IACR Transactions on Cryptographic Hardware and Embedded Systems
ISSN 2569-2925, Vol. 2022, No. 3, pp. 438–462. DOI:10.46586/tches.v2022.i3.438-462

Beware of Insufficient Redundancy

An Experimental Evaluation of Code-based FI Countermeasures

Timo Bartkewitz*[1], Sven Bettendorf*[1], Thorben Moos*[2], Amir Moradi*[3] and Falk Schellenberg*[4]

[1] TÜV Informationstechnik GmbH, Essen, Germany
t.bartkewitz@tuvit.de,s.bettendorf@tuvit.de
[2] UCLouvain, ICTEAM, Crypto Group, Louvain-la-Neuve, Belgium
firstname.lastname@uclouvain.be
[3] University of Cologne, Institute for Computer Science, Cologne, Germany
firstname.lastname@uni-koeln.de
[4] Max Planck Institute for Security and Privacy, Bochum, Germany
firstname.lastname@mpi-sp.org

Abstract. Fault injection attacks pose a serious threat to cryptographic implementations. Countermeasures beyond sensors and shields usually deploy some form of redundancy to detect or even correct errors. A few years ago, a novel design methodology called *Impeccable Circuits* has been introduced on how to correctly integrate Concurrent Error Detection (CED) schemes, based on Error-Detection Codes (EDCs), into cryptographic hardware circuits. The underlying adversary model limits attackers to inject at most t single-bit faults. By additionally considering the propagation of faults in combinational circuits, the countermeasure guarantees detection of any faulty computation caused by up to t single-bit faults.

In this work, we present an experimental analysis of the *Impeccable Circuits* countermeasure and its underlying assumptions in modern semiconductor technology. More precisely, we have taken hardware implementations of the lightweight block cipher SKINNY equipped with various forms of the EDC-based CED schemes and realized them as cryptographic co-processors on a 40 nm ASIC to experimentally evaluate their resistance to Laser Fault Injection (LFI) attacks. In short, our results show that it is fairly simple to overcome the protection offered by the integrated countermeasures when the length of the code n is smaller than twice its rank k (i.e., no full redundancy). This is not caused by any flaw in the underlying design methodology or concept, but merely demonstrates how easily the defined adversary model can be overcome. In our case, a standard black-box scan over the target using a common single-shot LFI setup is sufficient to occasionally inject more single-bit faults than those bounded by the underlying adversary model when $n < 2k$. The probability of such events proved to be large enough to perform successful key-recovery attacks via Differential Fault Analysis (DFA) in a matter of hours. Thus, we caution against limiting the redundancy in code-based FI countermeasures to less than the number of bits per word, especially in nanometer technologies, and point out that less-complex countermeasures like duplication showed a higher level of resistance in our experiments at a lower cost.

Keywords: Concurrent Error Detection, Code-based Countermeasures, Impeccable Circuits, Laser Fault Injection, ASIC, Hardware Implementation

*Authors list in alphabetical order; see https://www.ams.org/profession/leaders/culture/CultureStatement04.pdf

Received: 2022-01-15 Accepted: 2022-03-15 Published: 2022-06-08

1 Introduction

Implementations of cryptographic building blocks should not only resist mathematical cryptanalysis and be efficient from a resource consumption point of view, oftentimes they also need to provide protection against physical adversaries. Yet, virtually all cryptographic functions are not resistant to physical attacks when implemented in a straightforward manner. Hence, they have to be equipped with dedicated countermeasures to provide security against adversaries who can physically access the target implementation. This attack vector is particularly relevant for devices in the Internet of Things (IoT), which are given into the hands of users. Depending on the underlying security application, the user himself, and essentially anybody with physical access to the device, needs to be considered as a potentially malicious actor.

Physical attacks typically come in two different flavors. Passive and non-invasive physical attacks, often referred to as Side-Channel Analysis (SCA) attacks, require an adversary to carefully observe the physical properties or emissions of the device under attack in order to gain information about its internal state. Often, the device's timing [Koc96], power consumption [KJJ99] or electromagnetic emanation [GMO01] can reveal crucial information leading to key-recovery procedures. Development of countermeasures against such information leakages has become an important field of research ever since the first successful SCA attacks have been reported in public literature [Koc96, KJJ99]. Fault-Injection Attacks (FIAs), introduced in [BDL97], constitute the first attack vector belonging to the class of active physical attacks. The underlying idea is based on injecting faults during the computation of a cryptographic implementation and subsequently learning information about its intermediate values by analyzing the faulty outputs. This is often achieved through a comparison to fault-free outputs, as it is common practice in Differential Fault Analysis (DFA) attacks [BS97]. Initially proposed as a powerful attack against public key cryptosystems such as RSA [BDL97], fault attacks have quickly been extended to secret key algorithms such as block ciphers [BS97]. DFA attacks on block ciphers are similar to a differential cryptanalysis [BS90] of their last rounds. It has been shown that similar to SCA attacks, DFAs can be mounted on virtually any cryptographic implementation if no dedicated countermeasures have been integrated. Conducting such attacks on an implementation of a block cipher for example requires merely some engineering effort. Typical methods to inject faults into a computing device include focusing a laser beam on the semiconductor surface of the circuit [SA02, vWWM11, SFR⁺15, SFG⁺16, DBC⁺18, RBMM19, CCD⁺21], or transiently changing its supply voltage or the clock source, known as power glitch [KK99, ABF⁺02, KSV13, KJP14] and clock glitch [KK99, PV04, BGV11, ESH⁺11, KSV13]. Alternatively, faults can also be injected by means of electromagnetic pulses [QS02, DDR⁺12, DDRT12, OGM17, MDH⁺13, CH17].

Different countermeasures have been developed to prevent FIAs. Beyond sensors and shields which may be used to sense and prevent physical manipulation of the devices, many FI countermeasures aim at detecting (or correcting) faults on an algorithmic basis during the sensitive computations. Such a detection (resp. correction) of faults typically requires some form of redundancy. This can be realized by repeating the computation multiple times in a row (time redundancy), by instantiating the circuit multiple times (area redundancy), by applying an Error-Detection Code (EDC) or Error-Correction Code (ECC) (information redundancy), or by any combination of these options. Traditionally, whenever a fault is detected, the output is suppressed, so that the adversary cannot learn information from the faulty output. In this case, a classic DFA becomes impossible. However, other sophisticated FIAs like Statistical Ineffective Fault Attacks (SIFAs) [DEK⁺18] are able to overcome fault-injection countermeasures that are based on the detect-and-suppress principle. A SIFA adversary does not require any access to faulty outputs. Instead, correct outputs stemming from faulted computations indicate that the induced fault was ineffective, which can be exploited by SIFA adversaries to recover the secrets. Therefore, simply withholding faulty outputs from the adversary is insufficient to provide protection against advanced fault attacks like SIFA. Fault correction techniques, however, have the potential to protect against this kind of attack as well.

When developing a countermeasure against physical attacks (including both SCA and FIA) it is required to clearly define the adversary model. Without limiting the adversary, no proof for the security of an implementation can be given. In particular, it is not possible to craft an implementation of any cryptographic primitive, that remains secure in presence of an unlimited adversary. In masking schemes, for example, when used as an SCA countermeasure, the maximum order of the attack needs to be defined, which is abstracted as the maximum number of probes the adversary is allowed to simultaneously place on the circuit (i.e., the concept behind the probing security model [ISW03]). Exceeding this threshold could easily result in overcoming the protection as the adversary does not fit into the defined model anymore. In fault injection countermeasures, an adversary's capabilities are often modeled by defining the maximum number of state bits, or circuit wires, that can be faulted simultaneously, i.e., in one clock cycle [IPSW06]. If a countermeasure is capable of detecting (or correcting) all faults caused by up to n faulted bits or wires, but the adversary somehow succeeds in faulting $\geq n + 1$ bits or wires, the faulty computation may remain undetected (or uncorrected). In this scenario, despite a countermeasure being present and active, the attacker can obtain faulty ciphertexts and gain information about the secret key. It has been shown in [BHJ+17] for example, how a parity-based detection scheme can be circumvented by faulting more than 1 bit simultaneously. Additionally, it has been demonstrated that even a single faulty output can be sufficient to guess the secret key with non-negligible probability [TMA11]. Thus, it is crucial to make realistic assumptions about the capabilities of potential adversaries in order to properly adjust the security level. In this regard, the defined adversary model is the primary factor in determining the security and efficiency of the resulting implementation. On the one hand, there is always the risk of underestimating the attackers capabilities which might lead to a vulnerable implementation. On the other hand, being over-protective leads to very costly resource overheads. The goal of such decision making is usually to increase the effort for an adversary to such an extent that the assets which are protected may not be worth the amount of time and resources that have to be invested to perform a successful attack.

The *Impeccable Circuits* countermeasure defines an adversary model by the maximum number of single-bit faults (and their associated clock cycles) that are allowed to be injected into the underlying implementation [AMR+20]. Following this concept, the authors have introduced a design methodology to guarantee the detection of faults injected into any location of the circuit as long as the considered adversary model holds. Such statements are supported by proofs borrowing knowledge from coding theory, as the countermeasure is a Concurrent Error Detection (CED) scheme based on EDCs. To the best of our knowledge, this countermeasure is the first that provides guidelines for the correct implementation of code-based CED schemes in hardware circuits in presence of *fault propagation*. Fault propagation is a detrimental effect that may result in degradation of the error-detection capability that can be achieved with a certain code. Thus, we have chosen this countermeasure as the first one presented in literature that actually guarantees the detection of any fault in a hardware circuit that is covered by the underlying EDC [AMR+20]. This is an important feature for our analysis as we can be certain that any undetected fault is indeed caused by exceeding the capabilities defined by the adversary model and not by unforeseen fault propagation or similar commonly observed behavior of hardware circuits.

As described in [AMR+20], the main goal of the *Impeccable Circuits* authors was to protect against DFA attacks. The application of the proposed technique in a detect-and-suppress manner does not lead to implementations secure against SIFA. However, the scheme has been extended in follow-up works by employing ECCs to guarantee the correction of up to a certain number of single-bit faults [SRM20] and to a hybrid construction allowing the correction as well as detection of faults [RSM21]. Both follow-up works, known as *Impeccable Circuits II* and *Impeccable Circuits III*, consider a similar adversary model and provide proofs for their claims. These extended versions have the potential to protect against SIFA as well as DFA attacks.

