Noname manuscript No.
(will be inserted by the editor)

Attacks in Reality: The Limits of Concurrent Error Detection Codes against Laser Fault Injection

Jakub Breier · Wei He · Dirmanto Jap · Shivam Bhasin · Anupam Chattopadhyay

Received: date / Accepted: date

Abstract As a prominent attack approach against the security modules of integrated circuits, fault injection attacks (FIA) are able to breach the cryptographic primitives by analyzing the intentionally induced computation errors by adversaries. Parity-based Concurrent Error Detection (CED) techniques are often deployed as a countermeasure, owing to their low-overhead. Advanced linear and non-linear randomized encodings can be employed for constructing varying CED schemes.

In this paper, we first evaluate the detection capability of linear parity-protected ciphers implemented in commercial FPGA, using laser fault injection (LFI) technique. A single bit linear parity scheme is shown to be ineffective for error detection, since the LFI can typically flip multiple bits that are close to each other. On the other hand, a linear randomized parity scheme, with multiple bits parity, shows higher detection rates. Further, we study existing (randomized) non-linear encoding-based CED. With practical fault distributions on PRESENT cipher, non-linear randomized codes are extensively tested against fault injection. Although, known to have better theoretical detection bounds, non-linear encodings do not provide much improvements over simple randomized linear codes.

Keywords Hardware Security, Cryptography, Laser Fault Injection, Fault Detection, Bit-Flip

J. Breier, D. Jap and S. Bhasin
Physical Analysis and Cryptographic Engineering
Temasek Laboratories at Nanyang Technological University
Singapore
E-mail: {jbreier,djap,sbhasin}@ntu.edu.sg

W. He
Shield Lab, Central Research Institute
Huawei International Pte. Ltd.
Singapore
E-mail: hewei48@huawei.com
*The research was conducted when author was with Temasek Laboratories

A. Chattopadhyay
School of Computer Science and Engineering
Nanyang Technological University
Singapore
E-mail: anupam@ntu.edu.sg

1 Introduction

As a key driving factor for modern information technology evolution, cryptographic security in critical modules has gained immense importance, particularly because of massively distributed mobile telecommunication services, e-commerce, infrastructural security management and sensor networks. Confidential data is gathered, transmitted, processed and stored in distributed hardware devices, making them attractive targets for deliberate attacks on both software and hardware layers.

Security goals are typically realized by deploying modern cryptography in a form of proven algorithms and protocols. However, bad implementation of these algorithms and protocols can lead to serious exploits. Conventional security research mainly focuses on the cyber layer, whereas hardware layer vulnerabilities are neglected [27–29, 38]. Physical attacks are a set of exploits which target poor implementations of cryptography to compromise system security. Embedded devices are particularly vulnerable against attacks directly on lower level of hardware abstractions, such as power or EM based Side-Channel Attack (SCA) [2, 20], Hardware Trojan Horses (HTHs) [33], and Fault Injection Attacks (FIA) [7]. Substantial motivations arise for securing hardware layers, as summarized in the following.

– Embedded intelligent devices in system terminals are deployed remotely, portably and typically pow-

ered by batteries, so they are usually constructed using low-power/low-security devices considering the overall costs. Especially, the unattended endpoints are convenient for the adversary to perform close manipulations.
– Embedded system is basically an evolving concept that merges new disciplines as its expanded functional parts. Security strategies are hence required to be consistently upgraded in time. As a matter of fact, thorough security coverage of all the nodes, particularly the newly integrated and tiny sensor endpoints, becomes challenging.

Active physical attacks or fault injection attacks (FIA) [9], exploit erroneous behavior due to intentional injection of computation errors. The errors can be injected by disturbing one or several physical parameters including voltage, temperature, clock etc. Hardware Trojans [3,11,32] have been introduced as another kind of physical attack, by introducing malicious modifications in the circuit hardware. They can be designed to act passively (leak information) or actively (induce errors). When errors are successfully injected during the computation, the error is propagated to the output.

Many techniques have been proposed against the fault attacks [5,16,23]. Since the embedded devices are usually restricted by the limited power supply and insufficient computation power in the remote nodes, expensive countermeasures are often not applicable. To protect against faults in resource-constrained environments, the default choice for designers is Concurrent Error Detection (CED) and correction which stem from communication theory. CED is widely used by VLSI design and testing community for applications like memory testing [34], fault tolerance [12] etc. The simplest form of CED is *parity* [13]. It has also been used in context of fault injection attacks [35]. One of the first works, dealing with block cipher AES, applied a single parity on the whole 128-bit block [36]. This was further improved by using a single parity bit per byte or word [25,26]. Moreover, non-linear codes have also been applied in this context to achieve higher detection coverage [17]. Recently, a system-on-chip architecture was proposed [37] to detect and prevent a hardware Trojan insertion. As Trojan can maliciously modify sensitive data, its impact can be considered similar to fault injection [10]. Thus, the proposed solution relies on *randomized linear parity prediction* to detect any data modification by a malicious Trojan in hardware. Although the tested implementation deployed linear CED techniques, it was claimed that randomized non-linear codes can achieve better detection capabilities.

A successful parity detection against the laser fault attacks heavily depends on capabilities of the adversary to trigger even number of faults in the target device. If the chance of triggering a single fault or odd number of faults is high, the parity scheme provides acceptable protections. However, previous works mostly assess the achieved security from a purely theoretical standpoint, *i.e.,* the practical implementation situations in real devices are ignored. In our work, we explore a laser FIA on FPGA and investigate the probabilities of triggering different number of register bit flips using the pulse laser, to claim the vulnerabilities of the previously proposed parity schemes. Furthermore, we provide insights on different characteristics of linear and non-linear randomized parity codes under fault injection attacks. Most importantly, we show that non-linear randomized parity codes do not perform better than linear codes. Thus, given the implementation overhead of these advanced protection codes, the linear codes are a reasonable choice for designers.

The main contributions of this paper are listed as follows:

– We performed fault injection attacks using diode pulse laser against the parity protected lightweight PRESENT-80 cipher in Xilinx FPGA, which revealed possible fault models and distributions in a real chip. We used the results to evaluate the detectability of the induced computation errors in the target parity-enhanced cipher.
– We investigated a variety of encoding solutions: linear and non-linear randomized parity by simulations on PRESENT-80 cipher. Based on the fault models and distributions observed from the practical LFI experiment, we evaluated the vulnerabilities of the schemes, and found the most suitable choice regarding both security and cost.
– A series of security strategies are provided, *w.r.t.* presented implementation vulnerabilities in hardware scenarios, to mitigate the security pitfalls of parity-enhanced ciphers.

