Solution to HOST 2022 Microelectronics Security Challenge: IP Security Track

Team: Gator Hardware Oriented Security Team (GHOST)
Dipayan Saha, Amit Mazumder Shuvo, Md Kawser Bepary, Shuvagata Saha, Pantha Protim Sarker, Hasan Al-Shaikh, Mridha Md Mashahedur Rahman, Fahim Rahman
University of Florida, Gainesville, FL-32611
{dsaha, amazumdershuvo, mdkawser.bepary, sh.saha, psarker, mrahman1} @ufl.edu, fahimrahman@ece.ufl.edu

***Abstract*—Hardware IPs are vulnerable to malicious threats such as power side-channel analysis and fault injection attacks. Side-channel attacks based on correlation between power traces and input key can be mounted to extract the secret key. Machine-learning based attacks can be deployed to further improve the attack efficiency. On the other hand, fault injection attacks involve clock or voltage glitching to inject malicious faults in the design for possible corruption of critical data. In this work, we performed correlation based power side channel analysis to successfully extract the secret key in as little as 2000 power traces. We also employed machine learning based attack to reduce the required trace number to 1400. In addition, we successfully extracted the 3rd byte of 10th round key using clock-glitch fault injection attack paired with differential fault analysis (DFA). Moreover, we proposed 3 different countermeasures to mitigate side-channel vulnerabilities. We successfully implemented a low-overhead countermeasure that is resilient to correlation-based attacks even with 20k power traces. We also discussed multiple potential countermeasures to protect against fault injection attacks.**

***Index Terms*—Power Side-channel Analysis, Fault Injection Analysis, Hamming Distance (HD), IP Security**

I. Introduction

With the rise of connected devices, the security of the hardware IP is becoming increasingly important. AES [1] is such an IP which is a mathematically secure algorithm for data encryption-decryption. Even though it is mathematically impossible to crack, the implementation of this algorithm in hardware is vulnerable to power side-channel and fault injection attacks. Power side-channel attacks exploit the power consumption data of the hardware IP to reveal the secret key. In comparison, fault injection attacks divulge the key through injecting a fault into the design and analyzing the faulty output. To ensure the correct operation of the AES, securing the implementation against these kinds of attacks has accumulated much attention in the hardware security field.

Considering the significance of assessment of intellectual properties (IP), system on a chip (SoC), and electronic supply chain vulnerabilities, HOST 2022 has arranged Microelectronics Security Challenges in three tracks. In IP security track, the organizers call for solutions to side-channel and fault injection vulnerabilities to an AES core. In this work, we addressed this challenges successfully through attacks and implementation of countermeasure.

II. Attack Methodology and Countermeasure

HOST 2022 Microelectronics Security Challenge: IP Security Track challenges participants to detect power side channel vulnerabilities of IP and propose effective solutions to address the threat. As IP, RTL files of an AES-128 core had been provided. Participants were asked to implement this design into an FPGA system and perform side-channel and fault analyses on the design. The tasks of this challenge has been divided into two major categories: 1) identification of vulnerabilities, 2) mitigation and improvements. In this section, we elaborate these tasks.

A. Identification of Vulnerabilities

Simple Power Analysis Simple power analysis (SPA) involves visual inspection of the power traces and is typically applied to the cryptographic implementations where value of the key determines the operations being executed. For example, in Elliptic Curve Cryptographic (ECC) operation implemented with Double-and-Add algorithm, the key value determines the operation type. Different operations have different power signature and an attacker can classify the operations to extract the key. However, in the AES-128 implementation, the operations are not key dependent. Hence, it is not possible to extract the secret key directly with SPA, rather SPA is utilized to extract the feature points from the power traces which is used in other advanced attacks such as DPA, CPA, etc [2]. We use the same technique in this challenge to identify the portion of power trace corresponding to each round operation of AES.

Differential Power Analysis In this task, the participants have been asked to perform differential power analysis on the FPGA implementation of provided AES design and observe the key information leakage analysis.

Differential power analysis (DPA) exploits the data dependency of the power consumption using a large amount of power traces. As we before discussed, SPA analyzes the power consumption of a cryptographic device along the time axis. Unlike SPA, DPA focuses on the power consumption at a fixed moment of time as a function of the porocessed data.

In order to execute the assigned task, we carried out four variations of DPA on the provided AES core. The attack strategy adopts the following steps sequentially.

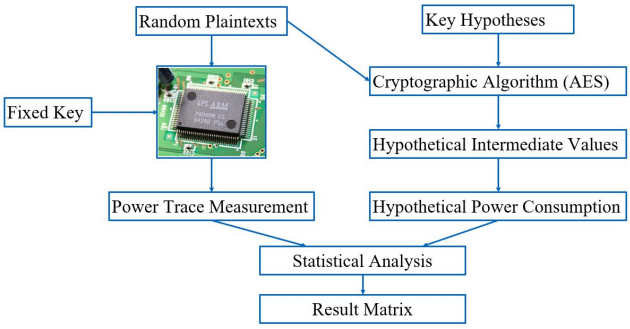

Fig. 1: Overview of differential power analysis

- Step 1 - Choosing intermediate result: At first, we have to choose an intermediate result, which is a function f of ciphertext or plaintext, d and a portion of secret key, k. In our case, we intent to perform attack in divide and conquer fashion. We choose 1st sub-byte of the key and plaintext as d and k. We target the 1st round of AES algorithm and substitute-box (S-box) is the function f in our attack model.
- Step 2 - Measurement of power consumption: As a second step, we collect the power trace of T length $t_i = (t_{i,1}, t_{i,2}, ..., t_{i,T})'$ for during the i^{th} encryption. We run the encryption for D different plaintexts. The setup of the power capture is described in III-A. During the each run of encryption, we also stored the plaintext value, d_i in vector $\mathbf{d} = (d_1, d_2, ..., d_D)'$. In this way, we form a power trace matrix $\mathbf{T} \in \mathbb{R}^{D \times T}$. It is ensured that the captured power traces are in proper alignment.
- Step 3 - Calculation of hypothetical intermediate values: In this step, at first we make a key hypotheses vector $\mathbf{k} = (k_1, k_2, ..., k_K)$, where K denotes the number of hypotheses. For each element of $\mathbf{k}$, a hypothetical intermediate value $v_{i,j}$ is calculated following equation 1

$$v_{i,j} = f(d_i, k_j) i = 1, ..., D; j = 1, ..., K \quad (1)$$

  In this way, matrix $\mathbf{V} \in \mathbb{R}^{D \times K}$ is form which contains hypothetical intermediate values for all combinations of k and d.
- Step 4 - Calculation of hypothetical power consumption values: In this step, matrix of the hypothetical intermediate values $\mathbf{V}$ is mapped to a new matrix $\mathbf{H}$, that contains all values of hypothetical power consumption. We follow following different ways for this mapping.