1.1 Our Contributions

The contributions of this work are primarily of experimental nature. As stated before, code-based FI countermeasures, like *Impeccable Circuits*, are based on the definition of an adversary model, where theoretical assumptions about the emergence, type, volume and propagation of faults in hardware circuits have to be made. Whenever the theoretical considerations are sound and the countermeasures are implemented correctly, all faults fitting into the underlying model should be detected (resp. corrected). However, it is very rarely investigated whether such theoretical assumptions hold in practice, and how difficult it is to induce faults which fall outside the theoretical model and therefore go undetected (resp. uncorrected).

In this work, we try to answer these questions based on experimental investigations conducted by means of Laser Fault Injection (LFI) attacks performed on real hardware. To this end, we have fabricated a 40 nm CMOS ASIC realizing multiple implementations of the SKINNY lightweight block cipher [BJK+16] equipped with fault-detection facilities following the design methodology of *Impeccable Circuits* [AMR+20]. For the implementations, we have considered different adversary models corresponding to different values t as the maximum number of single-bit fault injections covered. For the sake of comparison, we additionally implemented an unprotected SKINNY core as well as a variant where the traditional duplication technique is applied, i.e., instantiating the core twice and comparing the results.

In summary, without any target-specific fine-tuning, our LFI setup is able to induce *undetected* faults into all cores protected by code-based schemes where the length of the code n is smaller than two times the rank of the code k (i.e., less than full information redundancy). This is neither the case for the cores with $n \geq 2k$ nor for the duplication, which all realize full information redundancy. Since all particularly (area-)efficient variants of the countermeasure ($n < 2k$) proved to be vulnerable, it is indeed the simple duplication that offers the best trade-off between the area cost and the protection level in our experiments (with a single-shot laser). Our results demonstrate that albeit the high design complexity of *Impeccable Circuits*, its variants with insufficient redundancy are easy to bypass in practice using straightforward LFI.

One assumption often made for code-based FI countermeasures and their instantiation in hardware is that injecting more single-bit faults simultaneously into a state word is also more difficult and therefore somehow related to a stronger adversary model. Our results show that this is not necessarily the case. Clearly, in advanced nanometer-scaled CMOS technologies the spot size of even high-precision lasers is often significantly larger than a single logic gate and therefore may affect multiple cells in the same area at once [SFG+16, DBC+18]. Depending on the exact position and energy level of the laser shot, it is thus easily possible to fault multiple gates at the same time. Hence, it is mostly a matter of chance how many gate are faulted in the affected region, which also depends on the data values currently being processed. In conclusion, keeping the redundancy in code-based FI countermeasures small for efficiency reasons can be a dangerous approach as it is not necessarily a hard task, or requires particularly strong adversaries, to inject multiple faults at once into the same region. This is especially true in advanced semiconductor technologies and for gates in close proximity to each other. In this regard, based on our experimental results, we recommend to use codes with at least full information redundancy ($n \geq 2k$) that are able to cover any number of single-bit faults injected into a state word in either the original algorithm or the predictor.

Please note that all cores implemented on our ASIC prototype are protected by detection-based countermeasures. Thus, in this particular setting they cannot offer protection against SIFA, which we also demonstrate experimentally on one of our implemented SKINNY cores. However, as our experimental investigations mainly aim at evaluating the soundness of adversary models in practice, our results are transferable to other protection schemes covering SIFA, where similar adversary models are applied. Examples include recently published schemes based on fault-correction facilities [SRM20, BKHL20, RSM21] and those which combine SCA and fault-detection countermeasures [DDE$^+$20, SBD$^+$20]. We believe that our work provides valuable insights for the community as practical experiments of this kind (taping out a test chip, performing practical experiments with a professional LFI setup) require significant effort and resources, and at the same time are vital to verify the soundness of assumptions and hypotheses that are used to argue about the security of implementations.

2 Background

Generally, the objective of fault-injection attacks is to intentionally introduce malicious failure into a circuit leading to abnormal operation. This can be either changing the flow of a program running on a micro-processor, flipping some bits in the dynamic/static memory, changing the values stored in flip-flops, or manipulating the computation of combinational circuits made from logic gates. The resulting faulty outputs (or in some cases the information regarding detection of faults) may give the attacker an opportunity to reveal the secrets. Such faults can be injected by different means including *temperature* [BDL97, Sko02, KSV13, KJP14], *clock glitch* [KK99, PV04, BGV11, ESH$^+$11, KSV13], *power spikes* also called *power glitch* [KK99, ABF$^+$02, KSV13, KJP14], *electromagnetic pulse* [QS02, PV04, SH07, DDR$^+$12, DDRT12, MDH$^+$13, OGM17, CH17] and optical facilities such as *focused ion beam* [HNT$^+$13], and *laser beam* [SA02, vWWM11, SFR$^+$15, SFG$^+$16, DBC$^+$18, RBMM19, CCD$^+$21].

In this work, we deal with LFI, which is known as one of the strongest and the most precise fault-injection mechanisms. LFI allows the adversary to target a particular part of the circuit, whereas most of the other fault-injection methods cannot achieve such a precision. Since CMOS transistors are generally susceptible to light, depending on the laser spot size and the underlying feature size of the device under attack, the LFI may affect individual gates ranging from one to multiple cells. If the laser beam is very narrow and/or the device has been fabricated by a comparably large technology node, it is even possible to target only one transistor [SA02].

It indeed has become part of general knowledge that security-relevant devices, which may fall into the hands of potential adversaries, should be equipped with dedicated countermeasures. Obviously, one category is protection mechanisms to harden cryptographic circuits against fault-injection attacks. In the following, we shortly review the basics of fault protection mechanisms with a special focus on CED schemes, particularly the preliminaries of *Impeccable Circuits* [AMR$^+$20].

2.1 Countermeasures

Essentially, two different directions have been explored so far. The first category includes physical techniques developed to prevent or detect the injection of a fault. Metal shields [HPS99] are known as a common technique to protect light-sensitive components of the circuits from laser beams. Other schemes use sensors to detect any unusual physical influence on the chip surface (e.g., temperature, voltage, magnetism, or light) thereby halting the circuit to prevent any erroneous computations [MTN$^+$20]. At the positive side,

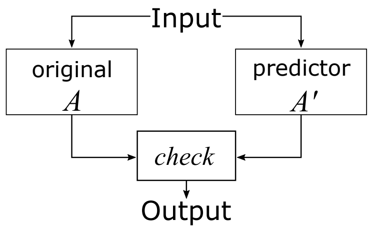


Figure 1: Overview of CED by means of a predictor A'.

these schemes are generic and independent of the underlying circuit, its functionality and its design architecture. They, however, usually lead to a high cost, may still be removed or deactivated by skilled adversaries and only protect against a particular type of fault injection. Thus, it is often possible to circumvent them by means of another fault-injection method not covered by the integrated countermeasure [SA02].

The second category of countermeasures is designed to be implemented at the algorithm level. Their aim is to detect or correct any fault *on the fly* during the operation of the circuit. Therefore, these mechanisms are called Concurrent Error Detection (CED) schemes. Several CED approaches have been introduced and analyzed in the literature to protect against fault-injection attacks [KWMK02, WKKG04, YW06, SSHA08, KR10, KR11, WJJ+15, GMJK15, BHJ+17, AMR+20, BHL21]. Figure 1 illustrates the basic idea. In addition to the original circuit A, a predictor module A' is built. A and A' may operate in parallel which necessitates their individual instantiations. The predictor A' can be an exact copy of A, which allows reuse of A multiple times. Alternatively, more complex algorithms such as parity or error detecting/correcting codes can be used to realize A'. At the end of such computations, the original and the predicted results are compared in order to examine whether the computation was faulty. Generally speaking, suppose that the input of the circuit is denoted by x and its output by $y = A(x)$. The predictor A' realizes a function which can be written as $S \circ A$, where S is called the *signature generator function*. Hence, the *check* module identified in Figure 1 examines the consistency of the predicted signature $y' = A'(x)$ with the actual signature computed as $y'' = S(y)$. When a fault is detected through this check, depending on the countermeasure, either the fault is corrected, the circuit is halted, or a constant/random output is generated.

2.1.1 Duplication

One of the easiest ways to realize the predictor A' is to copy A. Naturally, the signature generator function becomes the identity, i.e., $\forall\ Y, G(Y) = Y$. If A and A' should run in parallel, the area overhead is approximately a factor of two, plus the additional logic required to implement the check module.

Using this structure, any fault injected into only one of A and A' is detected. One possible way to bypass the check is to inject identical faults into A and A'. While this might not be trivial in practice, it has been shown in [SHS16] that – having a priori knowledge about the exact location of corresponding cells in A and A' – identical faults can be injected by means of two laser beams, hence passing through the check module. We discuss about this possibility in Section 5 and argue that such an attack becomes increasingly unlikely to succeed on ASIC implementations in advanced semiconductor technologies, especially when the designer does not strictly enforce that A and A' are realized by exactly the same netlist. Alternatively, one laser beam can target A, and another one may try to circumvent the check process.

In general, A' can include multiple copies of A, which enable correction of some faults, e.g., via majority voting. Naturally, multiple redundant computations increase the security at the cost of a higher area overhead. Alternatively, one can operate A multiple times and compare the result of successive computations. This minimizes the area overhead, but increases the latency accordingly, and may not be effective in presence of permanent faults.

2.1.2 Error-Detection Codes (EDCs)

In addition to duplication, EDCs as an essential technique from information theory, are suggested to be used to detect malicious faults. In the following, the necessary notions related to these codes are reviewed.

Definition 1 (Binary Code)**.** A binary $[n, k]$-code $\mathbf{C}$ with $n > k$, is a bijective mapping from the space of messages $\mathcal{X} = \mathbb{F}_2^k$ to the space of codewords $\mathcal{C} \subset \mathbb{F}_2^n$, i.e., each message $x \in \mathcal{X}$ is mapped to a unique codeword $c \in \mathcal{C}$ with $c = \mathsf{C}(x)$. The parameters n and k are referred to as the *length* and *rank* of the $[n, k]$-code $\mathbf{C}$, respectively. Besides, *signature size* refers to the difference between length and rank, i.e., $n-k$, also called *parity size.*

Definition 2 (Systematic Code)**.** A code in which the message x is embedded in the codeword c is called a systematic code, i.e., the codeword c is the concatenation of x with a signature (redundancy) x', i.e., $c = \langle x \| x' \rangle$, while the signature bits are generated from x. Systematic codes enable a simple split of the data paths between message and signature. Therefore, the original implementation of the target operation can stay as it is. Furthermore, the decoder can take the first k bits of a codeword to extract the message, i.e., no implementation cost.