Paper Structure: Section 2 recalls the technical background, mainly the simple and the advanced parity protection schemes for block ciphers, and the implementation principles in hardware. Section 3 presents the chip preparation work and the principle of bypassing a parity-protected cipher. The setup of the experimental platform for performing the laser fault injection is also described in this section. The practical attack experiments for characterizing the fault mechanism in the parity-1 enhanced cipher implemented in Virtex-5 FPGA are detailed in Section 4. In Section 5, the simulations of fault analysis against the advanced parity schemes are described and compared, based on the properties observed from the practical LFI attack. The conclusions of this paper are drawn in Section 6.

2 Technical Background

This section recalls background concepts used in the rest of the paper. It briefly discusses PRESENT block cipher that was used in our experiments and provides details on previously introduced concurrent error detection schemes.

2.1 PRESENT Lightweight Cipher

PRESENT [8] is a widely adopted lightweight block cipher that is suitable for resource-constrained hardware environments. Structure of PRESENT cipher from algorithmic level is shown in Fig. 1, which is constructed following the substitution- permutation network (SPN). Block size of PRESENT is 64 bits and the key size can be either 80 or 128 bits (denoted by PRESENT-80 and PRESENT-128, respectively). The non-linear Sbox is 4 bits. The entire encryption/decryption consists of 31 rounds, and the last round key is used as a post-whitening key. This cipher is a part of the international standard ISO/IEC 29192-2:2012 [1].

Two fault attack models towards PRESENT-80 have been described by Bagheri *et al.* [4]. In the first model, a single bit fault is required to be injected into the intermediate state at the beginning of the last round of the *Sbox* layer. In the second model, a fault needs to be injected into one nibble of the *Sbox*.

This work explores the possibility of injecting one, two, three and four bit-flips into slice registers of 65 nm FPGA, targeting a parity protection scheme. It helps in evaluating the effectiveness of parity schemes against an attacker with strong and precise laser equipment.

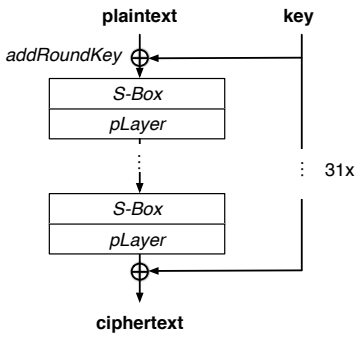

Fig. 1: PRESENT cipher algorithmic diagram.

2.2 Parity-based Concurrent Error Detection

Generally, a fault detection capability is achieved at the expense of either time or space redundancy. In the case of time redundancy, errors are detected by repeating the encryption process. No extra logic is required, but the throughput is reduced to around 50%. Parity is one of the simplest solutions when it comes to error detection based on space redundancy. It is heavily used in communication systems. When implemented in block ciphers, it can detect abnormalities during en-/decryption, while allowing efficient implementation, which makes it a good candidate for small embedded systems with low computational power. Different parity schemes have been proposed in literature aiming at covering different fault models, while keeping the cost low [6, 25, 26, 36].

The parity techniques can be implemented by two general approaches [14]:

- Parity-1: Only 1 parity bit is required for all the bits of datapath in each round of the cryptographic algorithm, *e.g.,* one parity bit checks the errors for all the bits of the data vector, such as 1-bit parity for the 128 bits in AES-128 [36].
- Parity-n: n parity bits are employed, and each parity bit is responsible for the error checking of a single data block in the cryptographic algorithm, *e.g.,* Parity-16 implements 1 parity bit per byte of AES-128 [25, 26].

Nevertheless, both of the above schemes can only detect odd number of faults occurred in the 128 data bits in Parity-1, or in the data word for Parity-n. In other words, if even number of faults appears, the schemes will be compromised. Another scheme, proposed by Karpovsky *et al.* [17] provides wider fault coverage, relying on a prediction circuit comprised of linear predictor, linear compressor and cubic function. Despite the uniform detection of both odd and even number of faults, the circuit overhead is too high to be applied in resource constrained scenarios. Considering the implementation efficiency and the fact that majority of fault models aims at single bit-flips, we hereby focus on the Parity-1 scheme in our work, and the conclusions directly apply to Parity-n as well.

2.3 Parity-1 Detection Scheme

There are several works proposing the usage of parity for fault detection [19, 24, 35], showing that despite not being able to detect more complex fault models, its simplicity still attracts attention. Fig. 2 depicts the parity

detection scheme proposed in [36], targeting AES. Actually it is universal to all Secret Key Cryptography (SKC) constructed by Substitution-Permutation Networks (SPNs). The round inputs are denoted as `X`. The round key `K` is added with `X` to produce `Y`. The nonlinear substitution box (`Sbox`) substitutes `Y` by `Z`. The linear diffusion layer permutes the bits of `Z` to give `U`, that is actually `X` for the next round. This parity prediction mainly consists of three computations. For clarity, parity of a bit vector $\cdot$ is denoted as $\mathrm{P}(\cdot)$. The computation then goes as follows:

- In key addition, the round input parity P(X) is XORed with round key parity P(K) to get P(Y).
- In Sbox, P(Y) is non-linearly changed. Since Sbox in SPN cipher is fixed and public, the Sbox output parity P(Z) can be precomputed. To check the integrity of the processed data, the input and output parity can be combined P(Y)⊕P(Z) and precomputed as an extension of a standard Sbox.
- The linear diffusion layer simply permutes the bits, so the parity is not changed in this step.

This process is depicted in Fig. 2.

$$P(Y) \oplus (P(Y) \oplus P(Z)) = P(Z) = P(out) \tag{1}$$

If no odd number of errors occurs:

$$P(out) = P(U). \tag{2}$$

Otherwise,

$$P(Y) \oplus (P(Y^*) \oplus P(Z)) = P(out) \neq P(U), \tag{3}$$

where P(Y*) represents the error-infected bit vector Y. Similarly to key addition part,

$$P(Y) = P(X) \oplus P(K). \tag{4}$$

Since P(X) is the P(out) of the previous round, we get:

$$P(Y) = P(out) \oplus P(K). \tag{5}$$

By checking if P(Y) is equal with $P(out) \oplus P(K)$, we can detect the encryption faults caused by the odd number of errors, in key addition in current round, or errors in Sbox or linear diffusion layer in previous round.