  a) Hamming-Distance (HD) Model:
  This model counts the number of 0→1 and 1→0 transitions during a certain time interval. The hypothetical power consumption calculation follows 2

$$h_{i,j} = HD(v_{i,j}, v'_{i,j}) \quad i = 1, ..., D; \;\; j = 1, ..., K \quad (2)$$

  b) Hamming-Weight (HW) Model:
  The Hamming-weight model is simpler in conparison to HD model. Here, we assume that the power consumption is proportional to the number of '1' bits in the data. Hypothetical power consumption value is calculated by 3

$$h_{i,j} = HW(v_{i,j}) \quad i = 1, ..., D; \;\; j = 1, ..., K \quad (3)$$

  c) MSB/LSB Model:
  Bit models such as MSB or LSB can also be applied for calculating hypothetical power consumption (shoewn in 4-5).

$$h_{i,j} = LSB(v_{i,j}) \quad i = 1, ..., D; \;\; j = 1, ..., K \quad (4)$$

$$h_{i,j} = MSB(v_{i,j}) \quad i = 1, ..., D; \;\; j = 1, ..., K \quad (5)$$

- Step 5 - Comparison between hypothetical and real power consumption: In this final step, hypothetical power consumption matrix $\mathbf{H}$ is compared with collected power matrix $\mathbf{T}$ and we denote the result of this comparison by $\mathbf{R}$. This comparison can be done in several statistical approaches. This statistical analysis can be based on difference of means, distance of means or correlation coefficient. Depending on this statistical analysis, DPA attack strategy differs. Often DPA based on correlation coefficient is named correlation power analysis (CPA).

  d) DPA with Difference of Means:
  At first, $\mathbf{T}$ is split into two group of power traces. The groups contain all the power traces whose indices correspond to the indices of zeros and ones in h_i, respectively. Let represent the average value of these group of power traces by $M1$ and $M0$. The difference between $M1$ and $M0$ is the result matrix R. It is expected that for the correct key guess, any element of the R matrix r_{ij} will be higher. These steps are described by 6-8

$$m_{1ij} = \frac{\sum_{l=1}^{n} h_{l,i} \cdot t_{l,j}}{\sum_{l=1}^{n} h_{l,i}} \quad (6)$$

$$m_{0ij} = \frac{\sum_{l=1}^{n} (1 - h_{l,i}) \cdot t_{l,j}}{\sum_{l=1}^{n} (1 - h_{l,i})} \quad (7)$$

$$R = M1 - M0 \quad (8)$$

  e) DPA with Distance of Means:
  The distance of means method is an improvement of the difference of means method because this method also takes the variances into account. Here, R is calculated by 9

$$r_{ij} = \frac{m_{1ij} - m_{0ij}}{s_{ij}} \quad (9)$$

  f) Correlation Coefficient Analysis (CPA):
  In this approach, relationship between h_i and t_j is calculated using correlation coefficient. $\mathbf{R}$ is formed by 10

$$r_{i,j} = \frac{\sum_{d=1}^{D} (h_{d,i} - \bar{h_i})(t_{d,j} - \bar{t_j})}{\sqrt{(\sum_{d=1}^{D} (h_{d,i} - \bar{h_i}))^2 (\sum_{d=1}^{D} (t_{d,j} - \bar{t_j}))^2}} \quad (10)$$

  Here, $\bar{t_j}$ and $\bar{h_i}$ are the mean values of t_j and h_i, respectively.

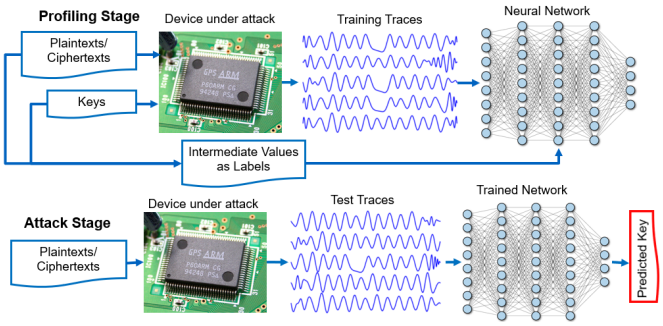

Fig. 2: Overview ML-based power side-channel attack

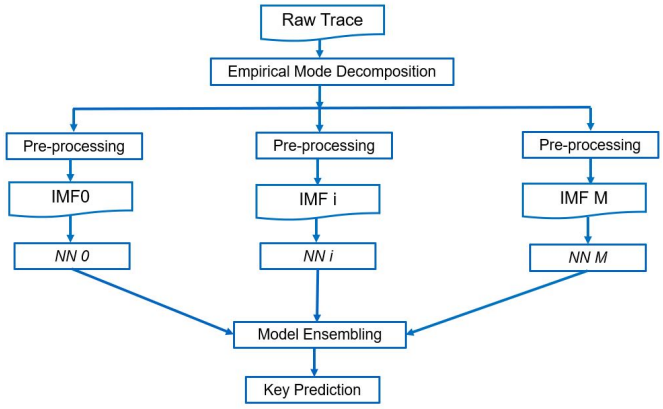

Fig. 3: Overview of our approach for ML-based side-channel attack

Additional Task- ML-based Side-Channel Analysis: Additionally, we performed ML-based side-channel attack. As we know, unlike DPA or CPA, ML-based attack has two stages: profiling and attack stage. An overview of a generic ML-based side-channel attack is shown in Figure 2. In general, a neural network is trained with power traces as training inputs and intermediate values as training labels. This can be formulated a 256-class classification problem.

Signal decomposition is another powerful processing technique that unleashes the intrinsic features unseen in raw signals. In the applied approach, we integrate both signal decomposition and deep learning for side-channel attack. At first, we used empirical mode decomposition (EMD) for decomposing a signal into multiple Intrinsic Mode Functions (IMFs). For each of the decomposed signal we trained a separate neural network. In the attack phase, model ensembling was used for calculated aggregated probability for key prediction. An overview of our approach is shown in Figure 3.

As neural network, we have used multilayer perceptron network. The network architecture has four fully connected layers. Each layer is followed by a ReLU activation function and Batch normalization layer. Mini batch optimization has been used with batch size of 250 and RMSprop has been applied for optimization with learning rate 0.0001.