Definition 3 (Linear Code)**.** The $[n, k]$-code for which the codeword space is a vector subspace over $\mathbb{F}_2^n$ is called *linear.* In other words, any linear combination of codewords is also a valid codeword, i.e., $\forall\ c_1 = \langle x_1 || x_1' \rangle, c_2 = \langle x_2 || x_2' \rangle \in \mathcal{C}$ with $c_1 = \mathsf{C}(x_1)$ and $c_2 = \mathsf{C}(x_2),\ c_3 = c_1 \oplus c_2 = \langle x_3 || x_3' \rangle$ such that $c_3 = \mathsf{C}(x_3) \in \mathcal{C}$.

Most of the works in the domain of protection against fault-injection attacks focused on systematic linear codes. This does not lead to any restrictions, as any linear non-systematic code can be transformed into a systematic linear code with the same properties, i.e., with the same rank and the same minimum distance given in Definition 5. For more information about the transformation procedure, we refer the interested reader to [MS77].

Definition 4 (Generator Matrices)**.** For a linear $[n, k]$-code, the $k \times n$ matrix G that maps a message to the corresponding codeword, is called the *generator matrix*, i.e., $\mathsf{C}(x) = x \cdot G$. Since the rank of the generator matrix is k, there are $n-k$ linear equations between the codeword bits to be satisfied. These equations can be shown as a matrix multiplication. The $n \times (n-k)$ matrix H that checks if an element of $\mathbb{F}_2^n$ is a possible codeword is called the *signature check matrix*[1].

The generator matrix G of a linear systematic $[n, k]$-code is of the form $G = [I_k | P]$ with I_k the identity matrix of size k, while the signature is generated by a $k \times (n-k)$ matrix P as $x' = \mathsf{S}(x) = x \cdot P$ with S being called the *signature generator function*, as defined in Section 2.1 as well.

Definition 5 (Minimum Distance)**.** The minimum distance d of an $[n, k]$-code $\mathbf{C}$ is defined as $d = \min_{\forall c_1 \neq c_2 \in \mathcal{C}}\ \mathrm{hw}(c_1 \oplus c_2)$, where hw denotes the Hamming weight. An $[n, k]$-code with minimum distance d is denoted as an $[n, k, d]$-code.

As an advantage, a code $\mathbf{C}$ with the minimum distance d can detect additive faults $e \in \mathbb{F}_2^n$, if $\mathrm{hw}(e) < d$, i.e., if the faulty codeword $\tilde{c}$ is written as $c \oplus e$. To this end, let us write $\tilde{c} = \langle \tilde{x} \| \tilde{x}' \rangle$. If the result of $\tilde{x}' \oplus (\tilde{x} \cdot P) = \tilde{x}' \oplus \mathsf{S}(\tilde{x})$ is not zero, the codeword $\tilde{c} \notin \mathcal{C}$, hence the fault is detected. This is the exact operation of the *check* module in Figure 1.

[1] In some literature, it is also called *parity check matrix.*

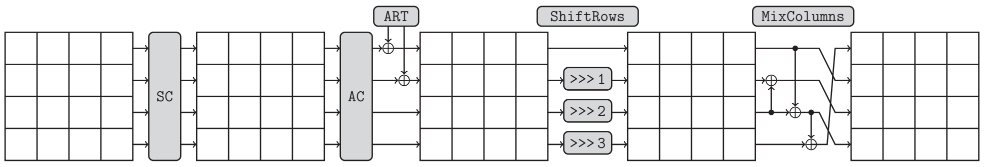


Figure 2: SKINNY cipher, operations of an encryption round [BJK+16].

3 Target

Based on the principles given in Section 2.1.2, the authors of [AMR+20] have introduced a design methodology to integrate an EDC into the hardware implementation of cryptographic algorithms. They have provided suggestions on how to design different modules, how to apply the scheme on the data path as well as on the control logic, and how to check the consistency of signatures. Additionally, they have defined two adversary models: (1) univariate $\mathcal{M}_t$ where the adversary is able to inject t single-bit faults during the entire operation of the cipher, e.g., a full encryption, and (2) multivariate $\mathcal{M}_t^*$ which extends the univariate model assuming that t single-bit faults are allowed to be injected per clock cycle. Please note that injecting t single-bit faults means that an adversary can make up to t individual cell-outputs, or synonymously wires, in the circuit faulty. In accordance with [AMR+20] we denote this as t single-bit (or single-cell or single-wire) faults instead of summarizing them as one t-bit fault (as often done in less hardware-oriented fault injection literature). In that regard, each independent cell whose output is faulted is considered as a separate entity which clearly makes sense for fine-grained hardware implementations. Based on the defined adversary models, the authors have applied their proposed technique on various ciphers and made the implementations available online[2]. We particularly focus on their SKINNY implementations, although the result of our analyses and findings are not limited to these particular implementations. Therefore, in the following, we start with a short explanation on the cipher and then focus on various implementations created by the original authors of [AMR+20].

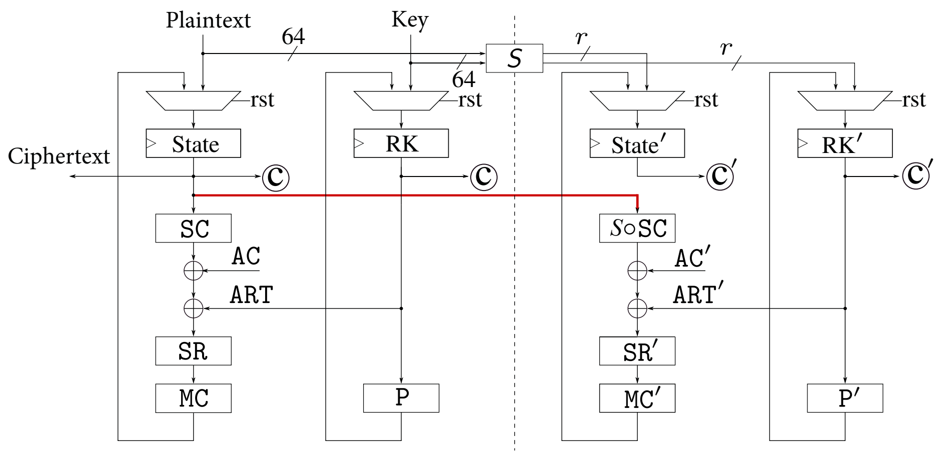
3.1 SKINNY

The SKINNY cipher family introduced in [BJK+16] supports different block sizes as well as key sizes. We focus on a certain variant, namely a block size and key size of 64 bits. As shown in Figure 2, the cipher state is seen as a 4×4 matrix of 4-bit nibbles, while `SubCells (SC)` refers to the application of a 4-bit bijective S-Box on each cell individually, `AddConstants (AC)` describes an XOR with round constants, `AddRoundTweakey (ART)` describes an XOR with the round keys (done on only two rows), `ShiftRows (SR)` rotates the rows similar to the AES decryption, and `MixColumns (MC)` describes a linear layer mixing the columns according to the circuit shown in Figure 2. In the SKINNY-64-64 variant, which requires 32 rounds to accomplish secure encryption/decryption, the key schedule is merely the application of a bit-permutation `P` on the 64-bit key state.

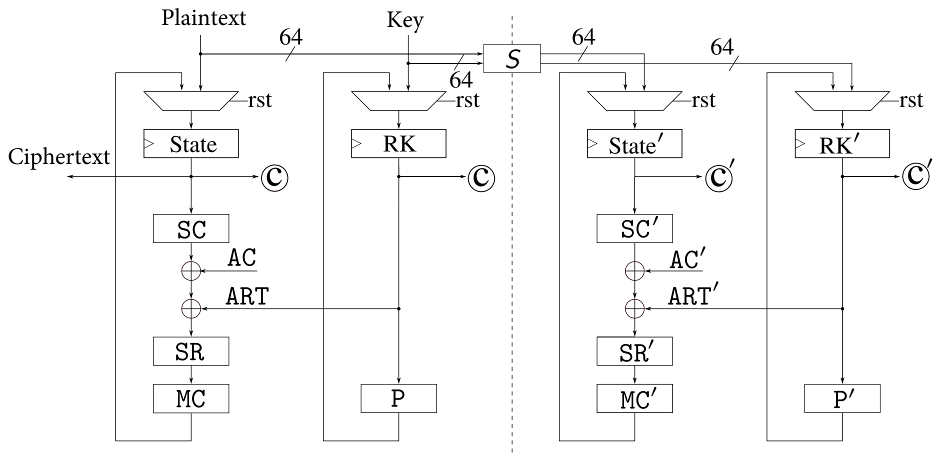
The authors of [AMR+20] have proposed three different $[n, k, d]$-codes ([5,4,2], [7,4,3], and [8,4,4]) for this block cipher. Every nibble of the cipher state is encoded by one of these codes providing 1, 3, or 4 bits of redundancy (signatures). Therefore, the implementations are referred to as `RED 1`, `RED 3`, and `RED 4`, respectively. These also translate to minimum distances 2, 3, or 4 respectively, implying that the implementations should detect any 1, 2, or 3 single-bit faults.[3] Regarding faults injected entirely into only one of the circuits

[2] https://github.com/emsec/ImpeccableCircuits

[3] A `RED 2` implementation is also provided by the authors with 2 bits of redundancy, but it is based on a [6,4,2]-code, i.e., with the minimum distance of 2 similar to `RED 1`. Hence, we excluded this variant.



(a) RED 1 and RED 3 with $r = 16$ and $r = 48$, respectively



(b) RED 4

Figure 3: Block diagram of the round-based SKINNY-64-64 encryption RED 1, RED 3, and RED 4 implementations. The left part of each design depicts the original encryption function, and the right part the predictor circuit, corresponding to A and A' given in Figure 1.