2.4 Advanced CED Techniques

We further investigate the fault coverage of more advanced CED techniques. To recall, there are many different types of possible faults that could be injected in the hardware. To limit the scope of our investigation, we only consider fault models that are exploitable. The most commonly used fault model for fault analysis in

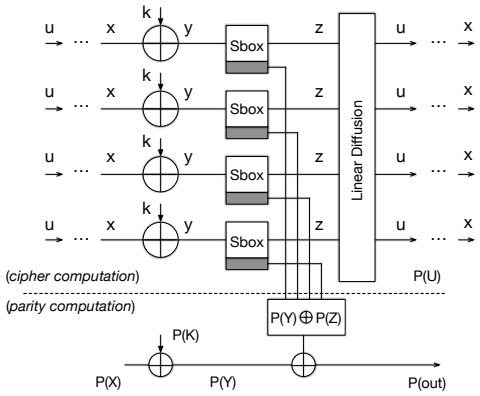

Fig. 2: Parity based concurrent error detection in SPN block cipher.

the literature is the bit(s)-flip, where one or several bit values are inverted. It could be formulated as $e = x \oplus x^*$, where $x \in \mathbb{F}_2^n$ is the original data, and $x^* \in \mathbb{F}_2^n$ is the faulty data. The number of bit flips could be defined as a Hamming weight-$HW(e)$ (or number of '1's) in the e.

2.4.1 Linear Codes and Non-linear Codes

A linear code of length n, rank k is a linear subspace C of $\mathbb{F}_2^n$, and the vectors in C are called *codewords*. As a linear subspace over $\mathbb{F}_2^n$, the code C sometimes could be represented as the span of basis codewords from the rows of the generator matrix G, which has a standard form $G = [I_k|A]$. This sort of coding is called systematic encoding, and it has the original data present in the codeword. Here, I_k is a square identity matrix with dimension k, and A is $k \times (n-k)$ matrix. Hence, any codeword $y \in C$ can be written as $y = xG$, where $x \in \mathbb{F}_2^k$ is the message with dimension k.

A parity check matrix H for (n,k)-linear encoding is a matrix where $Hy^T = 0 \Leftrightarrow \exists x \in \mathbb{F}_2^k$ such that $xG = y$. If G is a generating matrix in standard form, then H can be written as $[P|I_r]$ (called linear systematic code), where $P = A^T$ is a $(n-k) \times k$ matrix and I_r is a square identity matrix with dimension r.

For non-linear codes, we consider the cubic codes proposed in [18] as well as the inverse code proposed in [21]. As mentioned earlier, for a linear code, the codeword y could be rewritten as $y = xG = [xI|xA] = [x|(Px^T)^T]$. For simplicity, since it is a vector, we could write it as $(x,(Px))$. In [18], it is shown that for binary code, the construction of cubic code is $C_V = \{(x,w)|x \in \mathbb{F}_2^k, w = (Px)^3 \in \mathbb{F}_2^r\}$. In [21], the construction for the inverse code is $C_V = \{(x,w)|x \in \mathbb{F}_2^k, w = (Px)^{-1} \in \mathbb{F}_2^r\}$ instead, with $0^{-1} = 0$.

2.4.2 Randomized Parity Code

One example of a code is the parity code, defined as parity of a subset of the vector of length k, *i.e.*, $y = a_1x_1 \oplus \cdots \oplus a_kx_k$, where $a_i \in \{0, 1\}$. Here, we have $n = k+1$. The (n, k) parity (linear) encoding is then defined as $g : x \to y$ $(x \in \mathbb{F}_2^k, y \in \mathbb{F}_2^n)$, where $n = k + r$, and r is the number of parity bits.

The construction of randomized parity codes, as defined in [37], is as follows: From the set of all (n, k)-linear systematic parity codes $R^{n\times k}$, with all rows and columns in the parity check matrix being non-zero, we sample uniformly at random. The code that has been sampled is called *randomized parity code*. The randomization is done in order to prevent the adversary from knowing the parity check matrix, while the non-zero constraint is to prevent the zero function as well as to ensure that each bit is included in the parity function.

The method used to obtain the randomized parity code can be described as: Given the dimension k, parity bit r, output the randomized parity check matrix by choosing uniformly at random, a parity matrix H. If it satisfies the construction described earlier, output H, otherwise repeat and choose another parity matrix.

Fig. 3 shows a system, proposed recently by Wu *et al.* [37], which is claimed to be robust against hardware Trojans. For every internal component in the system, the logic, memory and the communication architecture, the corresponding CED techniques are elaborated. In this work, we particularly focus on the randomized parity encoding technique that was proposed to be part of the memory and communication architecture (bus) protection. In Fig. 4, it is shown how to perform the selection of the bits for randomized parity [37]. It was argued that the detection rate could be higher if non-linear code is used at the expense of additional area. For non-linear randomized code, the parity matrix H is chosen similarly, following the construction of randomized parity code described earlier. However, the parity matrices are based on non-linear construction instead.

From the algorithmic point of view, the parity scheme can be implemented across different bits. In this case, the parity can be calculated on either 128, 64, 32, 16, 8 or 4 bits data. As shown in the experiments later, we implemented the parity computation over 32 bits (for AES, adjusted to the MixColumn) and 16 bits (for PRESENT, adjusted to the pLayer). One consideration is that, for smaller bit size, it might improve the detection rate in case of a localized fault. However, in this case, the area required to store the parity bits increases as well. For example, depending on the architecture, it will require additional register(s) or memory to store the additional parity bits (since the parity scheme is

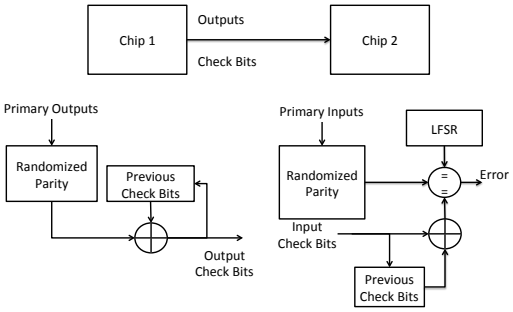

Fig. 3: Hardware Trojan detection based on randomized parity codes proposed in [37].

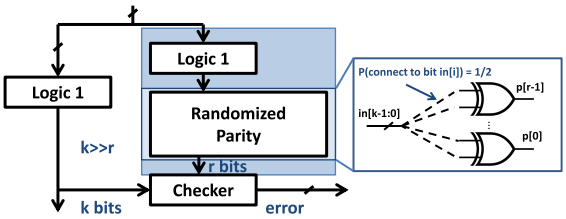

Fig. 4: Bit selection for randomized parity [37].

covering fewer bits, more parity bits are required to handle the data). Moreover, for the randomized parity, it requires randomness to select each individual parity check matrix, and thus, adds additional cost in terms of random number generation.

3 Chip Preparation and Profiling

3.1 Precision Necessity for Disrupting Parity Check

When it comes to practical fault attacks, one of the most precise techniques is the laser fault injection. This technique was traditionally used for failure analysis and was introduced for disturbing cryptographic circuits by Skorobogatov *et al.* in [31]. The laser impact on circuit can either be a bit-set/bit-reset, or it can introduce a delay in the signal propagation in routings to cause set-up time violation [30]. Due to requirements of most cryptographic fault models, bit-flips should be precisely injected in the desired bit(s) at the specific computation point. The power supply and clock disturbances normally cause unpredictable multiple faults over the entire affected logic network. Hence, laser-induced fault attacks provide the option with the most precise controllability.

