Precise Fault Injection to Enable DFIA for Attacking AES in Remote FPGAs

Xiang Li, Russell Tessier, and Daniel Holcomb
Department of Electrical and Computer Engineering
University of Massachusetts, Amherst, MA, USA

***Abstract*—Differential Fault Intensity Analysis (DFIA) is a class of biased-fault attacks that aim to recover secret keys from block ciphers such as Advanced Encryption Standard (AES). In DFIA an attacker collects a set of ciphertexts generated while carefully controlling the fault intensity, and then performs an analysis on the results that reveals the secret encryption key. In AES, DFIA requires injecting varied intensity faults during exactly the 9th round of encryption, which could be accomplished using clock or supply voltage glitching, although previous works give scant consideration to shaping the fault within a realistic scenario.**

In this work, we demonstrate DFIA against an FPGA implementation of AES without assuming arbitrary external control of clock or supply voltage. Instead we use on-chip ring oscillators (ROs) to create a precise and controllable voltage drop in the vicinity of the AES circuit, which causes timing faults to occur. The fault intensity is finely controlled by changing the number of activated ROs, and we explore how to optimize the timing of the RO activation to cause a fault in the 9th round as is required in DFIA. We use this approach to perform DFIA against AES on Xilinx Spartan-7 FPGA, show that it successfully extracts AES key bytes, and discuss its performance.

I. Introduction and Related Work

Fault attacks (FA) [1]–[14] in FPGAs can occur when excessive on-chip switching stresses the power distribution network (PDN), inducing a voltage drop that causes timing faults [1], [2]. Such attacks can happen remotely, without user access using a variety of power wasting circuits [8], [12]. For example, Mahmoud et al. use ROs to create timing faults in random number generators [10].

Fault and side channel attacks have previously been used to extract encryption keys from FPGAs via differential fault analysis (DFA) [8]. However, differential fault intensity analysis (DFIA), which can recover keys by differential analysis on possibly-faulty ciphertexts without requiring the acquisition of correct ciphertexts, has not been explored in the context of remote FPGAs. DFIA [3], [4] induces faults with clock glitches. Recognizing that clock glitching and voltage drops both cause delay faults, in this work we implement a DFIA attack on AES in an FPGA by using a variable number of RO power wasting circuits to cause supply voltage drops. The voltage drop can be shaped by adjusting the number of ROs and the duration for which they are active. The ability to control the shape of a voltage drop is critical for DFIA, which requires varied faults in the 9th round of AES [3] to extract the secret key. We demonstrate the attack with experiments carried out on a Xilinx Spartan-7 FPGA.

Our work has some similarities to previous on-FPGA DFA experiments using power wasters [8]. This earlier work targets bytes generated before the 9th AES round and requires an adversary to obtain faulty and fault-free ciphertexts to extract the key. DFIA, which our work employs, does not require targeting of specific bytes, nor does it require any fault-free ciphertexts.

II. Attack Model

As mentioned in the previous section, for DFIA, faults must be induced during a particular round of encryption. Ghalaty et al. [3] demonstrated DFIA using a controlled multiplexer to switch to a short-period clock for fault-inducing clock cycles. This type of clock control is challenging to implement in FPGAs since on-FPGA clocks are often derived using a main clock and phase-locked loops (PLLs). In this work, we control fault intensity by enabling differing numbers of single lookup table (LUT) ROs. In general, the number of faults per byte grows with an increasing number of enabled wasters. Using this approach, we implement the following fault attack scenario:

- The victim encrypts plaintext with a secret key that is unknown to the attacker.
- The attacker can trigger repeated encryption of the same plaintexts, but does not know the plaintexts.
- When the attacker triggers an encryption they know the start time of the encryption.
- The attacker collects the ciphertexts, which may be faulty.