*1) **Simple Fault Analysis***

A very common threat model can be exploited to perform simple fault analysis on the provided AES core. The done signal can be asserted after 1st round of the AES through fault injection. This will provide a premature encryption and the key can be directly extracted just from the ciphertext and plaintext. The relationship between the plaintext, key, ciphertext, and the output from the round one can be expressed as follows.

$$state\_reg\_1 = plaintext \oplus key \tag{11}$$

For faulty operation (premature assertion of the done signal), output from the round one directly goes to the ciphertext

$$ciphertext = state\_reg\_1 \tag{12}$$

So, from equation 11 and equation 12, we find that

$$key = plaintext \oplus ciphertext \tag{13}$$

*2) **Differential Fault Analysis***

We have performed single-bit fault injection attacks at the input to the 10th round of AES for differential fault analysis. This is a very common and effective attack model proposed in the literature since there is no *MixColumns* operation in the 10th round [3]. The following equations express the relationship between original state at the 10th round's input, faulty state at the 10th round's input, key of the 10th round and ciphetrtext.

$$C = ShiftRows(SubBytes(M^9)) \oplus K^{10} \tag{14}$$

If there is no fault, equation 14 can be written as

$$C_{ShiftRows(j)} = SubBytes(M_j^9) \oplus K^{10}_{ShiftRows(j)} \tag{15}$$

If a single-bit fault is injected at the j_{th} byte of the state register at the 10th round's input

$$D_{ShiftRows(j)} = SubBytes(M_j^9 \oplus e_j) \oplus K^{10}_{ShiftRows(j)} \tag{16}$$

So, from equation 15 and equation 16, we find that

$$C_{ShiftRows(j)} \oplus D_{ShiftRows(j)} = SubBytes(M_j^9) \oplus SubBytes(M_j^9 \oplus e_j) \tag{17}$$

Where M^9 = state register at the 10th round's input, K^{10} = 10th round's key, M_j^9 = j^{th} byte of the state register at the 10th round's input, e_j = single-bit fault at the j^{th} byte, $C_{ShiftRows(j)}$ = j^{th} byte of the original ciphertext after *ShiftRows operation*, and $D_{ShiftRows(j)}$ = j^{th} byte of the faulty ciphertext after *ShiftRows operation*.

So, we used equation 17 to perform differential fault analysis and to extract the j^{th} byte of 10th round's key. With three faulty ciphertext, the j^{th} byte of 10th round's key can be computed with 97% accuracy.

B. Mitigation and Improvements

*1) **Countermeasures against power side-channel analysis***

Power side-channel analysis of the AES core has shown to be a potent tool for extracting meaningful information from the AES core. From this perspective, implementing countermeasures against side channel attacks is of utmost importance. Countermeasures to power side channels can be implemented in different design stages such as RTL, gate level,

physical design, etc. To extract the key, the power side channel attack on AES exploits the correlation among data, operations, and power consumption. The focus of the countermeasures is to reduce or destroy this correlation. Different types of available countermeasures to power side-channel attacks are discussed below:

Masking: Masking is arguably the most popular countermeasure against side-channel analysis. This approach aims to change the intermediate values to randomize those and make the power consumption independent of the associated data. Masking can be implemented at both RTL and gate-level design. The disadvantage of the masking scheme lies in the very high resource overhead. A masked AES requires about three times more gates than the original AES implementation [2].

Hiding: The hiding scheme approaches the side channel countermeasure problem by increasing the AES design's signal-to-noise (SNR) ratio. This event can be performed by either reducing the signal power or increasing the noise power. This approach is another well-researched area for power side-channel countermeasure techniques. Several approaches have shown to be promising. Different hiding approaches can be-

- Introduction of a noise generator in the design - either at RTL level or gate level.
- Implementation of dual-rail precharge (DRP) circuit instead of CMOS [4] such as Sense Amplifier Based Logic (SABL) [5] and Wave Dynamic Differential Logic (WDDL) [6]. This implementation ensures that the circuit always consumes the same amount of power regardless of the operations or operands. Hiding schemes also suffer from the disadvantages of higher area, power, and timing requirements.

t-Private Circuit: t-private circuit countermeasure provides a theoretical framework for constructing a crypto-circuit. The advantage of this approach lies in its usefulness in its provability and application to other types of attacks such as electromagnetic radiation attacks, probing attacks, etc. In this countermeasure, the gates of the circuit are transformed in such a way that to get a single bit of information from the circuit an attacker must probe at least t+1 nodes. Though this is an effective countermeasure, the high area requirement renders the approach impracticable [2].

Countermeasures at the Physical Design: To ensure that the design always consumes the same amount of power regardless of operands in the DRP logic, the capacitances of paired logics need to be the same. As in current technology node interconnect plays a vital role in total capacitance, so it is necessary to match the wire lengths of the paired logic [4]. This matching ensures identical capacitance and thus renders constant power consumption at different gates.

Other Countermeasures: Other than direct changes to the main design of the AES, other countermeasures can also be implemented to make the design side-channel attack resistant. Some of these approaches modify the clock signal, making it harder to synchronize the traces. This kind of countermeasure makes it challenging to employ the DPA attack. These kinds of approaches [4] are mentioned below-

- **Skipping Clock Pulses** - In this approach, some of the clock pulses are skipped.
- **Random Clock Frequency Change** - This countermeasure randomizes the clock frequency by changing the clock period among different rounds using a random number generator.
- **Multiple Clock Domain** - This approach employs multiple clocks in the design. For every clock cycle, one of these clocks is selected at each round based on the output of a random number generator.

*2) **Countermeasures against fault injection analysis***

Besides power side-channel analysis, fault injection attacks have garnered much attention from researchers worldwide. Fault injection is an active attack where a fault is intentionally injected into a crypto core to reveal its secret key [7]. A typical fault injection attack is associated with causing malfunctions at a few registers or nodes in the crypto core. This error eventually propagates to the output. The attacker can divulge the secret by comparing the faulty output with the correct output. Based on the types of faults that can be injected, different techniques of faults are discussed below:

Alongside power side-channel analysis, fault injection attacks are also potent attacks in the attacker's arsenal. An expert, knowledgeable attacker can quickly reveal the secrets stored in the design by executing this attack. So, it is of utmost importance that measures are taken to thwart these kinds of attacks. Fault injection countermeasures can be implemented spatially or temporally. These options at the disposal of the designer are discussed below:

Spatial Countermeasures: In this countermeasure, the design to be secured is copied, and the outputs from all the copies are compared. Suppose there is a mismatch between the outputs. In that case, the fault detection mechanism can assume that a fault injection attack has occurred and take necessary actions to refrain the secret from being revealed. This countermeasure assumes that injecting a fault at the same node or sequential element in a circuit simultaneously is nearly impossible. With the trend of continuous shrinking of VLSI technology nodes, the devices are getting smaller, and the spatial approach is being proven to be a successful countermeasure. This approach comes with the disadvantage of having two or three-fold overheads.