To successfully perform a laser fault injection, following parameters need to be determined: (1) the in-

tensity and duration of the laser pulse; (2) the time delay from the trigger to laser activation (*i.e.* propagation delay); (3) the location where the interesting logics are deployed on the chip; (4) the penetration depth of the laser source and the effective laser spot size. In our work, a Xilinx Virtex-5 FPGA is selected as the implementation device for the parity protected cryptography. The generic structure of Xilinx FPGA mainly consists of an array of fundamental logic cells, called Configurable Logic Blocks (CLBs). In each CLB, there are two slices. Each slice contains 4 Look-Up-Tables (LUTs), multiplexers and flip-flops. In the implementation, 4 flip-flops in each slice can be configured as 4-bit registers, to store a nibble of round computation. Since 4 flip-flops are pre-fabricated into a single slice, the actual distances between them are extremely small, typically within several hundreds nm in 65 nm devices. To avoid the change in detection parity, adversaries need to upset either 2 or 4 bits simultaneously, for an odd-bit parity detection scheme.

3.2 Experiment Setup

The laser setup for our work is shown in Fig. 5. The embedded device is emulated by a Xilinx Virtex-5 FPGA in 65 nm technology with a flip-chip package. Since the substrate of the die inside the package is placed upside, the chip can be preprocessed by a mechanical solution for reducing the substrate thickness, in our case from $\approx$ 300 μm to $\approx$ 100 μm. Note that a thinner residual substrate leads to easier laser penetration, but it increases the risk of destroying logic resources or routing channels on the chip. The laser utilized is a 20 W diode pulse laser with 5$\times$ magnification lens (reducing the effective maximum power to 10 W). The wavelength is 1064 nm and the spot size of the laser beam is $\approx 60\times14$ μm^2. It is emphasized that only the very central part ($\approx$ 1/10) of the beam spot is effective mainly because of the laser refraction and energy absorption through the substrate. A PRESENT-80 coprocessor, enhanced with parity based error detection scheme (see Fig. 6), is implemented inside the FPGA. We compare the cipher output immediately after the key addition layer, together with the corresponding parity P(Y). The area overhead of the parity protected Sbox (prime target of the study) is shown in Table. 1.

The test platform is set up over a laser station, with a 3D motorized stage for high-precision chip scanning. Positioning step is $0.05\mu m$. The system is driven by a control software on the PC for performing the laser perturbation, operating the chip scan and recording the cipher output. By scanning the PRESENT-implemented region in FPGA and observing the faulty outputs, we have successfully located the position of the slice where the 4 flip-flops of the least significant nibble of the 64 bit data block of PRESENT-80 reside.

Table 1: Area overhead of parity protected Sbox.

Sbox	LUT	Flip-Flops
Unprotected	64	64
Parity-1 Protected	78	65
Overhead	21.75%	1.5%

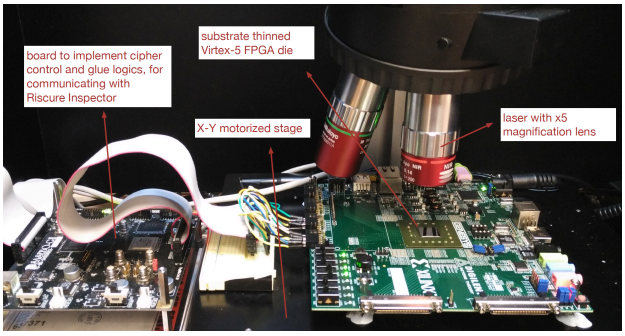

Fig. 5: Laser fault injection platform.

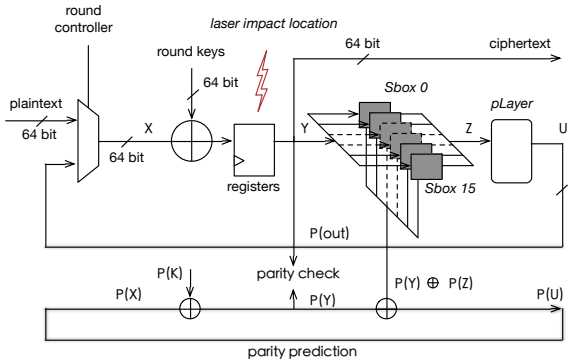

Fig. 6: A PRESENT-80 lightweight cipher implemented with parity base error detection scheme.

4 Characterization of Laser Bit-Flips on Parity-1 CED Scheme

4.1 Laser Fault Injection in FPGA Slice

We conducted a laser scan in the slice region where registers of bit 0, 1, 2, 3 are situated (the least significant nibble of the 64 bit data block). The purpose was to check the possibility of disturbing specific register bits in a nibble after the *addRoundKey* operation, as seen in

Fig. 6. Without loss of generality, we have targeted the last round of the cipher, with the results stated below. Register area sensitive to faults is $\approx 16{\times}7\ \mu m^2$ large. This is actually the exact location where the FPGA slice is placed, as can be seen in Fig. 7 from the `FPGA editor` view. We used the full laser power available for such lens type (10 W), with a variable glitch length between $250 - 300ns$.

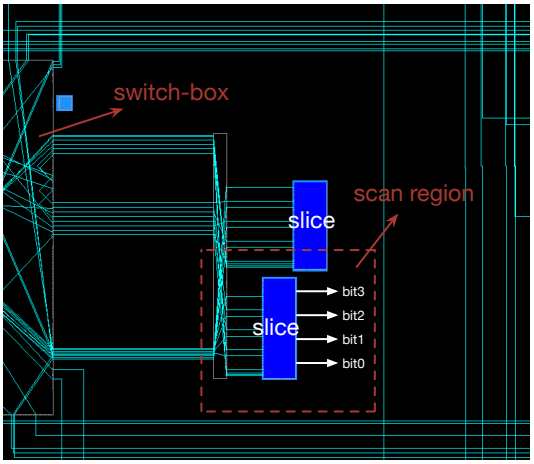

Fig. 7: A slice targeted for the laser scan, where the least significant nibble (bit{0-3}) of PRESENT-80 is placed.

The result from attacking the last round is shown in Fig. 8. Totally, 4947 faults have been recorded out of 12,000 scanned points. As can be seen in the plot, majority of faults are 4-bit flips (*i.e.,* the fault mask is `1111`, as represented by blue spots in Fig. 8), resulting to unchanged parity. The % of each faulty mask from all the faults are given in Table. 2. Note that the we are targeting the least significant nibble, *i.e.,* {bit0, bit1, bit2, bit3}, however the scan also reached the neighboring slice where 2^{nd} last nibble {bit4, bit5, bit6, bit7} resides (rightmost spots in Fig. 8). The experiments were performed using coarse-grained scan of the FPGA slice.