A or A', the RED 4 variant is able to detect any fault in a nibble. The authors created two distinct design architectures based on the size of the signature. When the signature size $n < 2k$ (i.e., RED 1 and RED 3), the design architecture shown in Figure 3(a) is used. Otherwise, when the signature size is at least as large as the message size (i.e., $n \geq 2k$ in RED 4), the implementation follows the architecture shown in Figure 3(b). The main difference between these two architectures is that in case of RED 4 the predictor circuit operates solely on the signature while in case of RED 1 and RED 3 the state of the cipher (denoted by 'State') is additionally required. This part is highlighted by a red connection in Figure 3(a). Below we give more details about each implementation.

3.1.1 RED 1

As given above, a [5,4,2]-code has been applied for this implementation, which is known as the parity code, i.e., the signature is the XOR result of all 4 bits of every nibble. Therefore, the size of the signature for the entire cipher state – as given in Figure 3(a) – is $r = 16$ bits. The signature generator function $S : \mathbb{F}_2^4 \rightarrow \mathbb{F}_2$ and $S \circ \texttt{SC}$, which generates the predicted signature at the S-Box output are given as follows.

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$S(x)$	0	1	1	0	1	0	0	1	1	0	0	1	0	1	1	0
$S \circ \texttt{SC}(x)$	0	0	0	0	1	0	1	1	0	1	0	1	1	1	1	0

The remaining round operations can independently be applied on signatures. More precisely, only one (red) connection from the original circuit A to the predictor circuit A' is required. In short, the XOR with the round constant ($\texttt{AC}'$) and with the round key ($\texttt{ART}'$) can be performed on the signatures, since these operations and the signature generator function of the applied [5,4,2]-code are binary linear. The same holds for $\texttt{SR}'$, $\texttt{MC}'$, and the permutation layer of the key schedule $\texttt{P}'$.

The authors have provided proofs indicating that it is sufficient to examine the consistency of signatures only once per cipher round, as long as the *independence property* is fulfilled. This means that no gate in the combinational circuit should be shared between the sub-circuits which realize different coordinate functions of the cipher round. This avoids a single fault to be propagated to more than one output bit of a cipher round. Therefore, the places identified by Ⓒ in Figure 3(a) are concatenated, and the actual signatures are computed through application of S. The check is finally done by comparing the actual signatures with the predicted ones, i.e., concatenation of places identified by Ⓒ$'$. This check is done following a certain procedure, which we do not show here for the sake of brevity. In short, it avoids the ciphertext to be generated if a single bit fault is detected. Note that the same is done on the control logic made by a 6-bit Linear Feedback Shift Register (LFSR) which generates the round constants as well. This design is expected to be secure in presence of a univariate attack with a single bit fault per encryption, i.e., the $\mathcal{M}_1$ adversary model.

3.1.2 RED 3

Enlarging the distance of the applied code, i.e., a [7,4,3]-code leads to an increase in the number of faults that the design is supposed to detect as well as in the size of the signatures, i.e., 3 bits per state nibble. The design architecture of RED 3 is similar to that of RED 1 with the following signature generation and prediction functions (for the S-Box).

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$S(x)$	0	3	7	4	5	6	2	1	6	5	1	2	3	0	4	7
$S \circ \texttt{SC}(x)$	3	2	5	0	3	1	7	2	4	6	6	0	5	4	1	7

The rest of the operations are similar to what we have explained for RED 1 in Section 3.1.1. Merely the signature size changed from 1 bit to 3 bits, i.e., 48-bit of redundancy for the whole cipher state. Consequently, this implementation is claimed to be secure under $\mathcal{M}_2$ adversary model, i.e., at most 2 single-bit faults per encryption.

3.1.3 RED 4

Here, since the underlying code is the extended Hamming code [8,4,4], the signature has the same size as the original data, i.e., both have 4 bits. Further, since the signature generator function S is bijective, the predictor $\texttt{SC}'$ can be realized without any connection from the original cipher state. In more detail, the design architecture shown in Figure 3(b) can be used which requires only initial generation of the signatures on plaintext and key. Therefore, $\texttt{SC}'$ should first apply the inverse of S, before generating the predicted signature, i.e., $\texttt{SC}' = S \circ \texttt{SC} \circ S^{-1}$ as given below.

Table 1: Post-layout implementation details of the different SKINNY variants.

SKINNY core	Area [GE]	Crit. Path [ns]	Power [µW]
unprotected	2670.00	3.95	154.75
duplication	4997.25	4.82	279.26
`RED 1`	4130.00	4.30	206.15
`RED 1` multivariate	4405.75	6.73	213.63
`RED 3`	5738.75	4.32	290.38
`RED 3` multivariate	6849.75	6.99	315.94
`RED 4`	6878.75	4.52	334.37
`RED 4` multivariate	8305.75	7.95	366.92

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$S(x)$	0	e	d	3	b	5	6	8	7	9	a	4	c	2	1	f
$S \circ \texttt{SC} \circ S^{-1}(x)$	c	8	1	0	2	a	d	3	4	7	5	e	b	9	6	f

The other operations on the signatures are almost identical to the original cipher operations since the size of the signature is the same as that of the original. $\texttt{SR}'$ is identical to $\texttt{SR}$; the same holds for $\texttt{MC}'$ and $\texttt{P}'$. This implementation is supposed to detect any 3 single-bit faults in an encryption process, i.e., security against an $\mathcal{M}_3$ adversary. It can additionally detect any fault of any size injected at only the original part of the circuit, i.e., A.

3.1.4 Multivariate

All above-explained implementations cover only univariate adversary models. The authors of [AMR+20] have explained how to turn a univariate-secure circuit to a multivariate-secure one by (1) placing additional check points at the input of every register and (2) adjusting the consistency check module. Therefore, all three above given implementations have a multivariate variant, which is supposed to detect any $t \in \{1, 2, 3\}$ single-bit faults at every clock cycle, i.e., security against $\mathcal{M}_t$ adversary.

3.2 ASIC Prototype

The target for our experimental analysis is an Application-Specific Integrated Circuit (ASIC) manufactured in a 40 nm low power CMOS technology. The chip is 1920 µm × 1920 µm large and contains routing on 8 metal layers. Its layout and a microscope photography of the bonded die are shown in Figure 4. For operation, the chip requires a core voltage of 1.1 V and an IO voltage of 2.5 V. The dies are packaged in a JLCC (J-Leaded Chip Carrier) ceramic package with 44 pins. To prepare the sample chips for our LFI experiments we have opened the packages from the backside using an X-Prep precision milling system and polished the dies to flatten the substrate surface. This process is shown in Figure 5. The ASIC has been developed for implementation-security evaluations in practice and contains more than two dozen different cipher cores.

All aforementioned countermeasures together with an unprotected core have been implemented on the 40 nm ASIC. We have taken the source code for the hardware implementations created by the original *Impeccable Circuits* authors which are provided on GitHub[2] and realized them as separate clock-gated instances on the chip. Table 1 summarizes the post-layout implementation details for all different SKINNY variants that are targeted in our experiments. It can be observed that the `RED 1` countermeasure requires the smallest overhead in all three categories, area, critical path delay and average power consumption (at 100 MHz). The `RED 3` and `RED 4` variants are costlier than the duplication, especially their multivariate versions.

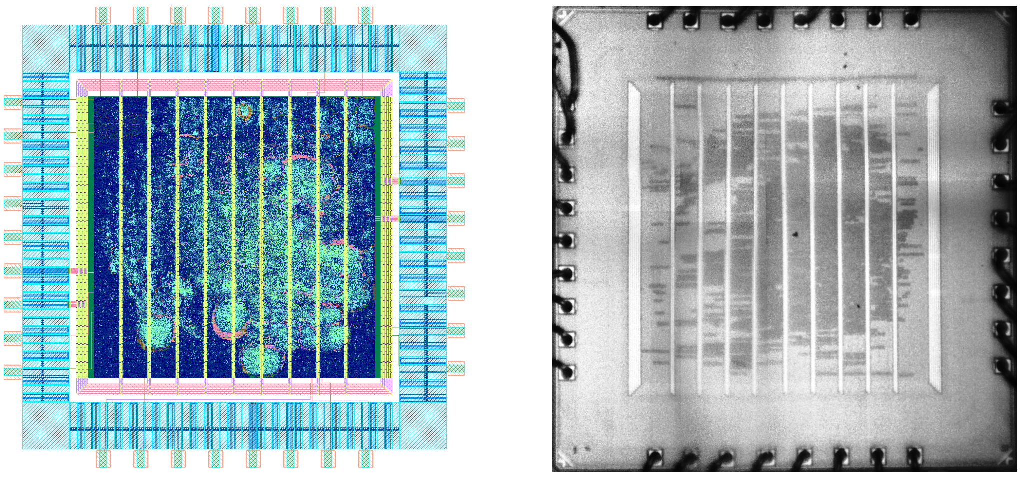

(a) ASIC Layout (b) ASIC Photo

Figure 4: Custom 40 nm ASIC prototype which implements the different SKINNY cores that are targeted in this work.

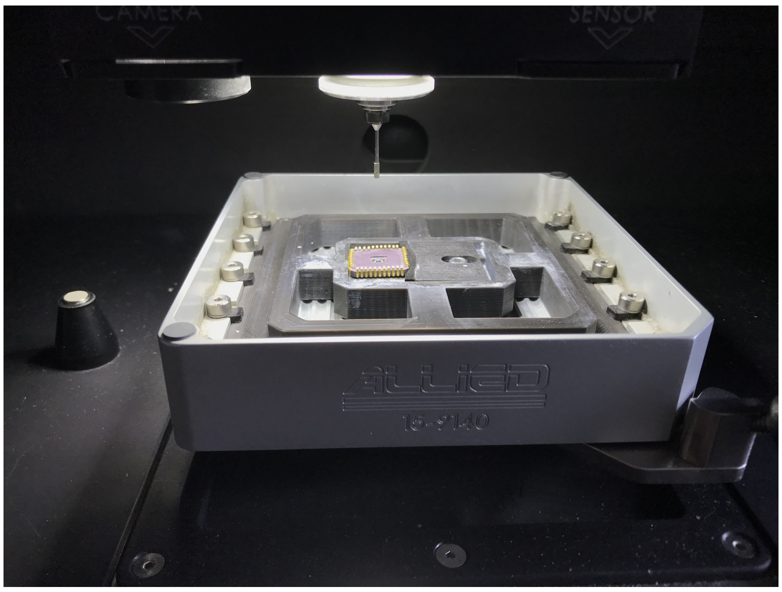


Figure 5: Preparation of an ASIC sample for the LFI experiments using an X-Prep precision milling system.