For illustration, we give an example by inducing faults in one state byte during the first round of AES. In each trial, the same input value is applied twice to the first round of AES with a differing number of ROs enabled. Fig. 1 shows the Hamming distance (HD) between the resulting bytes, with 50 trials performed for each case. The figure shows that HD increases with incremental fault intensity; similar numbers of activated ROs will induce one or two bits of difference, and vastly different numbers of ROs induce larger HDs. The correlation between HD and fault intensity is the information that DFIA uses to identify the value of each AES key byte.

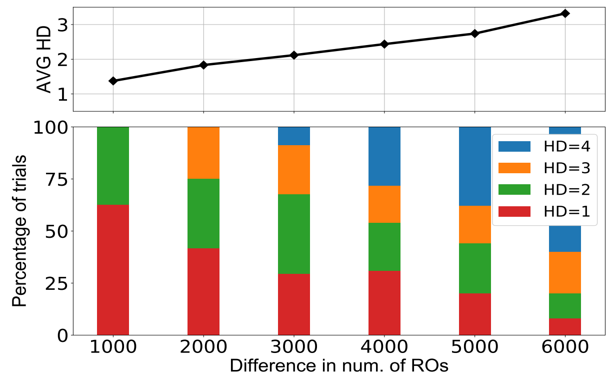


Fig. 1: Distribution of observed Hamming distances relative to the difference in number of activated ROs. Hamming distances tend to be larger when the number of activated ROs in two trials are dissimilar. Trials in which both encryptions produced the same byte value are excluded from consideration.

A. *DFIA on AES*

AES-128 is an iterative algorithm with 10 rounds; the first 9 include operations *S-Box*, *ShiftRows*, *MixColumns* and *AddRoundKey* while the final (10th) round does not have *MixColumns*. In hardware, one round is typically computed per clock cycle, with a register storing and updating intermediate state in each cycle. An attacker may know the timing of AES rounds through on-chip voltage sensors [5] or other means. Fig. 2b depicts where the fault is injected into AES causing wrong bytes captured in the register. The 10th round, in the next clock cycle, still needs to operate correctly, but it operates on the faulty inputs from the register.

We illustrate the DFIA procedure generically on one of the 16 key and state bytes, as the attack works the same against each byte. Intuitively, the attack exploits the similarity (low HD) between values of S in the state register when the repeated encryptions are subjected to similar fault intensities during the 9th round. The attacker, by controlling the number of ROs, knows when state byte values should be similar, but cannot observe the values directly. Because the state byte at the end of the 9th round can be predicted by reversing from ciphertext based on a key byte hypothesis, the attacker can identify the correct hypothesis by finding the one that predicts state bytes showing the expected similarity. Varying the fault intensities causes the number of faulty bits in S to increase or decrease accordingly, but only one key guess will reveal this.

$$C = sbox(S) \oplus K \tag{1}$$

$$S_{i,p} = sboxInv(\mathrm{C_i} \oplus \mathrm{K_p}) \tag{2}$$

Formally, a ciphertext byte C can be described by Eq. 1 where S is a state byte after the 9th round and K is a byte of the 10th round key. We define fault intensity as an increasing order from level 0 to level n. With one plaintext, multiple possibly-faulty ciphertext bytes are generated as $C_1, C_2, \ldots, C_n$ under different fault intensity levels $(0, 1, 2, \ldots, n)$. Among the levels, we use I_D to denote the set of levels that produce a ciphertext different from the preceding level. There are 256 key byte hypotheses (K_0, K_1, … ,K_{255}) available to the attacker. For a given plaintext, the byte $S_{i,p}$ is predicted by Eq. 2 using possibly-faulty ciphertext byte C_i and key byte hypothesis K_p. Only the correct key byte hypothesis predicts $S_{i,p}$ which matches the actual byte captured in the state register at the end of 9th round; other key hypotheses yield arbitrary predictions that do not correspond to the computation performed. An exemplary case with real data is illustrated in Fig. 2a; the predictions and resulting HD are shown for the correct key guess, and one incorrect key guess.