In total, 99.293% (1111:99.212% + 0101:0.081%) of all the faults are even bit flips in the target nibble, therefore the parity would stay unchanged with a high probability, but the ciphertext would be faulty. In this experiments, the attacker would be able to mount a fault attack with either 2-bit or 4-bit fault model, using a selected subset of the faulty outputs. Given a precise laser navigation to a desired slice, the attack could exclusively trigger the useful faulty outputs for specific fault models. Due to the nature of the scan, single bit flip was triggered with an extremely low probability of 0.1%. However, this attacker model is suitable for attacking the Parity-1 CED scheme.

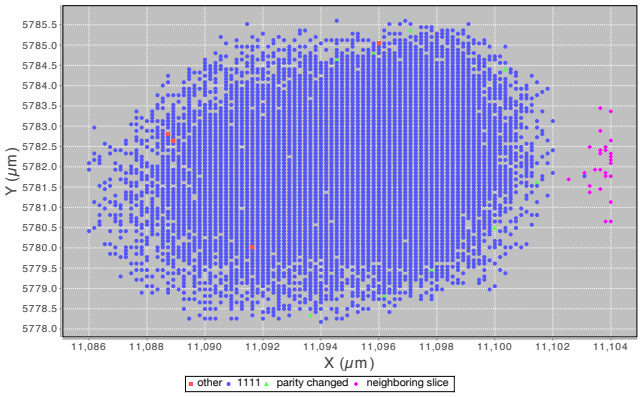

Fig. 8: Area plot showing laser-induced faults during the last *addRoundKey* operation.

Table 2: Faulty mask and appearance % for the register nibble of the last round fault perturbation.

nibble of `'target'` slice: {bit0, bit1, bit2, bit3}

mask	0001	0010	0011	0100	0101	0110	0111	1000
%	0.101	0	0	0	0.081	0	0.081	0
mask	1001	1010	1011	1100	1101	1110	1111	0000
%	0	0	0	0	0	0	99.212	×

nibble of `'neighboring'` slice: {bit4, bit5, bit6, bit7}

mask	0001	0010	0011	0100	0101	0110	0111	1000
%	0	0.020	0	0	0	0	0	0
mask	1001	1010	1011	1100	1101	1110	1111	0000
%	0	0.505	0	0	0	0	0	×

As a more permanent and dependable solution, complex codes are a fair alternative. The CED can be made robust by varying different parameters. For instance, one can choose a longer code which is still linear in construction, like parity, to have better detection rate but still low implementation overhead. On the other hand, non-linear codes are more complex in construction and thus have high implementation footprint, but they are believed to be more robust. Another solution is to use randomization with codes to limit the fault injection capability of the attacker. All these advanced CED techniques are discussed and analysed in detail in the rest of the paper.

5 Attack Simulation on Advanced CED Techniques

Based on the prior experimental analysis on the simple CED implementation, we discuss their behavior under fault injection of the advanced CED schemes. It describes the precise attacker model, the methodology deployed to simulate fault injection and the observed results.

Table 3: Experimental results from the FPGA slice scan targeting four round registers of PRESENT-80.

Fault model	% of faults
1-bit flip	57.25
2-bit flip	24.17
3-bit flip	15.19
4-bit flip	3.45

5.1 Attack Model

As previously discussed, fault injection attacks aim at injecting computation errors in the cipher execution. These errors, in general, follow a specific fault model which depends on the injection techniques, injection parameters and underlying target. Fault injection with the given model can be practically injected or simulated. Practical fault injection would represent non-uniformly distributed small subset of the simulations, preventing comprehension of overall trend. In contrast, simulation covers different fault scenarios, which could highlight the general trend, however, in case of practical setting, this might overestimate the trend, since most of the faults tend to be biased towards specific model. Thus, a combination of simulation and practical evaluation usually has to be considered for validation purposes.

5.2 Fine-Graned Slice Scan

In order to operate on real-world values in our simulations, we have performed a fine-grained scan of an FPGA slice. Unlike the scan from the previous section, which focused on providing the optimal result for breaking the even parity scheme, the scan in this section focused on targeting the whole slice area. The step size in this case was 0.5 μm and the glitch length was fixed to 282 ns. By this methodology, we could target specific registers and obtain single bit-flips. Such scan provided us with more appropriate results for evaluating the advanced parity schemes. Detailed results of this scan are stated in [15].

For profiling purposes, we have implemented block cipher PRESENT-80, where 4 of the total 64 round registers reside in the target slice. Out of 10,000 experiments, we received 3,918 faulty encryptions that were caused by flipping one or more bits in the registers. Percentages representing each bit-flip fault model are stated in Tab. 3.

5.3 Simulation Methodology

Simulations were performed in MATLAB, based on bit flip faults from Tab. 3. We varied the number of bit flips from 1 to n, where n is the data bit-width.

A multiple bit-flip fault was injected by performing a modulo-2 addition between the input x and a chosen fault mask m of the same bit-width. For small bit-width $n \leq 8$, combinations of x and m were chosen exhaustively. For bigger n, both x and m were chosen randomly from two independent uniformly distributed data for a representative number of scenarios. Thereafter, the detection rates were captured and plotted as shown in the following.

We simulated the case for the randomized encoding, based on the construction described in [37]. The length of message was 120 bits and the parity bits were varied from 3 bits to 8 bits. We run repeated experiments $(100,000\times)$ with randomly selected message, and for each, we randomly select the parity check matrix for the randomized encoding. As shown in Fig. 9(a), (c), the linear $(x, p(x))$ and inverse $(x, p(x)^{-1})$ encodings perform similarly. For cubic $(x, p(x)^3)$ encoding (Fig. 9(b)), the detection probability for even parity (r=4,6,8) is lower for small number of errors, which gradually improves when the number of bit-flips increases. For even parity ($r = 2q$), the cubic power mapping is not bijective. As the number of elements which is cube can be calculated by $\frac{2^r-1}{gcd(3,2^r-1)}$, for even r, $3|2^r - 1$, and thus, for lower number of error bits, the faulty predicted and calculated parity could have a collision, and thus the fault is undetected. As the number of erroneous bits increases, the fault could affect the parity bit as well which allow the detection. In general, it can be concluded that if the designer has the liberty to use multiple parity bits ($r \geq 5$), the performance of randomized encoding with linear or non-linear code is equivalent. For lower parity bits ($r \leq 4$), inverse stays similar to linear, while cubic can only outperform linear at a high number of bit-flips. In general, the detection is similar across different encoding schemes. This might be attributed to collisions due to the length of parity mapping. With longer parity, the effect of the collision could be reduced. However, considering the implementation perspective, it is common knowledge that non-linear encoding (cube or inverse) can be much more resource-consuming compared to basic linear encoding [34].