4 Experimental Results

In this section we present our practical Laser Fault Injection experiments targeting the SKINNY cores listed in Table 1. It is crucial to mention that we deliberately did not use our precise knowledge about the position of individual gates or circuits parts in the experiments, but tried to perform the attacks in a *black-box* setting. We have taken the usual steps an adversary would have to perform without any knowledge about the details of the implementation, from scanning the whole chip to find sensitive areas to targeting the discovered regions with more precision. Using this approach we make sure that our experiments can be performed by any adversary in possession of the device and an LFI setup, without having access to all design details.

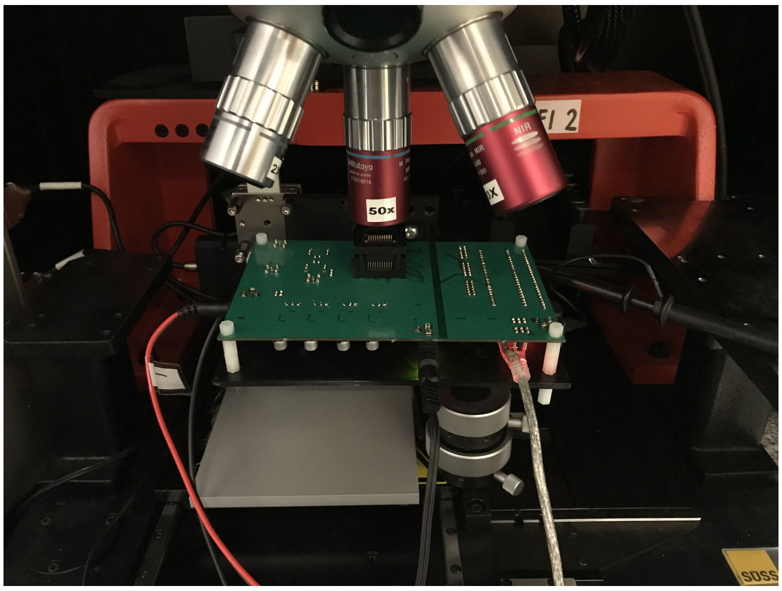


Figure 6: This figure shows the Laser setup used in our experiments. The packaged, opened and polished ASIC is mounted on a custom PCB and the chip surface is targeted with the Nd:YAG Laser to inject faults.

4.1 Setup

The deployed LFI setup is of laboratory grade using a Neodymium-doped Yttrium Aluminum Garnet (Nd:YAG) medium. The optic is capable of laser spot sizes down to 1 µm for a wavelength of 1064 nm while infrared backlight illumination enables identification of metallic structures through silicon for precise spot positioning. The energy is applied in shape of rectangular pulses having a duration of 5 ns at a level of a few up to several hundred nanojoule (nJ). Figure 6 shows a photo of the setup. The ASIC provides accurate signal edges indicating the start of the encryption process, which we used to trigger the laser.

4.2 Energy

We begin by analyzing the unprotected round-based SKINNY in order to determine the optimal energy level to inject faults on this device using two different spot sizes, 28 µm and 5.6 µm. We focus on the sensitive core area identified through an initial coarse full-chip scan (detailed below) and investigate the ratio between correct and faulty responses in function of the respective energy level. Our results for the larger spot, illustrated in Figure 7, show that the number of faulty responses increases steadily up to 23 nJ and then stagnates. Around this value we performed another test using smaller steps to find the optimal level, as seen on the right side of Figure 7. Based on these results we decided to use an energy strength of 22.5 nJ for further experiments with the large spot size (28 µm). To determine the optimal energy levels for the smaller spot size we repeated the corresponding investigation. In this case, the results indicated that less energy is required to most reliably inject faults, namely a value around 16 nJ. All energy levels are averaged over time because the laser is not capable of keeping the energy output exactly constant. However, small deviations are normal for current laser systems. A comparison between the two laser spot sizes clearly shows that the smaller spot is more likely to cause single-bit flips (although it also produces many multi-bit faults) while the large spot produces mostly faults with multiple bit errors.

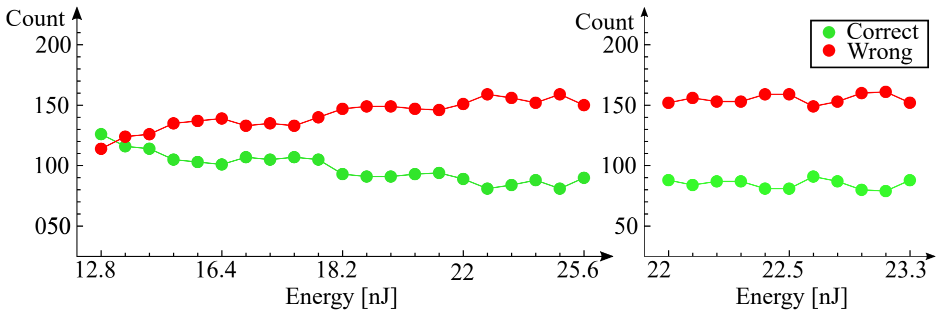


Figure 7: Number of observed faults in the unprotected implementation in function of a rising energy level of laser shots using a spot size of 28 µm.

4.3 Timing

Our measurements show that the SKINNY encryption begins 307.63 µs after startup and takes around 3 µs to finish. To define the delay for every round, we stepped over the whole core with a small step size of 0.05 µs. The received ciphertexts are analyzed in order to determine in which round the fault originated. We found that we can produce faults in rounds 26 and 27 when timing the laser to shoot between 310.25 µs and 310.35 µs after the trigger, while attacks on the last round require a delay of around 310.45 µs to 310.50 µs. However, the laser itself is not capable of always precisely shooting at the same time, thus small temporal deviations may occur. Regardless of that technical limitation, we can consider these timing information for all different SKINNY implementations in the following as they all are based on the same design architecture and require 33 clock cycles for one encryption.

4.4 Scan Procedure

We performed 3 different scans in each experiment targeting one of the SKINNY cores.

1. *Coarse Full-Chip Scan:* At the beginning, we perform a complete scan over the chip surface to verify the location of the targeted core (which is technically already known from the chip's layout). This test is performed with a spot size of 28 µm and the timing is set to inject faults in the middle of the encryption (we do not need informative faults at this point). To avoid a bias in the result, we target each spot three times.

2. *Target Core Scan 1:* The second scan is also performed with a spot size of 28 µm. However, it is focused exclusively on the verified core area and timed to inject faults during the last 0.5 µs of the encryption to collect meaningful outputs for a DFA. Furthermore, we now perform ten injections at each location.

3. *Target Core Scan 2:* The third scan is performed with a spot size of 5.6 µm. It is focused on the verified core area and timed to inject faults during the last 0.5 µs. The amount of shots is set to twenty per location.

The results of these scans are then analyzed and the chip's responses can be divided into multiple categories:

- *No Response*: the chip did not respond at all,

- *Only Zeros*: the faults were successfully detected and suppressed by the countermeasures (output `0x0`),

- *Correct Ciphertexts*: no effective fault could be injected, and
- *Unique Faults*, a new faulty ciphertext has been received (not seen before). These responses are of course of primary interest for our analysis.

4.5 DFA Results

The positions of the individual SKINNY cores in the layout and the results of the initial coarse full-chip scans are depicted in Figure 8. The scan results are shown as heatmaps, highlighting the fault-sensitive areas that can be identified while each specific SKINNY core is operated. For all eight experiments, three different regions are shown.

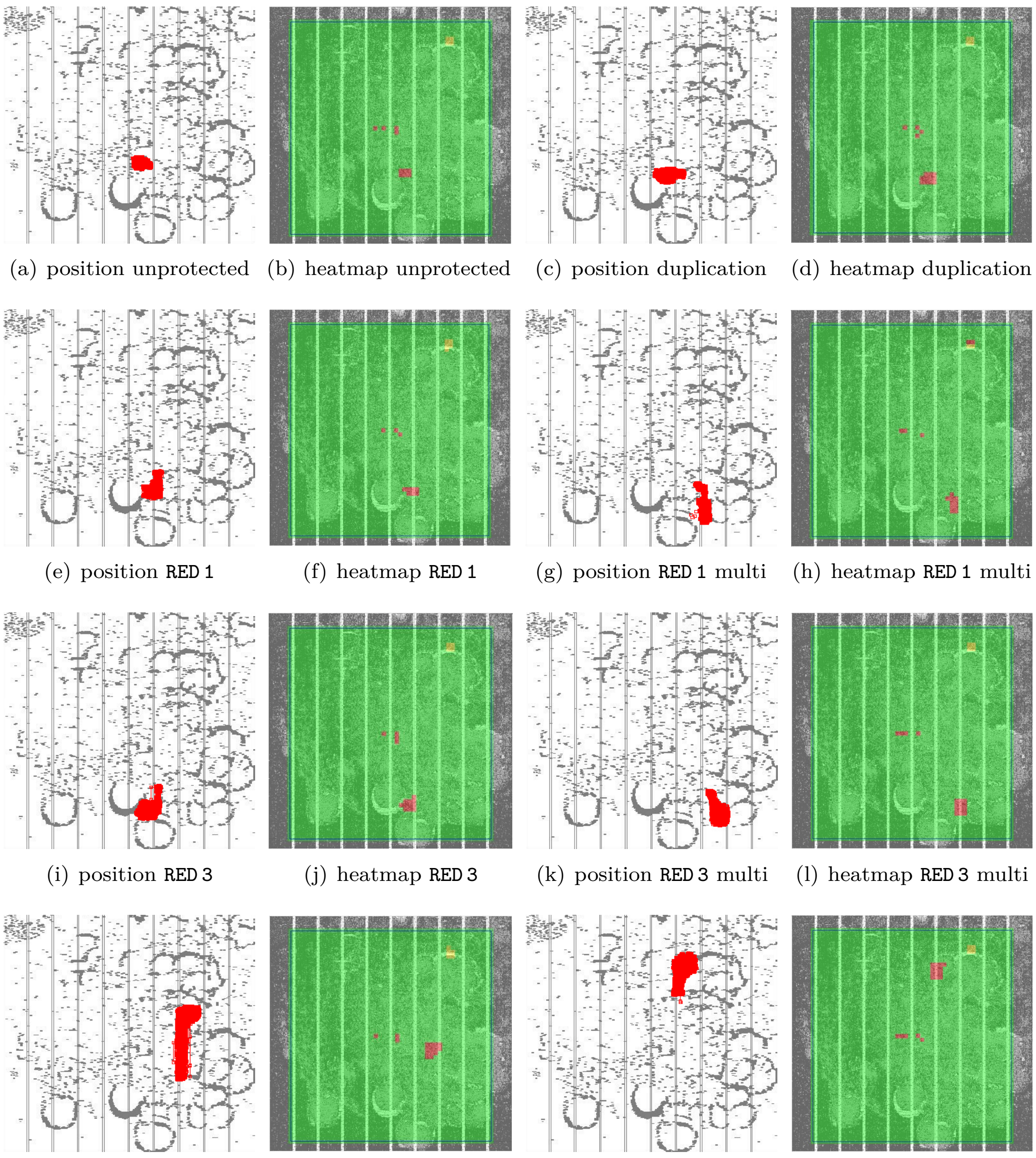
1. *The yellow square on the top right*: In that area, fault injections lead to the chip not responding at all. This location is constant for all analyzed circuits. Here lies control logic for selecting the cipher core to be executed.
2. *The 3-5 small red blocks in the middle*: If the laser shot is timed towards the end of the encryption, we receive a list of faulty ciphertexts that are very close to the correct output. However, we determined that the faulty outputs originate merely from flipped bits in the final ciphertext while they are saved into the output register. Of course, such faulty outputs are not useful for a DFA.
3. *The larger cohesive red area*: This is the location of the targeted SKINNY core. The correctness of the location can be verified by comparison to the highlighted positions in the layout.