Temporal Countermeasures: Temporal countermeasure is another approach that can ensure the safekeeping of secrets with low overhead costs. In this approach, the crypto computation is performed using the same design blocks at another time. The results of these two computations are compared to determine the presence of fault injection. This countermeasure has a low area overhead, but the timing overhead is increased. For this reason, this countermeasure can be implemented where the timing requirement is not stringent and area overhead must be minimal. It is a perfect candidate to implement in small mobile devices.

III. Experiment and Results

A. Experimental Setup

Chipwhisperer is a collection of tools for researching embedded hardware security. The toolset include trace capture

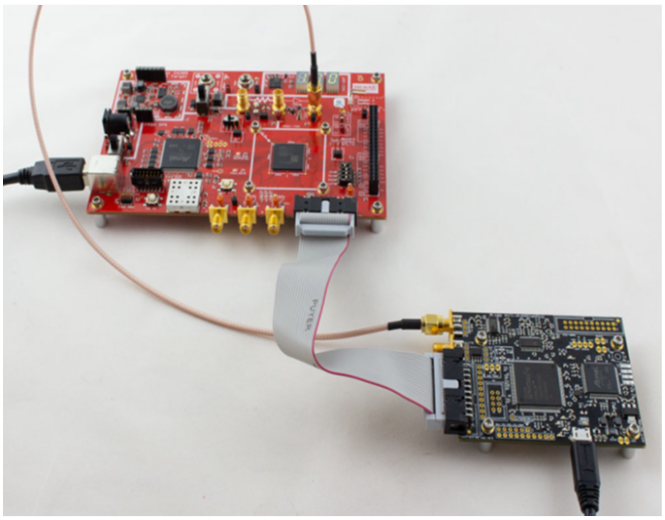

Fig. 4: CW305 FPGA board interconnected with Chipwhisperer-Lite Capture board

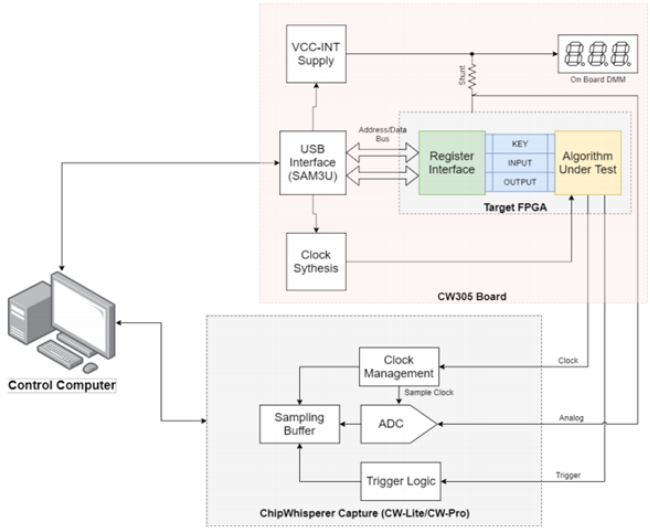

Fig. 5: Schermatic of the CW305 target board and the Chipwhisperer Capture board for trace collection

devices, hardware target device FPGA block, device firmware, and analysis software and libraries. In our experiments, a Chipwhisperer CW305 board equipped with a Xilinx Artix-7 A35 FPGA [8] is used as the standalone target to implement the AES core and a Chipwhisperer-Lite board equipped with a Spartan-7 FPGA [9] is used to capture the power traces required for the analysis. Figure 4 shows the experimental setup for the power trace collection.

The CW305 board features a USB port for communicating with the FPGA, an external PLL for clocking the FPGA, and a VCC-INT supply for programming. At a physical level, the CW305 provides an Address/Data Bus between the USB interface micro-controller and the FPGA. This address/data bus allows us to define a typical address/data bus on the FPGA instead and write arbitrary data into the FPGA. In order to perform side-channel power analysis on the target algorithm, we can use the register interface matching python scripts on the host computer to load input plaintexts, keys, triggering operations, etc. Chipwhisperer python library is used for the communication between the capture board and the target board. Figure 5 illustrates the schematic of the experimental setup including the Chipwhisperer-Lite capture board and CW305 target board.

B. Non-countermeasure Implementation

*1) **Simple Power Analysis:***

As described in Section II-A, we applied simple power analysis on the collected traces. A sample of the power trace is shown in Figure 6. From the figure, it can be seen that there are 10 identifiable cycles in the power trace. These cycles of power traces correspond to 10 rounds of AES. We can easily know the locations of each round in power trace by visual inspection. These cycles can also be tracked with the zero-crossing information or the peak positions. The round position knowledge gained from such simple power analysis can be helpful to perform DPA, CPA or ML-based attack.

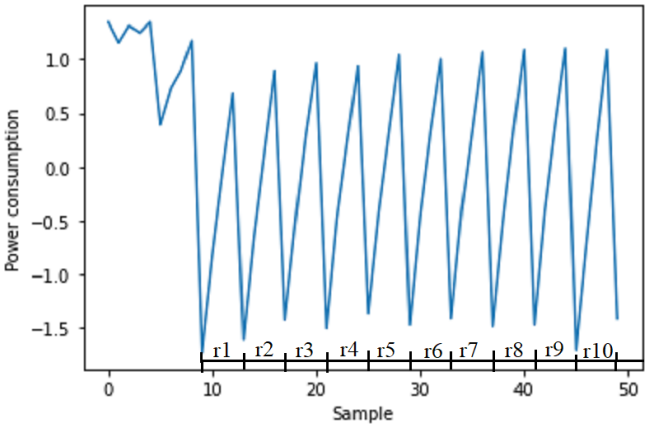

Fig. 6: Simple Power Analysis on the collected trace

*2) **Differential Power Analysis:***

For this task, we performed three different versions of DPA. The attack strategies of our approach are discussed in Section II-A. We implemented following DPA strategies:

a) CPA with HD model:

We performed CPA with HD model on the last round of AES. We used Chipwhisperer side-channel analysis tool-chain to perform this experiment. The performance table for this CPA attack is shown in Figure 7. The table shows that for all the 16 sub-keys, correct key could be guessed with this attack and the correlation values for correct key hypothesis are much higher than those values for incorrect hypotheses. Number of required minimum traces can be observed from Figure 8. The figure shows that for most of sub-keys, around 2k traces are required for key disclosure. Guessing entropy vs. number of traces, shown in Figure 9, also support the same conclusion.