To improve the detection rate, authors in [37] suggest to introduce a *memory effect*. Memory effect means that instead of dealing with codewords individually, the current or i^{th} codeword is computed as a combination of all $(i-1)^{th}$ codewords. With the memory effect, it becomes difficult for the attacker to manipulate several

stages, thus improving the detection rate in practice. The simulation results for this scenario are shown in Fig. 9(d), (e), (f). The detection probability was improved even for smaller codewords ($r = 3, 4$). Moreover, all the parity encodings perform similar, supporting the linear encoding under implementation cost consideration.

5.3.1 Application to PRESENT Cipher

We applied the randomized encoding on PRESENT block cipher. As described earlier, the fault could be injected at any time, starting from the beginning of the round, so we simulated the fault propagation during one round of a complete PRESENT. Since a PRESENT round could be divided into 4 groups of 4 nibbles (based on the properties of the `Sbox` and `pLayer` operations), we considered 16 bits input and output. The parity bits r were varied from 2 to 8, under the memory effect, and $r = 1$ was ignored as it would be just a standard parity bit, for linear and non-linear encoding. The fault is then injected during round computation. The parity computed over the faulty output is computed and compared against the parity of the expected output. A match results in detection failure.

Based on the simulation results, as shown in Fig. 10, we can see that the fault detection rate for different encoding schemes is similar to each other. Note that in our experiments, for longer message bit length, it was not possible to simulate all potential fault masks in a reasonable time. Hence, rather than exhaustive simulations, we carried the simulations until the error became negligible.

5.4 Validation on PRESENT

Further investigation were done on full PRESENT, based on the previously charaterized fault model. As previously shown, a fine grain scan of the DUT allowed us to achieve fault models shown in Tab. 3. We consider a full round of PRESENT, computing 64-bits in parallel. The experiments were conducted using different parity lengths (4, 8 and 16 bits parity). The fault caused 1 - 4 bit flips on a single nibble, randomly chosen from the 16 nibbles. The experiments were repeated $100,000\times$. In Tab. 4, we show the detection accuracy of random bit-flip faults in a nibble for different parity schemes. It can be observed that the accuracy for inverse function is similar to the linear encoding, and the cube function is performing worse than the others. In general, for 8-bit parity or longer, the accuracy of the achieved detection is greater than 98%. Given that the detection capability of randomized non-linear codes was no better than the linear counterpart (Tab. 4), we did not proceed with a real implementation. It is known that implementation overhead of non-linear codes is significantly higher than linear codes, with similar detection capabilities. Therefore, linear codes stand as an obvious choice.

Table 4: Detection accuracy of different randomized parity schemes (%)

Linear randomized parity				
Parity/No. error	1	2	3	4
r = 4	93.79	95.76	96.45	97.04
r = 8	99.62	99.75	99.76	99.82
r = 16	100.00	100.00	100.00	100.00
Cube randomized parity				
Parity/No. error	1	2	3	4
r = 4	82.40	83.76	84.54	84.46
r = 8	98.81	99.07	99.09	98.95
r = 16	100.00	100.00	100.00	99.98
Inverse randomized parity				
Parity/No. error	1	2	3	4
r = 4	93.57	95.61	96.02	96.40
r = 8	99.61	99.72	99.82	99.84
r = 16	99.99	100.00	100.00	100.00

5.5 Discussion on Mitigation Solutions

Considering good design practices, even parity is an obvious solution. However, it does not eradicate the problem, only shifts the issue to different parameters like choice of fault injection technique or injection strength, etc. In the parity implementation in FPGA, a single slice was targeted for the evaluation, where 4 registers were used as the round registers in the PRESENT-80 data-path. This does not incur any loss of generality since commercial placement and routing tools deploy the bits of the same bit-vector close to each other, which is true for both ASIC and FPGA. This is owing to the requirement of area and timing optimizations in the late design phases. In our case, we noticed the 4 bits of a bit vector to be always located in the same slice. As a matter of fact, multiple bit flipping in registers or similar logics by a single laser injection is practical for the commercial chips [15].

Our experimental results show that flipping an even number of bits using a single laser injection can be done easily. Fault models based on this result also exist, such as the nibble fault model described in [4]. To circumvent these security vulnerabilities, a special attention should be paid during the implementation. It is highly recommended to carefully investigate the multiple bit-fault models against a specific cipher, and swap the

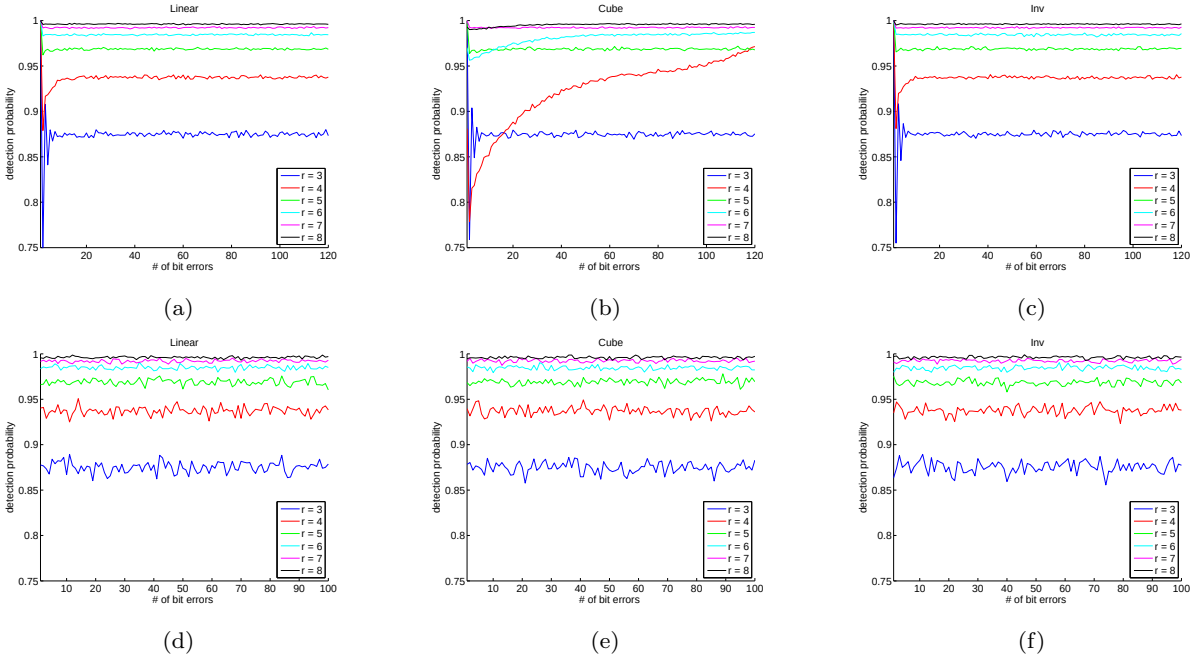

Fig. 9: Evaluation of randomized encodings: Linear vs Non-Linear. (a)-linear, (b)-cubic, (c)-inverse without memory effect, while (d), (e), (f) with implemented memory effect.