Thus, we only target the verified SKINNY core locations in the target core scans 1 and 2 in the following.

Unprotected. The target core scans with the large and the small spot size led to about 400 and 1 000 unique and informative faulty ciphertexts, respectively. This translates to a percentage of 57.6% and 8.1% of unique and informative faulty ciphertexts per attempt. Here, the term informative means that the faultytexts contain exploitable information for the DFA. Ciphertext register bit flips, earlier-round faults, detected faults and non-responses are among the responses that are not counted. With the collected data a full secret key extraction via DFA required less than one second of computation time.

Duplication. After more than 50 000 fault injection attempts in total, not a single informative faulty ciphertext was obtained. About 8 000 faults have been detected and the ciphertext has been suppressed (output `0x0`). Approximately 60 times the device did not respond. However, we will target this SKINNY core by means of SIFA at the end of this section.

`RED 1`. Both, the univariate and the multivariate variant of the `RED 1` scheme, which is based on the $[5, 4, 2]$-code, showed a susceptibility to DFA attacks in our experiments. The target core scans with the large and the small spot size led to about 10 and 150 unique and informative faulty ciphertexts for the univariate variant, respectively, and about 20 and 200 for the multivariate variant. In more detail, of all fault injection attempts the percentage of unique and informative faulty ciphertexts was between 0.3% and 0.4% for the univariate and 0.5% and 0.9% for the multivariate `RED 1`. Approximately 18 000 and 20 000 injected faults respectively have been detected and the output `0x0` was generated. Key extraction via DFA took about 3 seconds in both cases using the collected data. In summary, both `RED 1` variants are susceptible to our LFI attacks as multiple single-bit faults can be injected simultaneously with either of the spot sizes. This makes sense since


(a) position unprotected (b) heatmap unprotected (c) position duplication (d) heatmap duplication

(e) position `RED 1` (f) heatmap `RED 1` (g) position `RED 1` multi (h) heatmap `RED 1` multi

(i) position `RED 3` (j) heatmap `RED 3` (k) position `RED 3` multi (l) heatmap `RED 3` multi

(m) position `RED 4` (n) heatmap `RED 4` (o) position `RED 4` multi (p) heatmap `RED 4` multi

Figure 8: Positions of the individual SKINNY cores on the ASIC and heatmaps of the fault-sensitive areas generated from the initial coarse full-chip scan in each experiment.

the smallest logic gates on the ASIC are around 0.42 μm^2 large and therefore much smaller than either of the laser spot sizes.

`RED 3`. Our results regarding the univariate and the multivariate variant of the `RED 3` scheme are comparable to the `RED 1` results, with the main difference being that the occurrence of unique and informative faulty ciphertext was reduced by approximately a factor of ten. In more detail, the percentage of useful faulty ciphertexts among all attempts was between 0.03% and 0.08% for the univariate and between 0.05% and 0.06% for the multivariate variant. In total, about 40 and 30 unique and informative faulty ciphertexts have been obtained, as they remained undetected, while about 25 000 and 26 000 injected

Table 2: Results of our fault injection experiments targeting the SKINNY cores for key extraction via DFA using two different spot sizes. Only unique and informative faults are counted.

SKINNY Core	Laser Spot Size	Inform. Faults/Attempt	Key Extraction
unprotected	28.0 µm	57.5758%	✓
	5.6 µm	8.0994%	✓
duplication	28.0 µm	0.0000%	✗
	5.6 µm	0.0000%	✗
`RED 1`	28.0 µm	0.3141%	✓
	5.6 µm	0.4123%	✓
`RED 1` multivariate	28.0 µm	0.9110%	✓
	5.6 µm	0.5030%	✓
`RED 3`	28.0 µm	0.0298%	✓
	5.6 µm	0.0817%	✓
`RED 3` multivariate	28.0 µm	0.0570%	✓
	5.6 µm	0.0586%	✓
`RED 4`	28.0 µm	0.0000%	✗
	5.6 µm	0.0000%	✗
`RED 4` multivariate	28.0 µm	0.0000%	✗
	5.6 µm	0.0000%	✗

faults have been successfully detected. Key extraction via DFA took about 1 and 3 seconds using the collected data. In summary, both `RED 3` variants are susceptible to our LFI attacks since a sufficient amount of multiple simultaneous single-bit errors could be injected with the laser setup in order to circumvent the countermeasure.

`RED 4`. The `RED 4` scheme proved to be the most secure of the code-based fault injection countermeasures. After more than 140 000 total attempts with different spot sizes not a single informative faulty ciphertext could be obtained, as all received responses either have been correct ciphertexts or the output `0x0`. This can be explained with the redundancy of the `RED 4` scheme. Any fault in a nibble that is only injected into either the normal algorithm A or the predictor A' leads to an incorrect code word. Since A and A' operate independently and are implemented in separate modules (although directly next to each other on the chip), our experiments targeting one spot at a time are not sufficient to inject undetected faults.

Summary. Our results, especially the precise percentages of unique and informative faulty ciphertexts among all fault injection attempts for both laser spot sizes, are listed in Table 2. It is also indicated whether a successful DFA could be performed (✓) or not (✗). To summarize, the SKINNY cores with less than 4 bits of redundancy per nibble (here the codes with $n < 2k$) are susceptible to simple DFA attacks, while the cores with 4-bit redundancy per nibble (duplication and codes with $n \geq 2k$) are robust against such attacks.

4.6 SIFA Results

In order to demonstrate that detection-based countermeasures which provide solid resistance to DFA can still be circumvented using SIFA, we will perform such an analysis exemplarily on the duplication countermeasure. Since we have not received any faulty ciphertexts for this SKINNY core, a classic DFA is not possible here. However, even if the adversary is unable to obtain faulty ciphertexts SIFA adversaries can exploit the occurrence of

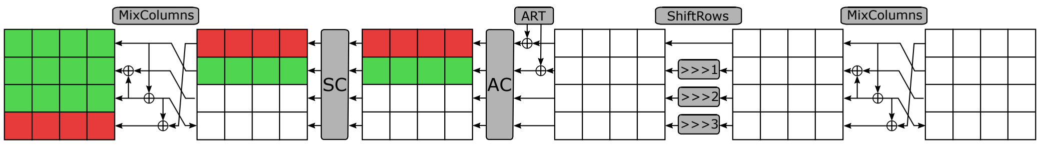


Figure 9: The influence of the round key on the internal state after a round. The red lines show dependencies on the first byte of the key and the green lines on the second byte.

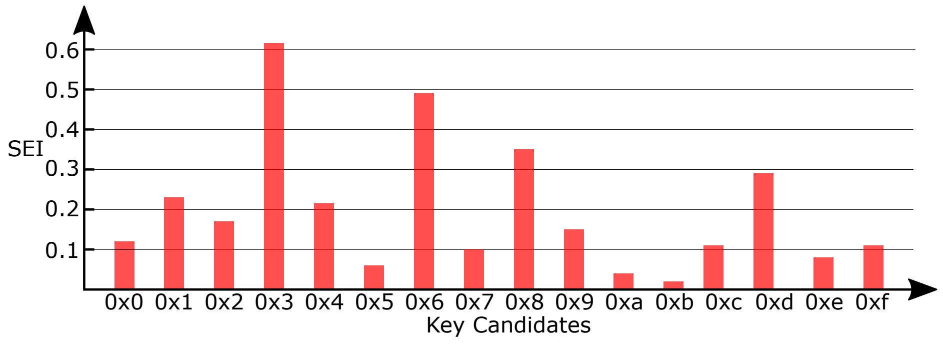


Figure 10: The result of our SEI test for the different key candidates.

ineffective faults to extract the secret key. For a successful SIFA, we first attempt to collect responses for a fault injection that is ideally effective in about 50 percent of the cases (e.g., a 1-bit stuck-at-0 or stuck-at-1 fault). The distribution of intermediate values that lead to a fault-free ciphertext will then be highly non-uniform. The injected fault should be reasonably constant over all encryptions to ensure optimal conditions and the least amount of noise. To ensure this, we choose the 5.6 µm laser spot, knowing that it is likely to affect fewer bits at once. Since we assume a black-box attacker, we assume that we have no knowledge about the exact implementation, except for our self-measured LFI heatmap. Therefore, we do not know where each register is located on the chip, and we have to blindly try to find a suitable spot for SIFA. Due to the fact that we try to inject faults in the last round of the encryption, we set the delay of the laser to 310.5 µs. During our measurements, we shoot a total of 12 000 times and run through 120 different spots. Since we are looking for a fault that in the best case only effects 50 percent of the cases, we only consider spots where we received roughly the same number of detections as correct outputs. The bits marked red in Figure 9 are then targeted for our attack.