More detailed leakage analysis for CPA attack with HD model is shown in Figure 10. The figure shows how correlation values change along the time axis. It can be seen that around 45^{th}-49^{th} samples of the power trace, there is significant

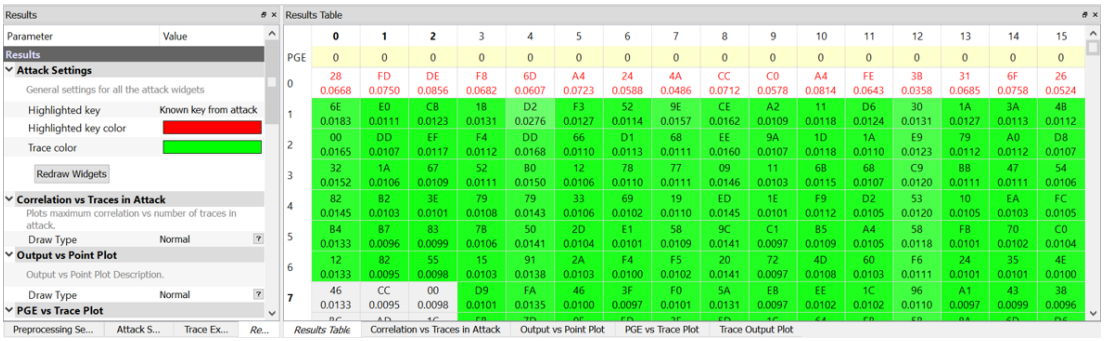

Fig. 7: Performance of CPA with hamming distance model on the last round of AES for non-countermeasure implementation

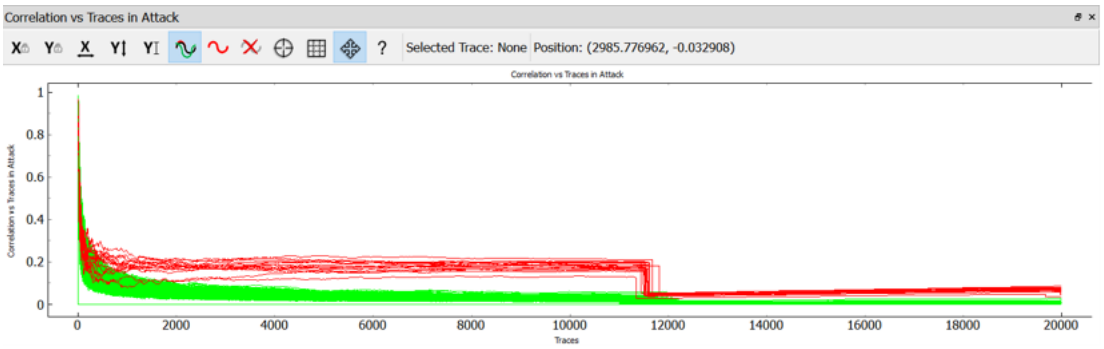

Fig. 8: Correlation vs number of traces for last round CPA with HD model for non-countermeasure implementation

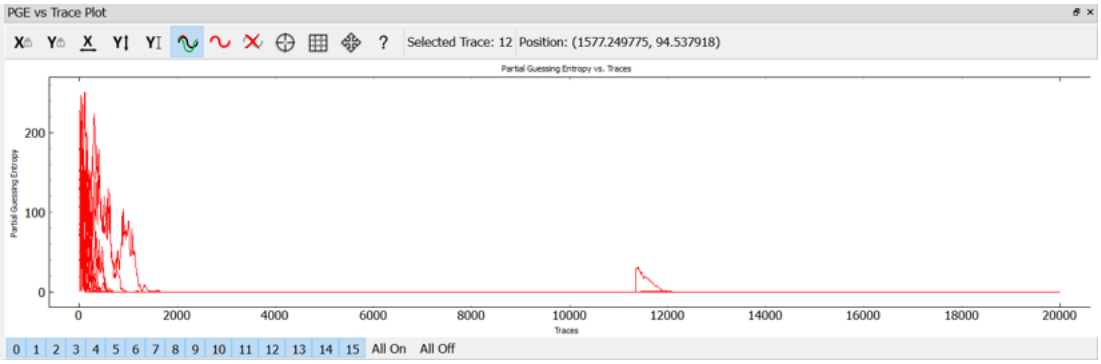

Fig. 9: Partial guessing entropy vs. number of traces for last round CPA with HD model for non-countermeasure implementation

change in correlation values for correct key hypothesis. From SPA, we have already shown that this is exactly the location of last round of AES. Since, in this case we have targeted the last round, side-channel leakage has been identified in that location.

b) CPA with HW model:

We have also implemented CPA with HW model with our python scripts. In this case, we have targeted the 1st sub-byte of the key. Figure 11 shows how correlation values for different key hypotheses change with different number of applied power traces in this attack. We can see that although for correlation value for the right key guess is comparatively low initially for low number of traces, with the increase of traces it increases significantly and finally for 7000 traces it reaches among top values. Correlation coefficients for different key hypotheses for this attack using 7000 power traces are shown in Figure 12 . From the figure, it can be seen that for the correct key guess (key=43), second highest correlation is observed. Though key recovery has failed, the correct key is the top guess. It is expected that with more increased amount of power traces, the correct key guess will be at the top choice.

c) DPA with distance of means:

We have implemented DPA with distance of means on the collected traces. We targeted the 1st sub-byte of the key. In this case, we applied MSB bit model for power hypothesis. The performance of the attack is shown in Figure 13. The figure shows how distance between two groups of power traces changes with increasing amount of traces for different key hypotheses. It can be seen that the distance values for correct key never got higher than those values for other key guesses even if when 20k traces were applied. It clearly shows that DPA could not reveal the correct key in this case.

Fig. 10: HD leakage in Sbox output for non-countermeasure implementation

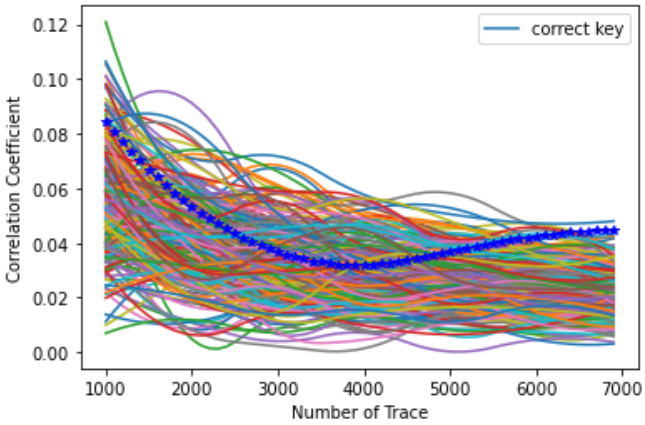

Fig. 11: Correlation coefficient for variation of number of traces for CPA with HW model on non-countermeasure implementation