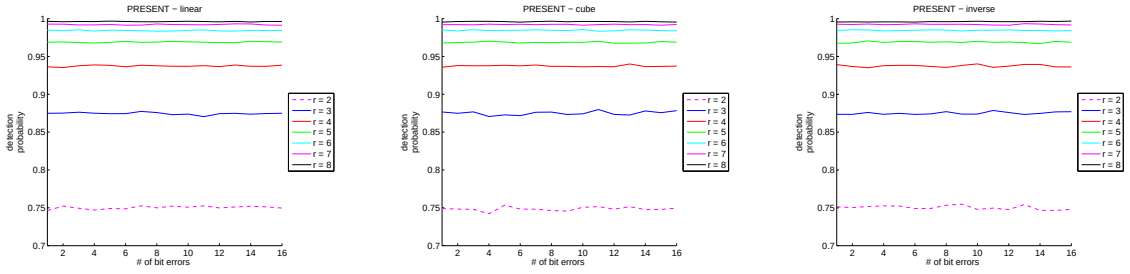

Fig. 10: Evaluation of PRESENT operations

unrelated bits of nibbles into different slices or deploy them far from each other during the placement phase of the FPGA or ASIC implementation.

The simulation of advanced (linear, non-linear, and randomized parity) schemes also shows vulnerabilities based on the attack result. Since the observations may differ significantly under different attack scenarios, it is necessary to thoroughly investigate the implemented devices and the possible attack vectors for choosing the adequate parity scheme and the implementation tactics.

Moreover, our experimental results were obtained by using a relatively budget-friendly laser station. As some other works show (e.g. [22]), by using more advanced setups, such as Hamamatsu PHEMOS-1000, the precision of the faults can be further improved.

6 Conclusions

Fault injection attacks have been widely studied in recent years due to the severe threats against the security-critical circuit systems. As a typical intrinsic countermeasure against fault injections, parity concurrent error detection (CED) is often utilized for detecting faults in hardware [37]. Parity bit(s) basically flag the alarm once the number of flipped bits is satisfied depending on the used parity scheme. Generally, basic and randomized (linear and non-linear) encodings can be employed for constructing varying parity solutions.

In this paper, we thoroughly studied the fault mechanism of circuit logic elements in FPGA environment, and performed a practical laser fault injection into a

single bit CED-protected block cipher in Xilinx Virtex-5 FPGA. Experimental results show a probability of 99.293% for flipping an even number of data bits by a single laser injection, which cannot be detected by the implemented countermeasure that exclusively detects odd number of faults, indicating the limitation of the approach. We then investigated the detection probability of randomized parity-protected cipher, comparing the linear and non-linear codes, based on experimental assumptions from previous tests. The results show that, unlike previously hypothesised, a non-linear randomized parity codes do not perform better than linear randomized parity codes, even with higher resource and performance overhead. Considering both the security and cost, linear randomized parity code is recommended as the applied parity protection strategy against fault injection attacks. Based on the experimental results we observed, we proposed implementation strategies that help to increase the fault detection efficiency.

In the future work, we plan to investigate the resistance of Error-Correcting Code (ECC) memories against the practical laser fault attack in an off-the-shelf server computer.

References

1. Iso/iec 29192-2:2012, information technology-security techniques-lightweight cryptography-part 2: Block cipher (2012) (2012)
2. Agrawal, D., Archambeault, B., Rao, J.R., Rohatgi, P.: The em side-channel (s). In: Cryptographic Hardware and Embedded Systems-CHES 2002, pp. 29–45. Springer (2003)
3. Anderson, M.S., North, C., Yiu, K.K.: Towards countering the rise of the silicon trojan. (2008)
4. Bagheri, N., Ebrahimpour, R., Ghaedi, N.: New differential fault analysis on present. EURASIP Journal on Advances in Signal Processing **2013**(1), 1–10 (2013)
5. Barenghi, A., Breveglieri, L., Koren, I., Naccache, D.: Fault injection attacks on cryptographic devices: Theory, practice, and countermeasures. Proceedings of the IEEE **100**(11), 3056–3076 (2012)
6. Bertoni, G., Breveglieri, L., Koren, I., Maistri, P., Piuri, V.: Error analysis and detection procedures for a hardware implementation of the advanced encryption standard. Computers, IEEE Transactions on **52**(4), 492–505 (2003)
7. Biham, E., Shamir, A.: Differential fault analysis of secret key cryptosystems. In: Advances in Cryptology-CRYPTO'97, pp. 513–525. Springer (1997)
8. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J., Seurin, Y., Vikkelsoe, C.: PRESENT: An ultra-lightweight block cipher. Springer (2007)
9. Boneh, D., DeMillo, R.A., Lipton, R.J.: On the Importance of Eliminating Errors in Cryptographic Computations. Journal of Cryptology **14**(2), 101–119 (2001)
10. Breier, J., He, W.: Multiple Fault Attack on PRESENT with a Hardware Trojan Implementation in FPGA. In: 2015 International Workshop on Secure Internet of Things (SIoT), pp. 58–64 (2015). DOI 10.1109/SIOT.2015.15
11. Force, T.: High performance microchip supply (2005)
12. Gaisler, J.: Concurrent error-detection and modular fault-tolerance in a 32-bit processing core for embedded space flight applications. In: Fault-Tolerant Computing, 1994. FTCS-24. Digest of Papers., Twenty-Fourth International Symposium on, pp. 128–130. IEEE (1994)
13. Gallager, R.: Low-density parity-check codes. IRE Transactions on information theory **8**(1), 21–28 (1962)
14. Guo, X., Mukhopadhyay, D., Karri, R.: Provably secure concurrent error detection against differential fault analysis. IACR Cryptology ePrint Archive **2012**, 552 (2012)
15. He, W., Breier, J., Bhasin, S., Jap, D., Ong, H.G., Gan, C.L.: Comprehensive laser sensitivity profiling and data register bit-flips for cryptographic fault attacks in 65 nm fpga. In: C. Carlet, M.A. Hasan, V. Saraswat (eds.) Security, Privacy, and Applied Cryptography Engineering: 6th International Conference, SPACE 2016, Hyderabad, India, December 14-18, 2016, Proceedings, pp. 47–65. Springer International Publishing, Cham (2016)
16. Joye, M., Tunstall, M.: Fault Analysis in Cryptography. Springer (2012)
17. Karpovsky, M., Kulikowski, K.J., Taubin, A.: Robust protection against fault-injection attacks on smart cards implementing the advanced encryption standard. In: Dependable Systems and Networks, 2004 International Conference on, pp. 93–101. IEEE (2004)
18. Karpovsky, M.G., Taubin, A.: New class of nonlinear systematic error detecting codes. IEEE Trans. Information Theory **50**(8), 1818–1820 (2004). DOI 10.1109/TIT.2004.831844. URL `http://dx.doi.org/10.1109/TIT.2004.831844`
19. Kermani, M., Reyhani-Masoleh, A.: Parity-based fault detection architecture of s-box for advanced encryption standard. In: Defect and Fault Tolerance in VLSI Systems, 2006. DFT '06. 21st IEEE International Symposium on, pp. 572–580 (2006). DOI 10.1109/DFT.2006.50
20. Kocher, P., Jaffe, J., Jun, B., Rohatgi, P.: Introduction to differential power analysis. Journal of Cryptographic Engineering **1**(1), 5–27 (2011)
21. Kulikowski, K.J., Karpovsky, M.G., Taubin, A.: Fault attack resistant cryptographic hardware with uniform error detection. In: L. Breveglieri, I. Koren, D. Naccache, J. Seifert (eds.) Fault Diagnosis and Tolerance in Cryptography, Third International Workshop, FDTC 2006, Yokohama, Japan, October 10, 2006, Proceedings, *Lecture Notes in Computer Science*, vol. 4236, pp. 185–195. Springer (2006). DOI 10.1007/11889700_17. URL `http://dx.doi.org/10.1007/11889700_17`
22. Lohrke, H., Scholz, P., Boit, C., Tajik, S., Seifert, J.P.: Automated detection of fault sensitive locations for reconfiguration attacks on programmable logic pp. 1–6 (2016)
23. Malkin, T.G., Standaert, F.X., Yung, M.: A comparative cost/security analysis of fault attack countermeasures. In: L. Breveglieri, I. Koren, D. Naccache, J.P. Seifert (eds.) Fault Diagnosis and Tolerance in Cryptography, *Lecture Notes in Computer Science*, vol. 4236, pp. 159–172. Springer Berlin Heidelberg (2006). DOI 10.1007/11889700_15. URL `http://dx.doi.org/10.1007/11889700_15`
24. Mozaffari-Kermani, M., Reyhani-Masoleh, A.: A lightweight concurrent fault detection scheme for the aes s-