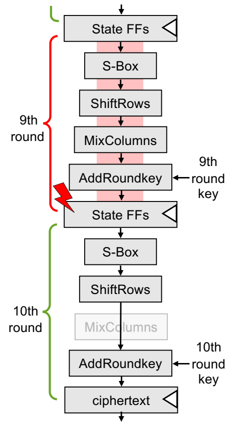
For a given plaintext, the average Hamming distance between adjacent fault intensity levels according to each key hypothesis p is given by Eq. 3. The analysis is extended to m plaintexts in Eq. 4. Provided that fine control of fault intensities causes similar state bytes to be captured (see Fig. 1), avgHD$_p$ will be small when K_p is the correct key byte. The diffusion of *sboxInv* in Eq. 2 causes mispredicted bytes to be random, so avgHD$_p$ for all incorrect key guesses will be centered around 4, which is the expected Hamming distance between random 8-bit strings. The attacker decides the correct key to be the one that has minimum avgHD$_p$. In addition to AES, DFIA can also be applied to some other substitution-permutation networks like PRESENT and LED [4].

$$\mathrm{HD[plaintext]}_p = \frac{1}{|I_D|} \sum_{i \in I_D} \mathrm{HammingDist}(S_{i,p}, S_{i-1,p}) \tag{3}$$

$$\mathrm{avgHD}_p = \frac{1}{m} \sum_{j=0}^{m} \mathrm{HD[plaintext}_j]_p \tag{4}$$

	Prediction with correct key byte		Prediction with wrong key byte	
number of ROs	9th round state byte	HD	9th round state byte	HD
100	0x01	-	0x37	-
200	0x01	0	0x37	0
⋮	0x01	0	0x37	0
6600	0x01	0	0x37	0
6700	0x09	1	0xB0	4
⋮	0x09	0	0xB0	0
8300	0x09	0	0xB0	0
8400	0x19	1	0xA1	2
⋮	0x19	0	0xA1	0

(a) Example Results



(b) Schematic

Fig. 2: DFIA aims to inject faults in the 9th round of AES. Similar fault intensities induce low-HD values in a 9th round state byte. The correct key is identifiable because its predictions reveal low-HD values in the state byte.

III. Characterization of Fault Injection

The requirements of DFIA guide our selection of fault attack parameters such as the number of ROs between levels, and when to enable and disable the wasters. We address these parameter choices in this section of the paper.

A. *Experiment setup*

We perform our experiment on an Arty S7 development board featuring a Xilinx Spartan-7 XC7S50-CSGA324 FPGA with 8,159 slices and 65,200 FFs. We put 12,000 ROs separated from AES by at least one row, which can be enabled in groups of 100, on the FPGA to cause excessive switching activity and high power consumption. The 12,000 ROs, AES and control logic collectively consume 72% of LUTs and 87% of slices. More ROs could be placed when DFIA is applied to larger FPGAs. The AES-128 victim design in this work is a compact round-based implementation that consumes 379 slices and 280 FFs. Round keys are pre-computed. Encrypting each plaintext takes 11 cycles; the first cycle XORs plaintext with initial key and the 10 rounds each take 1 cycle to finish.

B. *Coarse Attack Timing and Number of ROs*

We target the first round of AES operation as a test case to characterize attack timing. Arbitrary plaintexts are used and clock period is set as 8.5 ns. We activate ROs and then start AES after a certain delay which is swept as the parameter in this test. For each delay value, the percentage of correct bits in the first round output is measured. Though the ROs remain on after the first AES round, this does not affect the first round output that is captured. A lower percentage of correct bits implies a larger voltage drop which is causing more path delay faults to occur. The result in Fig. 3 shows instant voltage drop