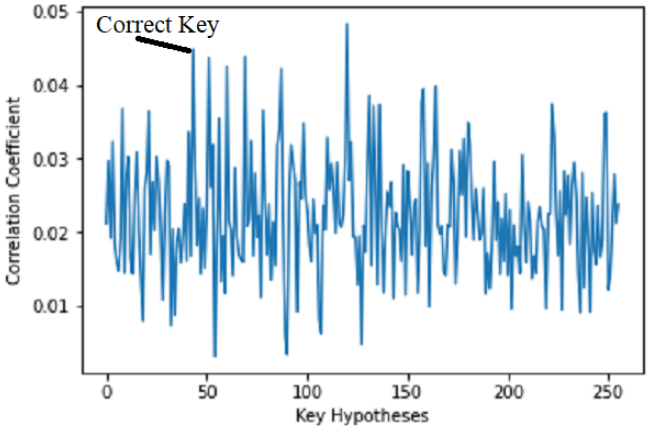

Fig. 12: Correlation coefficient for different key hypotheses in case of CPA with HW model using 7000 power traces for non-countermeasure implementation

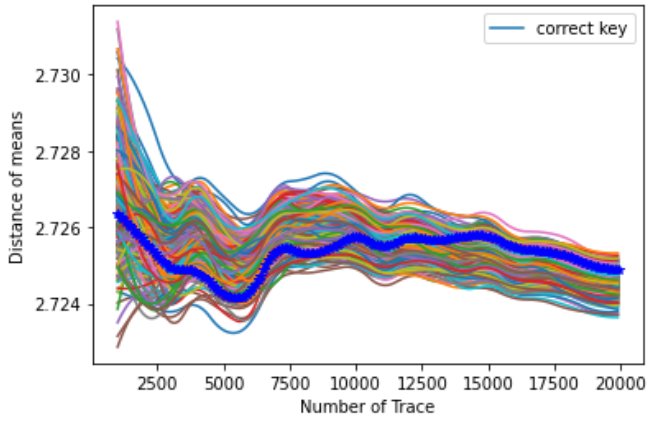

Fig. 13: Correlation coefficient for variation of number of traces for DPA with distance of means on non-countermeasure implementation

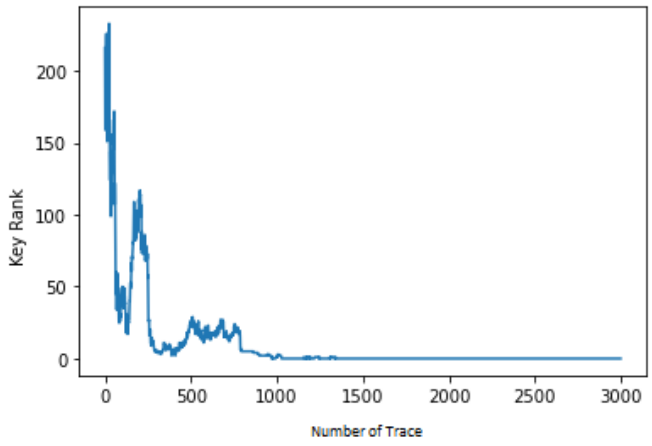

Fig. 14: Key Rank vs. number of power traces for ML-based power side-channel attack for non-countermeasure implementation

*3) **ML-based Side-Channel Analysis:***

For performing ML-based attack, described in Section II-A, we collected 40k power traces for random plaintexts. We split the dataset into split into training, validation and test set with 80%, 10%, ans 10%, respectively. The performance for the ML-based attack is shown in Figure 14. The figure shows that only around 1400 traces are need to extract the correct key with this approach.

*4) **Differential Fault Analysis:***

We have introduced timing faults at the state register of the round 10^{th} of the AES core through clock glitch. Insertion of a clock glitch shortens the effective time period of the clock and introduces metastability by violating setup-time/hold-time constraints. These violations causes metastability and flips the output of the register under attack. We have tuned the clock parameters of Chipwhisperer to generate glitches. We set glitch

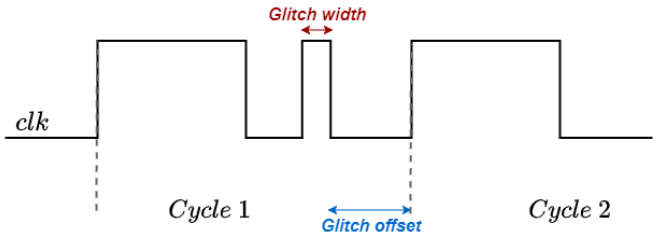

Fig. 15: Illustration of the glitch parameters

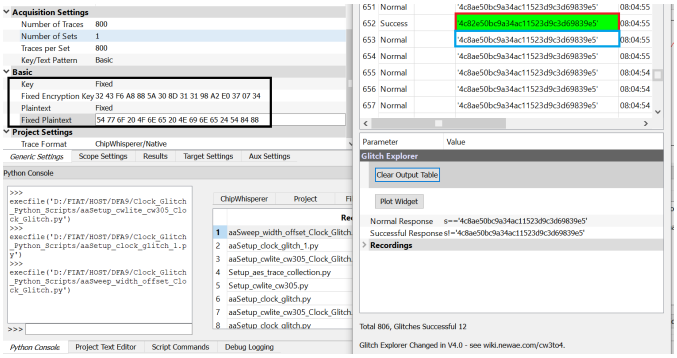

Fig. 16: Clock Glitch Attack with *Plaintext 1*

offset as -14.609%, glitch width as 0.39% for a 10 MHz clock. We changed both the glitch offset and glitch width gradually to inject a single-bit controlled fault at any byte of the state register input to the round 10. Fig. 15 shows the parameters of the clock glitch.

Then we performed DFA attack on the 10th round. We got 2 faulty ciphertexts for 2 different attack where the plaintexts are different, however, the keys are the same. Injected faults caused single-bit controlled fault injection at the input to the 10th round of the AES. Successful attack scenraios are illustrated in Fig. 16 with *Plaintext 1* and in Fig. 17 with *Plaintext 2*. The red marked box includes fault-injected value of the state register and blue marked box includes original value of the state register at the 10th round's input in both figures. In both cases, the 14th byte of the state register got changed by single-bit fault injections. As a result, the 3rd byte of the ciphertext will be faulty for both cases.