The SEI value provides information about the probabilities of the key candidates. The higher the SEI value, the more likely it is that the key candidate is correct. When calculating the SEI, two different approaches lead to the same result. The *MC* and *SR* of the penultimate round do not need to be inverted. In our case, we still invert these to check if we can see some dependencies in row one and two since they are influenced by the second row of the key register, as shown in Figure 9. Figure 10 illustrates the SEI results for the different key candidates of the fifth nibble of the internal state. Due to the inverted *SR* operation, this nibble does not influence the fifth nibble but the sixth one of the last round key. As our results show, `0x3` is the correct nibble of the key.

As we already mentioned, we have also examined the other two rows dependent on the round key. The results are not as clear, but the key nibble is still identified correctly. The *SR* operation does not influence the first row, so the nibble position is the same as the key nibble position. Thus, targeting this one location on the chip already leads to successful extraction of two nibbles of the last round key and shows that the duplication countermeasure is susceptible to SIFA.

5 Discussions and Conclusions

In this work we have experimentally evaluated the effectiveness of code-based countermeasures to provide resistance against Laser Fault Injection (LFI) attacks. In particular, we have put multiple instances of the block cipher SKINNY equipped with different variants of the *Impeccable Circuits* countermeasure on a 40 nm CMOS ASIC. The countermeasure, as introduced in [AMR+20], relies on Concurrent Error Detection (CED) schemes based on Error-Detection Codes (EDC). Our analysis of the protected circuits leads us to the conclusion that using a common laser fault injection setup, without any target-specific fine-tuning or precise knowledge about the underlying design or layout, it is possible to reliably inject faults that remain undetected into the SKINNY cores that are protected by codes with insufficient redundancy. In particular, we find that the SKINNY variants protected by [5,4,2]- and [7,4,3]-codes do not cover all faults that can trivially be injected by an LFI adversary. Since the smallest logic gates on the 40 nm ASIC are about 0.42 μm^2 in size, they are significantly smaller than typical laser spot sizes ($\geq$ 5.6 µm in our experiments). Thus, LFI adversaries can inject multiple faults in the same region without any additional effort compared to a single-bit fault injection. Only the most costly version of the countermeasure based on an [8,4,4]-code provides sufficient protection against DFA attacks. However, our experiments show that protection against DFA attacks can also be achieved by implementing a very simple duplication scheme, which requires about 27% less area (resp. 40% less in the multivariate case) and is much easier to implement. Therefore, we conclude that simple redundancy schemes can be preferable from a cost efficiency standpoint in a practical setting over complex CED schemes if a single-shot laser adversary is considered. We would like to stress the importance of verifying the assumptions and hypotheses, which are made during the design of countermeasures and used to argue about their security, in real-world experiments. Engineering intuition is not always sufficient to predict the behavior of semiconductor devices under stress. As solid examples, we would like to refer to [SRM20, RSM21, DDE+20], where highly complex designs are proposed to protect against single-bit fault adversaries, which – based on our experimental investigations – are easy to bypass.

5.1 Larger SKINNY Variants

We would like to stress that, although our experimental investigation concentrates exclusively on the SKINNY-64-64 variant, our main results and conclusions are mostly independent of the deployed cipher, cipher variant and implementation style. Romulus, a finalist in the NIST lightweight cryptography competition, instantiates a variant of SKINNY that is called SKINNY-128-384+, which is a reduced-round SKINNY-128-384 (40 instead of 56 rounds) with a 128-bit block and 384-bit tweakey size [GIK+]. The interested reader might question whether our results apply in the same manner to this larger variant. Clearly, the main difference to the SKINNY-64-64 in our experiments is the use of an 8-bit S-box in SKINNY-128-384+. Thus, different and larger codes need to be selected when applying the *Impeccable Circuits* countermeasure. This is discussed in Section 5.2 of [AMR+20] for the AES 8-bit S-box exemplarily. Yet, there is no conceptual difference with respect to our main results. As long as the redundancy is smaller than a state word ($n < 2k$), an implementation similar to Figure 3(a) instead of Figure 3(b) is required and the predictor does not operate solely on the signature. In that case it is always true that some multi-bit faults in the original algorithm cannot be detected (or corrected). If, however, the redundancy is at least as large as a state word ($n \geq 2k$), the countermeasure will provide much better security, but also be (significantly) more expensive than simple duplication (c.f., Table 1 of [AMR+20]). While the probability of occurrence of t simultaneous single-bit faults may decrease for larger t, we observed that the threshold after which it becomes too small for a successful attack is clearly higher than 8 considering our target and setup.

5.2 Protection against SIFA

Given our experimental results we suggest that detection-based countermeasures with sufficient redundancy (duplication or [8,4,4]-code) can provide solid resistance against DFA adversaries. However, achieving high security levels in practice against both DFA and SIFA is more challenging. Of course, due to the fact that SIFA is a statistical attack and that its process requires to collect a certain number of ciphertexts where the injected fault was ineffective, it can be a reasonable countermeasure to shut down the device or establish a new key as soon as the first effective fault is detected. For all devices where such an emergency procedure is not possible we suggest implementing triplication or quadruplication approaches (or even higher levels of redundancy) with a majority voting to determine the correct fault-free result (instead of suppressing the output in case of a fault detection). Such implementations should enable the correction of any number of faults that is injected into 1 or 2 of the redundant circuits respectively. Thus, simple redundancy with majority voting is the counterpart to countermeasures based on Error-Correction Code (ECC) and our results do not inspire confidence that the ECC schemes can be more resource-friendly, as they likely need full redundancy to provide protection. However, adversaries who are able to reliably inject multiple symmetric faults into the circuit can overcome simple redundancy schemes. Alternatively, the adversary may inject a fault on one of the cores and the other fault on the majority voting module.

Based on our experience we assume that such an attack becomes increasingly unlikely to succeed the smaller the feature size of the underlying technology is, at least when using a laser setup. The reason for this assumption is not only that each laser shot will inevitably affect multiple logic gates at once, but also the increasing level of process variations in advanced nanometer technologies. Even if two identical circuits are placed right next to each other in symmetrical fashion they will likely behave differently when exposed to the same physical effect. Additionally, when synthesizing and implementing a duplication scheme for example, the original algorithm and the predictor will usually not be realized by the exact same netlist (e.g. with respect to drive strengths) and will not be placed in exactly symmetrical fashion (unless enforced by the designer). Thus, injecting symmetrical faults in a triplication or quadruplication countermeasure with majority voting is expected to be highly difficult. We believe that permanent fault injections pose a greater threat to this kind of countermeasure.

Acknowledgments

The work described in this paper has been supported in part by the German Research Foundation (DFG) under Germany's Excellence Strategy - EXC 2092 CASA - 390781972 and through the project 406956718 'SuCCESS' and by the European Union (EU) through the ERC project 724725 (acronym SWORD).

References

[ABF$^+$02] Christian Aumüller, Peter Bier, Wieland Fischer, Peter Hofreiter, and Jean-Pierre Seifert. Fault Attacks on RSA with CRT: Concrete Results and Practical Countermeasures. In *CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 260–275. Springer, 2002.

[AMR$^+$20] Anita Aghaie, Amir Moradi, Shahram Rasoolzadeh, Aein Rezaei Shahmirzadi, Falk Schellenberg, and Tobias Schneider. Impeccable Circuits. *IEEE Trans. Computers*, 69(3):361–376, 2020.

[BDL97] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the Importance of Checking Cryptographic Protocols for Faults (Extended Abstract). In *EUROCRYPT 1997*, volume 1233 of *Lecture Notes in Computer Science*, pages 37–51. Springer, 1997.

[BGV11] Josep Balasch, Benedikt Gierlichs, and Ingrid Verbauwhede. An In-depth and Black-box Characterization of the Effects of Clock Glitches on 8-bit MCUs. In *FDTC 2011*, pages 105–114. IEEE Computer Society, 2011.

[BHJ$^+$17] Jakub Breier, Wei He, Dirmanto Jap, Shivam Bhasin, and Anupam Chattopadhyay. Attacks in Reality: the Limits of Concurrent Error Detection Codes Against Laser Fault Injection. *J. Hardw. Syst. Secur.*, 1(4):298–310, 2017.

[BHL21] Jakub Breier, Xiaolu Hou, and Yang Liu. On Evaluating Fault Resilient Encoding Schemes in Software. *IEEE Trans. Dependable Secur. Comput.*, 18(3):1065–1079, 2021.

[BJK$^+$16] Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. The SKINNY Family of Block Ciphers and Its Low-Latency Variant MANTIS. In *CRYPTO 2016*, volume 9815 of *Lecture Notes in Computer Science*, pages 123–153. Springer, 2016.

[BKHL20] Jakub Breier, Mustafa Khairallah, Xiaolu Hou, and Yang Liu. A Countermeasure Against Statistical Ineffective Fault Analysis. *IEEE Trans. Circuits Syst.*, 67-II(12):3322–3326, 2020.

[BS90] Eli Biham and Adi Shamir. Differential Cryptanalysis of DES-like Cryptosystems. In *CRYPTO 1990*, volume 537, pages 2–21. Springer, 1990.

[BS97] Eli Biham and Adi Shamir. Differential Fault Analysis of Secret Key Cryptosystems. In *CRYPTO 1997*, volume 1294 of *Lecture Notes in Computer Science*, pages 513–525. Springer, 1997.

[CCD$^+$21] Pierre-Louis Cayrel, Brice Colombier, Vlad-Florin Dragoi, Alexandre Menu, and Lilian Bossuet. Message-Recovery Laser Fault Injection Attack on the Classic McEliece Cryptosystem. In *EUROCRYPT 2021*, volume 12697 of *Lecture Notes in Computer Science*, pages 438–467. Springer, 2021.

[CH17] Ang Cui and Rick Housley. BADFET: Defeating Modern Secure Boot Using Second-Order Pulsed Electromagnetic Fault Injection. In *WOOT 2017*. USENIX Association, 2017.

[DBC$^+$18] Jean-Max Dutertre, Vincent Beroulle, Philippe Candelier, Stephan De Castro, Louis-Barthelemy Faber, Marie-Lise Flottes, Philippe Gendrier, David Hély, Régis Leveugle, Paolo Maistri, Giorgio Di Natale, Athanasios Papadimitriou, and Bruno Rouzeyre. Laser Fault Injection at the CMOS 28 nm Technology Node: an Analysis of the Fault Model. In *FDTC 2018*, pages 1–6. IEEE Computer Society, 2018.