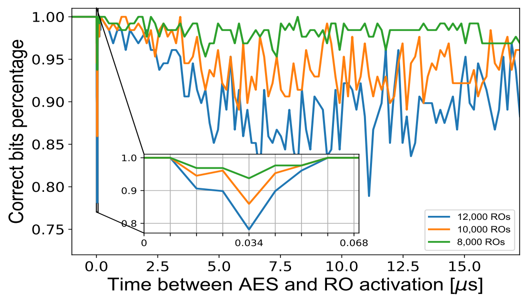
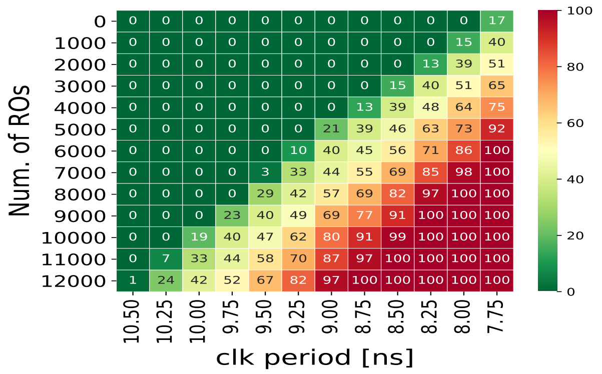


Fig. 3: Profile of faults versus time since RO activation.



Num. of ROs	10.50	10.25	10.00	9.75	9.50	9.25	9.00	8.75	8.50	8.25	8.00	7.75
0	0	0	0	0	0	0	0	0	0	0	0	17
1000	0	0	0	0	0	0	0	0	0	0	15	40
2000	0	0	0	0	0	0	0	0	0	13	39	51
3000	0	0	0	0	0	0	0	0	15	40	51	65
4000	0	0	0	0	0	0	0	13	39	48	64	75
5000	0	0	0	0	0	0	21	39	46	63	73	92
6000	0	0	0	0	0	10	40	45	56	71	86	100
7000	0	0	0	0	3	33	44	55	69	85	98	100
8000	0	0	0	0	29	42	57	69	82	97	100	100
9000	0	0	0	23	40	49	69	77	91	100	100	100
10000	0	0	19	40	47	62	80	91	99	100	100	100
11000	0	7	33	44	58	70	87	97	100	100	100	100
12000	1	24	42	52	67	82	97	100	100	100	100	100

Fig. 4: Number of faulty ciphertexts across different clock periods and number of activated ROs.

and bounce-back after RO activation which is also observed on Intel Arria 10 [13] and Kintex-7 FPGAs [15]. The largest voltage drop happens at the 4th cycle after RO activation, corresponding to a delay of 34 ns between RO activation and the AES round. The quick reaction and recovery provides a promising scenario for injecting faults into a single AES round.

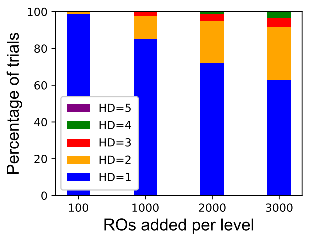
C. Quantifying Delay Change

To cause timing faults, the slowdown by ROs should be significant enough to induce path delay faults at a targeted clock frequency. Choosing appropriate attack parameters therefore requires understanding of timing margins and the amount of slowdown caused by the power wasters. From the timing report in Vivado, the critical path delay of AES module is 14.48 ns, which is known to be conservative.

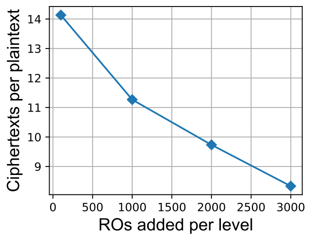
We perform experiments to estimate the timing margins and path delay profile. 100 plaintexts are encrypted across a variety of clock periods and number of activated ROs. The clock periods are swept from 7.75 ns to 10.5 ns in steps of 250 ps, and the number of ROs is swept from 0 to 12,000 in steps of 1,000 ROs. For each intensity and clock period setting, we apply the plaintexts to AES and check the number of incorrect bits produced in the first round output. The result in Fig. 4 shows that faults start emerging when the clock period is 7.75 ns without ROs, which gives an indication of the true delay at nominal conditions of the longest path sensitized during the 100 encryption rounds. To equate ROs to a delay change, we look for multiple pairings of clock and ROs that have a similar number of faults. For example, in Fig. 4, a clock period of 10.25 ns with 12,000 ROs, and a clock period of 7.75 ns without ROs cause a similar number of timing faults. From this we infer that the impact of 12,000 ROs is roughly equivalent to the 2.5 ns difference in clock period, and hence that 12,000 ROs cause around a 32.2% slowdown.