Finally, we performed differential fault analysis to extract

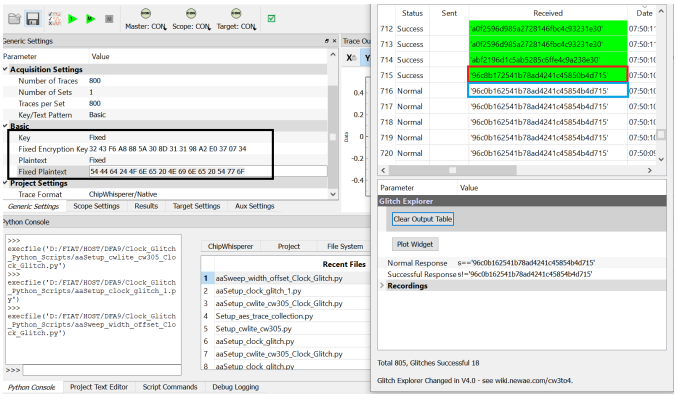

Fig. 17: Clock Glitch Attack with *Plaintext 2*

```
Fixed Key  0x3243f6a8885a308d313198a2e0370734
10th Round's Key  0xcbf39708ac73d39523cf94af518317ac

Plaintext1  0x54776f204f6e65204e696e6524548488
Original Cipher 0xe2f9a2d17155c1be7a894dd7a7fdc182
Faulty Cipher   0xe2f9a2d17155c1be7a894dd7a790c182
C_orig xor C_faulty   0x6d0000

Plaintext2  0x544464244f6e65204e696e652054776f
Original Cipher 0x5b5c8b518cf0dd3f0f425c3a7139abc6
Faulty Cipher   0x5b5c8b518cf0ddd50f425c3a026babc6
C_orig xor C_faulty   0xea0000000073520000
```

Fig. 18: Original and faulty ciphers for two different plaintexts

```
Analysis of the possible search space

82 0xee 8a 0x83 8 6d   1
a4 0xb4 a6 0xd9 2 6d   1
e0 0x1c f0 0x71 10 6d   1


3d 0x1e 3f 0x4c 2 52   1
41 0xba 51 0xe8 10 52   1
c0 0x83 c8 0xd1 8 52   1
c7 0xff e7 0xad 20 52   1

Key search space for Plaintext1
['0xee', '0x83', '0xb4', '0xd9', '0x1c', '0x71']

Key search space for Plaintext2
['0x1e', '0x4c', '0xba', '0xe8', '0x83', '0xd1', '0xff', '0xad']

key_byte
['0x83']
```

Fig. 19: Extraction of key from the analysis of search spaces

the 3rd byte of the 10th round's key. Fig. 18 illustrates the original and faulty ciphertexts for the two attack scenarios.

We have developed a python script that helps to obtain the search space of the 3rd byte of the 10th rounds key. The common element between the search spaces for two attack scenarios gives the correct key byte. Fig. 19 shows that our analysis successfully extracted the 3rd

C. Implementation of Countermeasures

a) *Implemented Approach (Galois Field Transformation):*

The Byte Substitution operation of the target AES design is implemented with a look-up table (LUT). Such LUT block is considered as the most vulnerable part of AES implementation. Composite Galois Field arithmetic can be applied in the Sbox Transformation of the AES algorithm [10]. Isomorphic mapping can be employed to convert $GF(2^8)4$ to $GF(((2^2)^2)^2)$, so that multiplicative inverse can be easily obtained. Implementation of the galois field algorithm prevents the direct mapping of LUT which is the primary source of leakage. The composite field arithmetic involves Galois Field $GF(2^8)$ transformations containing two operations- Multiplicative Inversion and Affine Transformation, as shown in Figure 20.

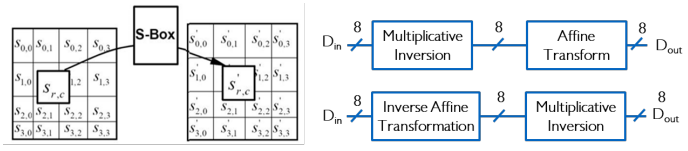

Fig. 20: SubByte and Inverse SubByte Transformations in Sbox

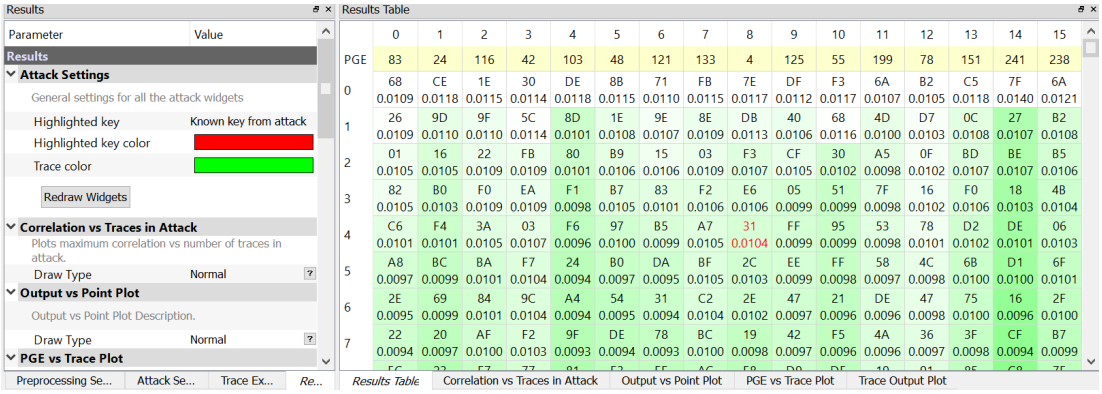

Fig. 21: Performance of CPA with HD model on countermeasure implementation

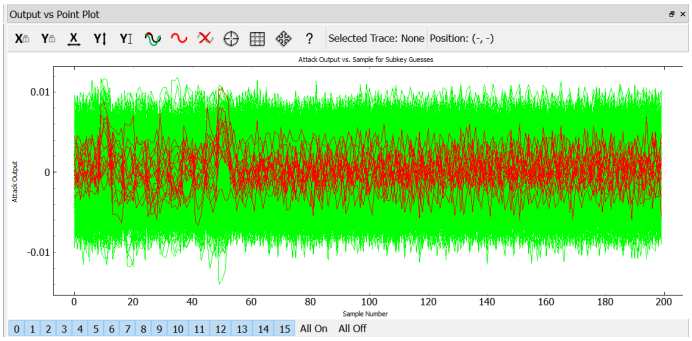

Fig. 22: Correlation vs number of traces for last round CPA with HD model on countermeasure implementation