boxes using normal basis. In: E. Oswald, P. Rohatgi (eds.) Cryptographic Hardware and Embedded Systems CHES 2008, *Lecture Notes in Computer Science*, vol. 5154, pp. 113–129. Springer Berlin Heidelberg (2008). DOI 10.1007/978-3-540-85053-3_8. URL http://dx.doi.org/10.1007/978-3-540-85053-3_8

25. Mozaffari-Kermani, M., Reyhani-Masoleh, A.: Concurrent structure-independent fault detection schemes for the advanced encryption standard. Computers, IEEE Transactions on **59**(5), 608–622 (2010)
26. Mozaffari-Kermani, M., Reyhani-Masoleh, A.: A lightweight high-performance fault detection scheme for the advanced encryption standard using composite fields. Very Large Scale Integration (VLSI) Systems, IEEE Transactions on **19**(1), 85–91 (2011)
27. Perkins, C., Muller, G.: Using discrete event simulation to model attacker interactions with cyber and physical security systems. Procedia Computer Science **61**, 221–226 (2015)
28. Sandberg, H., Amin, S., Johansson, K.: Cyberphysical security in networked control systems: An introduction to the issue. Control Systems, IEEE **35**(1), 20–23 (2015)
29. Schmittner, C., Ma, Z., Schoitsch, E., Gruber, T.: A case study of fmvea and chassis as safety and security co-analysis method for automotive cyber-physical systems. In: Proceedings of the 1st ACM Workshop on Cyber-Physical System Security, pp. 69–80. ACM (2015)
30. Selmane, N., Guilley, S., Danger, J.L.: Practical setup time violation attacks on aes. In: Dependable Computing Conference, 2008. EDCC 2008. Seventh European, pp. 91–96. IEEE (2008)
31. Skorobogatov, S., Anderson, R.: Optical Fault Induction Attacks. In: B. Kaliski, ç. Koç, C. Paar (eds.) Cryptographic Hardware and Embedded Systems - CHES 2002, *Lecture Notes in Computer Science*, vol. 2523, pp. 2–12. Springer Berlin Heidelberg (2003). DOI 10.1007/3-540-36400-5_2. URL http://dx.doi.org/10.1007/3-540-36400-5_2
32. Tehranipoor, M., Koushanfar, F.: Guest editors' introduction: Confronting the hardware trustworthiness problem. IEEE Design & Test of Computers **27**(1), 8–9 (2010)
33. Tehranipoor, M., Koushanfar, F.: A survey of hardware trojan taxonomy and detection (2010)
34. Wang, Z., Karpovsky, M., Kulikowski, K.J.: Design of memories with concurrent error detection and correction by nonlinear sec-ded codes. Journal of Electronic Testing **26**(5), 559–580 (2010)
35. Wen, L., Jiang, W., Jiang, K., Zhang, X., Pan, X., Zhou, K.: Detecting fault injection attacks on embedded real-time applications: A system-level perspective. In: 2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on Cyberspace Safety and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems, pp. 700–705 (2015). DOI 10.1109/HPCC-CSS-ICESS.2015.165
36. Wu, K., Karri, R., Kuznetsov, G., Goessel, M.: Low Cost Concurrent Error Detection for the Advanced Encryption Standard. In: In Proceedings of the IEEE International Test Conference (ITC 2004), pp. 1242–1248 (2004)
37. Wu, T.F., Ganesan, K., Hu, Y.A., Wong, H.P., Wong, S.S., Mitra, S.: TPAD: hardware trojan prevention and detection for trusted integrated circuits. IEEE Trans. on CAD of Integrated Circuits and Systems **35**(4), 521–534 (2016). DOI 10.1109/TCAD.2015.2474373. URL http://dx.doi.org/10.1109/TCAD.2015.2474373
38. Zhu, B., Joseph, A., Sastry, S.: A taxonomy of cyber attacks on scada systems. In: Internet of things (iThings/CPSCom), 2011 international conference on and 4th international conference on cyber, physical and social computing, pp. 380–388. IEEE (2011)