D. Hamming Distance with Different Fault Intensities

Although DFIA does not require fault-free ciphertexts, we assume that the design is overclocked but operating correctly at nominal conditions. The 8.75 ns clock period used in this section allows the AES design to be fault-free when the ROs are off, and susceptible to producing a variety of faulty values when ROs are activated. Fault intensities can be ordered as incremental levels by adding activated ROs between each level. A larger difference of ROs between levels bring about larger changes in the voltage drop, which increases the HDs between neighboring levels. To investigate the resolution of HD, 60 plaintexts are repeatedly applied to AES while varying the number of ROs between adjacent levels. The result in Fig. 5a shows that the HD distribution tends to be larger when more ROs are activated between levels. Because the ideal outcome is to have 1 as the HD value in order to make the correct key stand out, the larger HDs may diminish the attacker's resolution in distinguishing the correct key. Fig. 5b shows how many different ciphertexts arise when sweeping the fault intensity for each plaintext; the percentages in Fig. 5a consider only these cases where the ciphertext changes across levels.



(a) HD distribution



(b) Number of ciphertexts

Fig. 5: Increasing the step size of the fault intensity produces fewer cases with low Hamming distance, and a smaller number of usable ciphertexts per plaintext.

E. Attack Specificity

Subsection III-B showed the existence of a fault spike shortly after RO activation which enables the possibility to attack the 9th round in AES. In this subsection, we determine suitable parameters for the timing and duration of the RO activation, keeping the same 8.75 ns clock period from the prior subsection. To find the optimal parameters, we sweep the timing and duration to activate 12,000 ROs from 1st to 9th round. 1000 random plaintexts are applied to AES and the output of each round is checked to find the earliest round with faults in each trial. Only five suitable configurations that successfully attack the 9th round are identified, and these are summarized in Fig. 6. Activating the ROs for 1 or 2 cycles starting from the 7th round of encryption are both successful in causing 9th round faults, as depicted in the 2nd and 3rd subplots of Fig. 6. Between these choices, we select the former because it does not induce any faults in the 10th round which would render the 9th round fault useless toward distinguishing the correct key.

IV. DFIA EVALUATION

In this section, we present the result of attacking AES and discuss some trade-offs with the control of fault intensities.

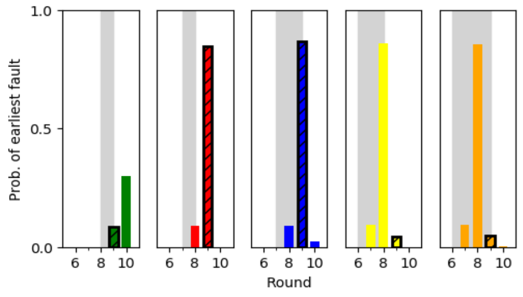


Fig. 6: Five attack scenarios identified for attacking the 9th round in AES. The gray box indicates the cycles in which the ROs are active for each scenario. Activating the ROs in the 7th round for 1 or 2 cycles causes faults in the 8th round in approximately 9% of encrypted plaintexts, and in the 9th round for 86.6% and 84.6% respectively.

Based on the results in the prior section, for DFIA, ROs are activated in the 7th round of encryption for just one cycle and the number of ROs is swept to control intensity. The victim AES is running at 8.75 ns clock period as in the previous section.