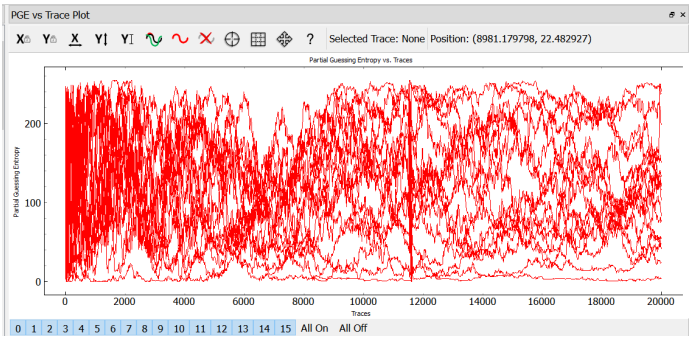

Fig. 23: Partial guessing entropy vs. number of traces for last round CPA with HD model for countermeasure implementation

Computation of the multiplicative inverse in composite fields cannot be directly applied to an element which is based on $GF(2^8$. The complex $GF(2^8)$ can decomposed to lower order fields of $GF(2^1)$, $GF(2^2)$, $GF(2^2)$, $GF((2^2)^2)$ [10]. The following irreducible polynomials are used:

$$GF(2^2) \rightarrow GF(2) : x^2 + x + 1$$
$$GF((2^2)^2) \rightarrow GF(2^2) : x^2 + x + \phi$$
$$GF(((2^2)^2)^2) \rightarrow GF((2^2)^2) : x^2 + x + \lambda$$
$$Where\ \phi = \{10\}2\ and\ \lambda = \{1100\}2.$$

We implemented the Galois Field decomposition to replace the LUT-based Sbox implemented in the provided AES design. Then, we programmed the FPGA and collected power traces using the ChipWhisperer experimental setup. We performed CPA with HD model on the last round of operation to compare with the original attack results.

The performance of the attack using 20k traces on this countermeasure table is shown in Figure 21 in tabular format. The table shows that for none of the 16 sub-keys could be found. Also the the correct key is not even in the top key predictions. Figure 22 shows how correlation values for different key hypotheses change with different number of applied power traces in this attack. We can see that although for correlation value for the right key guess is comparatively low initially for low number of traces, with the increase of traces it increases significantly. From the figure it can be seen

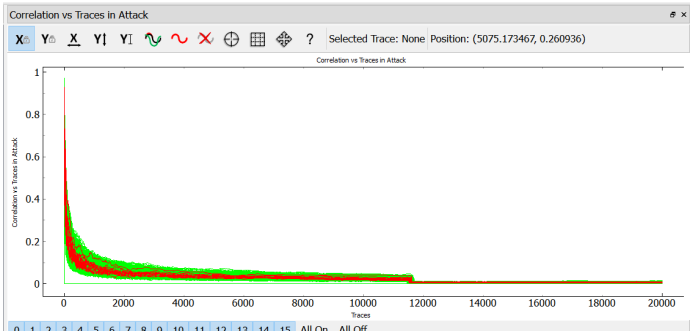

Fig. 24: HD leakage in Sbox output for countermeasure implementation

that the correlation values for correct key never got higher than those values for other key guesses even if when 20k traces were applied. It clearly shows that CPA could not reveal the correct key in this case. We can observe similar performance in Figure 23.

We also estimated the overhead added to the design because of the implemented countermeasure. Our countermeasure implementation requires minimum overhead to achieve significant side-channel robustness. Specifically, the countermeasure has increased the LUT utilization of the AES module by 42% and caused 14% power overhead.

*b) **Attempted Approaches:***

We also attempted to apply a masking implementation and a random clock frequency as side-channel countermeasures. The masking involves multiplication of a constant random matrix with the input key and plaintext to scramble to input to the AES module. As a result, the power signatures loose the correlation with the input key. The cipher output is multiplied again with the same constant random matrix to get the correct ciphertext.

The random clock frequency countermeasure can be implemented at the RTL level, providing the means to take advantage of the automated design flow. Moreover, countermeasure schemes such as hiding, masking, noise generators, etc., can add significant overhead to the design. Our proposed randomly changing clock frequency can be implemented as a tiny module, which takes the system clock as input, and we can get the clock signal with a random clock period at the output. This module's output is used as the clock input to the AES design. The clock frequency is randomized with a pseudo-random number generator. Our approach shows minimal overhead over the AES design and can contribute to significant resistance of the AES design from the power side-channel attack.

IV. Conclusion

HOST 2022 Microelectronics Security Challenge: IP Security Track has been arranged to promote security awareness about intellectual properties among the students. As a challenge, the participants have been asked to implement an AES on a FPGA and perform side-channel and fault injection attacks. The challenge also includes the development of possible countermeasures against these threats. In this work, we addressed these challenges. We performed several differential power side-channel attacks. We could successfully recover the key with around 2k power traces using CPA with HD model. We also performed ML-based attack and extracted the key with lower number of traces. we successfully extracted the 3^{rd} byte of 10^{th} round key using clock-glitch fault injection attack paired with differential fault analysis (DFA). In addition, we proposed several possible countermeasures to mitigate side-channel and fault injection vulnerabilities. Moreover, we successfully implemented a low-overhead side-channel countermeasure. The countermeasure implementation is found to be enough resilient to remain protected against correlation-based attacks even with 20k power traces.

References

[1] J. Daemen and V. Rijmen, "Reijndael: The advanced encryption standard." *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, vol. 26, no. 3, pp. 137–139, 2001.

[2] S. Bhunia, S. Ray, and S. Sur-Kolay, *Fundamentals of IP and SoC security*. Springer, 2017.

[3] C. Giraud, "Dfa on aes," in *International Conference on Advanced Encryption Standard*. Springer, 2004, pp. 27–41.

[4] S. Mangard, M. Oswald, and T. Popp, *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Springer, 2007, other identifier: 0387308571.

[5] K. Tiri and I. Verbauwhede, "Securing encryption algorithms against dpa at the logic level: Next generation smart card technology," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2003, pp. 125–136.

[6] ——, "A logic level design methodology for a secure dpa resistant asic or fpga implementation," in *Proceedings Design, Automation and Test in Europe Conference and Exhibition*, vol. 1. IEEE, 2004, pp. 246–251.

[7] A. Barenghi, G. M. Bertoni, L. Breveglieri, M. Pellicioli, and G. Pelosi, "Fault attack on aes with single-bit induced faults," in *2010 Sixth International Conference on Information Assurance and Security*. IEEE, 2010, pp. 167–172.

[8] N. T. Inc., "Cw305 artix fpga target board," https://www.newae.com/products/NAE-CW305, accessed: 2022-06-25.

[9] ——, "Chipwhisperer-lite 32-bit capture board," https://www.newae.com/products/NAE-CWLITE-ARM, accessed: 2022-06-25.

[10] V. Babu Dara and P. Sankara Rao, "Design and implementation of galois field based aes-256 algorithm for optimized cryptosystem," *International Journal of Science and Research (IJSR)*, vol. 3, 2014.