaty, N.F., Yuce, B., Taha, M., Schaumont, P.: Differential fault intensity analysis. In: 2014 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC), pp. 49–58. IEEE (2014)
11. Kang, M.J., Kang, J.W.: Intrusion detection system using deep neural network for in-vehicle network security. PloS one **11**(6), e0155781 (2016)
12. Kermani, M.M., Jalali, A., Azarderakhsh, R., Xie, J., Choo, K.K.R.: Reliable inversion in $GF(2^8)$ with redundant arithmetic for secure error detection of cryptographic architectures. IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. **37**(3), 696–704 (2018)
13. Kim, C.H.: Differential fault analysis against aes-192 and aes-256 with minimal faults. In: 2010 Workshop on Fault Diagnosis and Tolerance in Cryptography, pp. 3–9. IEEE (2010)
14. Kolosnjaji, B., Demontis, A., Biggio, B., Maiorca, D., Giacinto, G., Eckert, C., Roli, F.: Adversarial malware binaries: evading deep learning for malware detection in executables. In: 2018 26th European Signal Processing Conference (EUSIPCO), pp. 533–537. IEEE (2018)
15. Kondor, R., Trivedi, S.: On the generalization of equivariance and convolution in neural networks to the action of compact groups. Preprint arXiv:1802.03690 (2018)
16. Kwon, D., Kim, H., Kim, J., Suh, S.C., Kim, I., Kim, K.J.: A survey of deep learning-based network anomaly detection. Cluster Comput. 1–13 (2017)
17. Lashermes, R., Reymond, G., Dutertre, J.M., Fournier, J., Robisson, B., Tria, A.: A DFA on AES Based on the Entropy of Error Distributions. In: 2012 Workshop on Fault Diagnosis and Tolerance in Cryptography, pp. 34–43. IEEE (2012)
18. Li, W., Liao, L., Gu, D., Li, C., Ge, C., Guo, Z., Liu, Y., Liu, Z.: Ciphertext-only fault analysis on the LED lightweight cryptosystem in the Internet of Things. IEEE Trans. Depend. Secure Comput. (2018)
19. Li, W., Zhang, W., Gu, D., Cao, Y., Tao, Z., Zhou, Z., Liu, Y., Liu, Z.: Impossible differential fault analysis on the LED lightweight cryptosystem in the vehicular ad-hoc networks. IEEE Trans. Depend. Secure Comput. **13**(1), 84–92 (2016)
20. Li, Y., Sakiyama, K., Gomisawa, S., Fukunaga, T., Takahashi, J., Ohta, K.: Fault sensitivity analysis. In: International Workshop on Cryptographic Hardware and Embedded Systems, pp. 320–334. Springer (2010)
21. Liu, Y., Cui, X., Cao, J., Zhang, X.: A hybrid fault model for differential fault attack on AES. In: 2017 IEEE 12th International Conference on ASIC (ASICON), pp. 784–787. IEEE (2017)
22. Ordas, S., Guillaume-Sage, L., Maurine, P.: Electromagnetic fault injection: the curse of flip-flops. J. Cryptogr. Eng. **7**(3), 183–197 (2017)
23. Patranabis, S., Chakraborty, A., Nguyen, P.H., Mukhopadhyay, D.: A biased fault attack on the time redundancy countermeasure for AES. In: International workshop on constructive side-channel analysis and secure design, pp. 189–203. Springer (2015)
24. Piret, G., Quisquater, J.J.: A differential fault attack technique against spn structures, with application to the aes and khazad. In: International Workshop on Cryptographic Hardware and Embedded Systems, pp. 77–88. Springer (2003)
25. Ramezanpour, K., Ampadu, P., Diehl, W.: Fault intensity map analysis with neural network key distinguisher. In: Proceedings of the 3rd ACM Workshop on Attacks and Solutions in Hardware Security Workshop, pp. 33–42 (2019)
26. Ramezanpour, K., Ampadu, P., Diehl, W.: FIMA: Fault intensity map analysis. In: Constructive Side-Channel Analysis and Secure Design, pp. 63–79. Springer (2019)

27. Ramezanpour, K., Ampadu, P., Diehl, W.: RS-Mask: Random space masking as an integrated countermeasure against power and fault analysis. Preprint arXiv:1911.11278 (2019)
28. Ramezanpour, K., Ampadu, P., Diehl, W.: SCAUL: Power side-channel analysis with unsupervised learning. Preprint arXiv:2001.05951 (2020)
29. Reshma, K., Priyatharishini, M., Devi, M.N.: Hardware trojan detection using deep learning technique. In: Soft Computing and Signal Processing, pp. 671–680. Springer (2019)
30. Schellenberg, F., Finkeldey, M., Gerhardt, N., Hofmann, M., Moradi, A., Paar, C.: Large laser spots and fault sensitivity analysis. In: 2016 IEEE International Symposium on Hardware Oriented Security and Trust (HOST), pp. 203–208. IEEE (2016)
31. Schneider, T., Moradi, A., Güneysu, T.: ParTI–towards combined hardware countermeasures against side-channel and fault-injection attacks. In: Annual International Cryptology Conference, pp. 302–332. Springer (2016)
32. Singh, A., Kar, M., Chawla, N., Mukhopadhyay, S.: Mitigating power supply glitch based fault attacks with fast all-digital clock modulation circuit. In: 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 19–24. IEEE (2019)
33. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(1), 1929–1958 (2014)
34. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–9 (2015)
35. Tunstall, M., Mukhopadhyay, D., Ali, S.: Differential fault analysis of the advanced encryption standard using a single fault. In: IFIP International Workshop on Information Security Theory and Practices, pp. 224–233. Springer (2011)
36. Van Erven, T., Harremos, P.: Rényi divergence and Kullback-Leibler divergence. IEEE Trans. Inf. Theory **60**(7), 3797–3820 (2014)
37. Yuce, B., Ghalaty, N.F., Santapuri, H., Deshpande, C., Patrick, C., Schaumont, P.: Software fault resistance is futile: effective single-glitch attacks. In: 2016 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC), pp. 47–58. IEEE (2016)