A. Key Extraction from AES

We firstly demonstrate the evaluation of extracting key bytes from AES by DFIA. The 12,000 ROs are configured as 121 fault intensity levels, with 100 ROs added per level. For each randomly generated plaintext, we iterate all RO levels with the selected injection timing and perform the DFIA analysis on the resulting ciphertexts. The result of a successful attack toward four arbitrary key bytes is shown in Fig. 7. In order to quantify how many ciphertexts are needed before the correct key stands out, we use Measurements-to-Disclosure (MTD) as the metric. Specifically, we consider a key byte as the correct one when its average HD is lowest among all guesses and remains so for at least 20 consecutive ciphertexts. The threshold for MTD ensures that a key is only declared as correct once it consistently outperforms other guesses, and this increases robustness against randomness in the first few trials. Out of the 16 key bytes in AES, 12 are successfully extracted; the remaining 4 key bytes have shorter paths and produce an insufficient number of faulty ciphertext bytes at this particular clock frequency.

B. Performance

Subsection III-D has shown that the HD tends to be larger when subjected to more ROs added per fault level. The larger HDs might require more plaintexts to extract the keys, because less useful information is derived from each plaintext. On the other hand, when using a finer number of ROs per level, the ciphertexts remain the same (HD=0) across most fault intensity levels, and these trials do not contribute any distinguishing information about the key and represent wasted work by the attacker. We evaluate the tradeoff between these opposing objectives of minimizing plaintexts and minimizing trials performing the DFIA attack with 12,000 ROs while sweeping the number of ROs per level. For each setting, we make use of all intensity levels and measure the number of plaintexts and trials that are required to extract the key bytes. The result in Fig. 8a shows that as the number of ROs per level increases, the number of required plaintexts also increases. This is attributable to two factors: (1) larger HD between levels makes the correct key less distinguishable from other key guesses, and (2) fewer ciphertexts for each plaintext are generated. Therefore, if the goal is to minimize the number of plaintexts, it is desirable to use the finest possible fault intensity levels in order to maximize the information extracted from each. On the other hand, Fig. 8b shows that the number of trials to extract a key decreases as number of ROs per fault level increases, because there are fewer fault levels that generate identical and useless ciphertexts. Note that the observations about selecting the number of ROs per level to minimize plaintexts or trials hold universally across all 12 bytes that are broken in the attack.

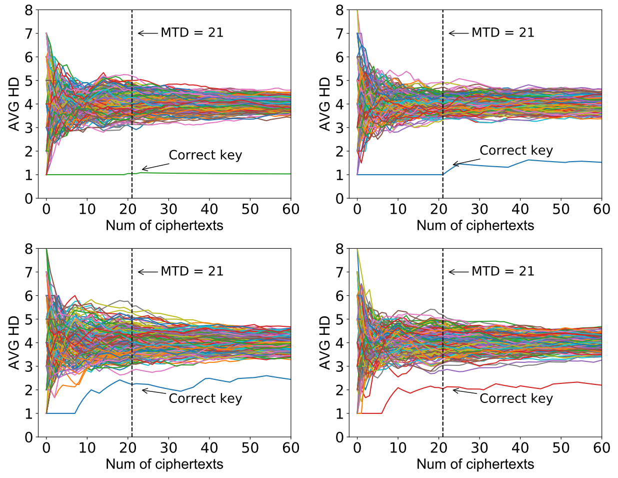


Fig. 7: Average HD traces under 256 key bytes versus number of ciphertexts. The MTD is 21 in each case, and the extracted key byte is confirmed to be the correct one.

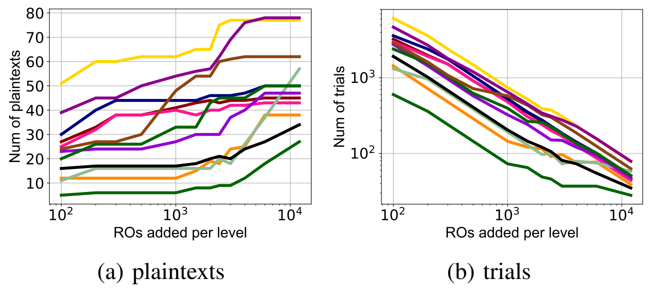


Fig. 8: The traces represent 12 extracted bytes. The number of plaintexts and trials required to extract key bytes vary with the step size of injected fault intensity.