[DDE$^+$20] Joan Daemen, Christoph Dobraunig, Maria Eichlseder, Hannes Groß, Florian Mendel, and Robert Primas. Protecting against Statistical Ineffective Fault Attacks. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(3):508–543, 2020.

[DDR+12] Amine Dehbaoui, Jean-Max Dutertre, Bruno Robisson, P. Orsatelli, Philippe Maurine, and Assia Tria. Injection of transient faults using electromagnetic pulses -Practical results on a cryptographic system-. *IACR Cryptol. ePrint Arch.*, page 123, 2012.

[DDRT12] Amine Dehbaoui, Jean-Max Dutertre, Bruno Robisson, and Assia Tria. Electromagnetic Transient Faults Injection on a Hardware and a Software Implementations of AES. In *FDTC 2012*, pages 7–15. IEEE Computer Society, 2012.

[DEK+18] Christoph Dobraunig, Maria Eichlseder, Thomas Korak, Stefan Mangard, Florian Mendel, and Robert Primas. SIFA: Exploiting Ineffective Fault Inductions on Symmetric Cryptography. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(3):547–572, 2018.

[ESH+11] Sho Endo, Takeshi Sugawara, Naofumi Homma, Takafumi Aoki, and Akashi Satoh. An on-chip glitchy-clock generator for testing fault injection attacks. *J. Cryptogr. Eng.*, 1(4):265–270, 2011.

[GIK+] Chun Guo, Tetsu Iwata, Mustafa Khairallah, Kazuhiko Minematsu, and Thomas Peyrin. Romulus v1.3. https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/finalist-round/updated-spec-doc/romulus-spec-final.pdf.

[GMJK15] Xiaofei Guo, Debdeep Mukhopadhyay, Chenglu Jin, and Ramesh Karri. Security analysis of concurrent error detection against differential fault analysis. *J. Cryptogr. Eng.*, 5(3):153–169, 2015.

[GMO01] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic Analysis: Concrete Results. In *CHES 2001*, volume 2162 of *Lecture Notes in Computer Science*, pages 251–261. Springer, 2001.

[HNT+13] Clemens Helfmeier, Dmitry Nedospasov, Christopher Tarnovsky, Jan Starbug Krissler, Christian Boit, and Jean-Pierre Seifert. Breaking and entering through the silicon. In *CCS 2013*, pages 733–744. ACM, 2013.

[HPS99] Helena Handschuh, Pascal Paillier, and Jacques Stern. Probing Attacks on Tamper-Resistant Devices. In *CHES 1999*, volume 1717 of *Lecture Notes in Computer Science*, pages 303–315. Springer, 1999.

[IPSW06] Yuval Ishai, Manoj Prabhakaran, Amit Sahai, and David A. Wagner. Private Circuits II: Keeping Secrets in Tamperable Circuits. In *EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 308–327. Springer, 2006.

[ISW03] Yuval Ishai, Amit Sahai, and David A. Wagner. Private Circuits: Securing Hardware against Probing Attacks. In *CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, 2003.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In *CRYPTO 1999*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

[KJP14] Raghavan Kumar, Philipp Jovanovic, and Ilia Polian. Precise fault-injections using voltage and temperature manipulation for differential cryptanalysis. In *IOLTS 2014*, pages 43–48. IEEE, 2014.

[KK99] Oliver Kömmerling and Markus G. Kuhn. Design Principles for Tamper-Resistant Smartcard Processors. In *Smartcard*. USENIX Association, 1999.

[Koc96] Paul C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *CRYPTO 1996*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996.

[KR10] Mehran Mozaffari Kermani and Arash Reyhani-Masoleh. Concurrent Structure-Independent Fault Detection Schemes for the Advanced Encryption Standard. *IEEE Trans. Computers*, 59(5):608–622, 2010.

[KR11] Mehran Mozaffari Kermani and Arash Reyhani-Masoleh. A Lightweight High-Performance Fault Detection Scheme for the Advanced Encryption Standard Using Composite Fields. *IEEE Trans. Very Large Scale Integr. Syst.*, 19(1):85–91, 2011.

[KSV13] Dusko Karaklajic, Jörn-Marc Schmidt, and Ingrid Verbauwhede. Hardware Designer's Guide to Fault Attacks. *IEEE Trans. Very Large Scale Integr. Syst.*, 21(12):2295–2306, 2013.

[KWMK02] Ramesh Karri, Kaijie Wu, Piyush Mishra, and Yongkook Kim. Concurrent error detection schemes for fault-based side-channel cryptanalysis of symmetric block ciphers. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, 21(12):1509–1517, 2002.

[MDH$^+$13] Nicolas Moro, Amine Dehbaoui, Karine Heydemann, Bruno Robisson, and Emmanuelle Encrenaz. Electromagnetic Fault Injection: Towards a Fault Model on a 32-bit Microcontroller. In *FDTC 2013*, pages 77–88. IEEE Computer Society, 2013.

[MS77] F.J. MacWilliams and N.J.A. Sloane. *The Theory of Error-Correcting Codes*. Mathematical Studies. Elsevier Science, 1977.

[MTN$^+$20] Kohei Matsuda, Sho Tada, Makoto Nagata, Yuichi Komano, Yang Li, Takeshi Sugawara, Mitsugu Iwamoto, Kazuo Ohta, Kazuo Sakiyama, and Noriyuki Miura. An IC-level countermeasure against laser fault injection attack by information leakage sensing based on laser-induced opto-electric bulk current density. *Japanese Journal of Applied Physics*, 59(SG):SGGL02, feb 2020.

[OGM17] Sébastien Ordas, Ludovic Guillaume-Sage, and Philippe Maurine. Electromagnetic fault injection: the curse of flip-flops. *J. Cryptogr. Eng.*, 7(3):183–197, 2017.

[PV04] Dan Page and Frederik Vercauteren. Fault and Side-Channel Attacks on Pairing Based Cryptography. *IACR Cryptol. ePrint Arch.*, 2004:283, 2004.

[QS02] Jean-Jacques Quisquater and David Samyde. Eddy current for magnetic analysis with active sensor. *Esmart*, 09 2002.

[RBMM19] Joaquin Rodriguez, Alex Baldomero, Víctor Montilla, and Jordi Mujal. LLFI: Lateral Laser Fault Injection Attack. In *FDTC 2019*, pages 41–47. IEEE, 2019.

[RSM21] Shahram Rasoolzadeh, Aein Rezaei Shahmirzadi, and Amir Moradi. Impeccable Circuits III. In *ITC 2021*, pages 163–169. IEEE, 2021.

[SA02] Sergei P. Skorobogatov and Ross J. Anderson. Optical Fault Induction Attacks. In *CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 2–12. Springer, 2002.

[SBD+20] Thierry Simon, Lejla Batina, Joan Daemen, Vincent Grosso, Pedro Maat Costa Massolino, Kostas Papagiannopoulos, Francesco Regazzoni, and Niels Samwel. Friet: An Authenticated Encryption Scheme with Built-in Fault Detection. In *EUROCRYPT 2020*, volume 12105 of *Lecture Notes in Computer Science*, pages 581–611. Springer, 2020.

[SFG+16] Falk Schellenberg, Markus Finkeldey, Nils Gerhardt, Martin Hofmann, Amir Moradi, and Christof Paar. Large laser spots and fault sensitivity analysis. In *HOST 2016*, pages 203–208. IEEE Computer Society, 2016.

[SFR+15] Falk Schellenberg, Markus Finkeldey, Bastian Richter, Maximilian Schapers, Nils Gerhardt, Martin Hofmann, and Christof Paar. On the Complexity Reduction of Laser Fault Injection Campaigns Using OBIC Measurements. In *FDTC 2015*, pages 14–27. IEEE Computer Society, 2015.

[SH07] Jörn-Marc Schmidt and Michael Hutter. Optical and EM Fault-Attacks on CRT-based RSA: Concrete Results. In *Austrochip 2007*, pages 61–67. Verlag der Technischen Universität Graz, 2007.

[SHS16] Bodo Selmke, Johann Heyszl, and Georg Sigl. Attack on a DFA Protected AES by Simultaneous Laser Fault Injections. In *FDTC 2016*, pages 36–46. IEEE Computer Society, 2016.

[Sko02] Sergei Skorobogatov. Low temperature data remanence in static RAM. *University of Cambridge Computer Laboratory Technical Report*, 536, June 2002.

[SRM20] Aein Rezaei Shahmirzadi, Shahram Rasoolzadeh, and Amir Moradi. Impeccable Circuits II. In *DAC 2020*, pages 1–6. IEEE, 2020.

[SSHA08] Akashi Satoh, Takeshi Sugawara, Naofumi Homma, and Takafumi Aoki. High-Performance Concurrent Error Detection Scheme for AES Hardware. In *CHES 2008*, volume 5154 of *Lecture Notes in Computer Science*, pages 100–112. Springer, 2008.

[TMA11] Michael Tunstall, Debdeep Mukhopadhyay, and Subidh Ali. Differential Fault Analysis of the Advanced Encryption Standard Using a Single Fault. In *WISTP 2011*, volume 6633 of *Lecture Notes in Computer Science*, pages 224–233. Springer, 2011.

[vWWM11] Jasper G. J. van Woudenberg, Marc F. Witteman, and Federico Menarini. Practical Optical Fault Injection on Secure Microcontrollers. In *FDTC 2011*, pages 91–99. IEEE Computer Society, 2011.

[WJJ+15] Liang Wen, Wei Jiang, Ke Jiang, Xia Zhang, Xiong Pan, and Keran Zhou. Detecting Fault Injection Attacks on Embedded Real-Time Applications: A System-Level Perspective. In *HPCC 2015, CSS 2015, ICESS 2015*, pages 700–705. IEEE, 2015.

[WKKG04] Kaijie Wu, Ramesh Karri, Grigori Kuznetsov, and Michael Gössel. Low Cost Concurrent Error Detection for the Advanced Encryption Standard. In *ITC 2004*, pages 1242–1248. IEEE Computer Society, 2004.

[YW06] Chih-Hsu Yen and Bing-Fei Wu. Simple Error Detection Methods for Hardware Implementation of Advanced Encryption Standard. *IEEE Trans. Computers*, 55(6):720–731, 2006.