V. Conclusion

This work presents the first DFIA on AES that is suitable for remote FPGAs. The use of ROs to create precisely controllable voltage glitches provides an attacker with fine control of fault injection, allowing for a diverse set of faulty ciphertexts to be generated. We show experimentally that AES key bytes are successfully extracted in the attack, and furthermore study how to tailor fault injection parameters for the attack, and for various competing objectives that the attacker might have.

Acknowledgment: This work has been supported in part by grants from the National Science Foundation (NSF) and Raytheon Technologies.

References

[1] K. Matas, T. M. La, K. D. Pham, and D. Koch, "Power-hammering through glitch amplification–attacks and mitigation," in *IEEE 28th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 2020, pp. 65–69.

[2] M. M. Alam, S. Tajik, F. Ganji, M. Tehranipoor, and D. Forte, "RAM-Jam: Remote temperature and voltage fault attack on FPGAs using memory collisions," in *2019 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, 2019, pp. 48–55.

[3] N. F. Ghalaty, B. Yuce, M. Taha, and P. Schaumont, "Differential fault intensity analysis," in *2014 Workshop on Fault Diagnosis and Tolerance in Cryptography*. IEEE, 2014, pp. 49–58.

[4] N. F. Ghalaty, B. Yuce, and P. Schaumont, "Differential fault intensity analysis on PRESENT and LED block ciphers," in *International Workshop on Constructive Side-Channel Analysis and Secure Design*. Springer, 2015, pp. 174–188.

[5] O. Glamočanin, L. Coulon, F. Regazzoni, and M. Stojilović, "Are cloud FPGAs really vulnerable to power analysis attacks?" in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2020, pp. 1007–1010.

[6] D. R. Gnad, F. Oboril, S. Kiamehr, and M. B. Tahoori, "Analysis of transient voltage fluctuations in FPGAs," in *2016 International Conference on Field-Programmable Technology (FPT)*. IEEE, 2016, pp. 12–19.

[7] D. R. Gnad, F. Oboril, and M. B. Tahoori, "Voltage drop-based fault attacks on FPGAs using valid bitstreams," in *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 2017, pp. 1–7.

[8] J. Krautter, D. R. Gnad, and M. B. Tahoori, "FPGAhammer: remote voltage fault attacks on shared FPGAs, suitable for DFA on AES," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 44–68, 2018.

[9] Y. Li, K. Sakiyama, S. Gomisawa, T. Fukunaga, J. Takahashi, and K. Ohta, "Fault sensitivity analysis," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2010, pp. 320–334.

[10] D. Mahmoud and M. Stojilović, "Timing violation induced faults in multi-tenant FPGAs," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2019, pp. 1745–1750.

[11] A. Moradi, O. Mischke, C. Paar, Y. Li, K. Ohta, and K. Sakiyama, "On the power of fault sensitivity analysis and collision side-channel attacks in a combined setting," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2011, pp. 292–311.

[12] G. Provelengios, D. Holcomb, and R. Tessier, "Characterizing power distribution attacks in multi-user FPGA environments," in *2019 29th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 2019, pp. 194–201.

[13] ——, "Power distribution attacks in multitenant FPGAs," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 12, pp. 2685–2698, 2020.

[14] A. Spruyt, A. Milburn, and Ł. Chmielewski, "Fault injection as an oscilloscope: fault correlation analysis," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 192–216, 2021.

[15] K. M. Zick, M. Srivastav, W. Zhang, and M. French, "Sensing nanosecond-scale voltage attacks and natural transients in FPGAs," in *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, 2013, pp. 101–104.