Three Eyed Raven: An On-chip Side Channel Analysis Framework for Run-time Evaluation

M Dhilipkumar, Priyanka Bagade, Debapriya Basu Roy
Department of Computer Science & Engineering, Indian Institute of Technology Kanpur, India
{dhilip, pbagade, dbroy}@cse.iitk.ac.in

***Abstract*—Side-channel attacks exploit the physical leakages from hardware components, such as power consumption, to break secure cryptographic algorithms and retrieve their secret key. Evaluating implementations of cryptographic algorithms against such analysis is crucial but traditional frameworks require expensive external devices like oscilloscopes, making the process expensive and time-consuming. Recent advancements in on-chip sensors offer a cost-effective, fully on-chip SCA framework, eliminating the need for external devices. In this paper, we propose *Raven*, an on-chip SCA framework with hardware implementations of Test Vector Leakage Assessment (TVLA), Correlation Power Analysis (CPA), and Deep Learning-based Leakage Assessment (DL-LA), for run-time evaluation of cryptographic implementations. RAVEN leverages on-chip sensors to efficiently assess side-channel security, without requiring any external measurement devices or any customized evaluation platform. Our proposed hardware implementations of TVLA, CPA, and DL-LA are lightweight and the entire architecture including the sensors can fit within the lightweight and low-cost AMD-Xilinx PYNQ FPGA platform. The proposed framework is verified on an FPGA implementation of AES-128 and the corresponding result of TVLA, CPA, and DL-LA closely matches with these algorithm's software implementation while requiring significantly less time and storage.**

I. Introduction

Side-channel attacks utilize physical information like power consumption, electromagnetic radiation, timing of cryptographic implementations to recover their secret keys. This makes secure cryptographic algorithms like AES, Elliptic Curve Cryptography (ECC) vulnerable. As a result, cryptographic algorithms must undergo rigorous side-channel evaluation to detect potential side-channel leakage. A typical side-channel analysis framework includes high-bandwidth digital/mixed-signal oscilloscopes and customized FPGA boards, such as the `CW305` from ChipWhisperer [1] or `SAKURA-G` [2], which capture side-channel traces with high signal-to-noise ratios (SNR). While effective, this setup is also costly and time-consuming due to the need for specialized boards and the slow transfer of recorded traces to a CPU for evaluation, making side-channel assessments a time-consuming process.

Support for this work was provided by C3i (cybersecurity and cybersecurity for Cyber-Physical Systems) Innovation Hub, IIT Kanpur , "Information Security Education and Awareness (ISEA) Project Phase-III", Science and Engineering Research Board (SERB) through Startup Research Grant (SERB /CS /2022377) and Ministry of Electronics and Information Technology (MEITY /CS /2024250)

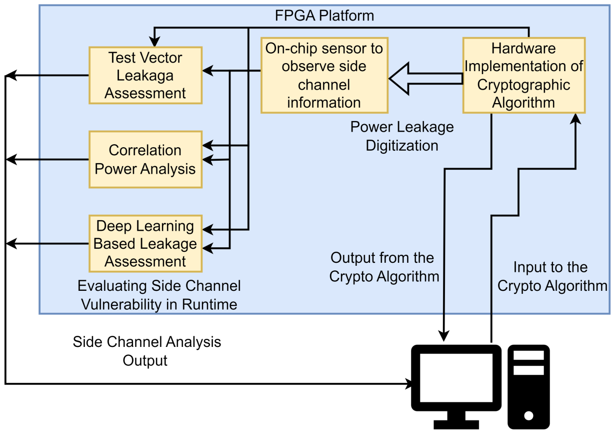

FPGA Platform
Test Vector Leakaga Assessment
On-chip sensor to observe side channel information
Hardware Implementation of Cryptographic Algorithm
Power Leakage Digitization
Correlation Power Analysis
Deep Learning Based Leakage Assessment
Evaluating Side Channel Vulnerability in Runtime
Output from the Crypto Algorithm
Input to the Crypto Algorithm
Side Channel Analysis Output

Fig. 1: Proposed On-Chip Side Channel Evaluation Framework *RAVEN*

FPGA-based sensors for recording side-channel leakage have recently gained attention in the literature, primarily targeting remote attacks on cloud-shared FPGAs (e.g., Amazon AWS). Common sensors include Ring Oscillator (RO) [7], Time-to-Digital Converter (TDC) [6], and Routing Delay Sensors (RDS) [14]. These sensors have been used to create side-channel recorders, hardware Trojans, and covert channels to leak secret information about the cryptographic algorithms. In this paper, we introduce *RAVEN*, an on-chip side-channel evaluation framework leveraging FPGA-based sensors to analyze cryptographic vulnerabilities using the *RDS* circuit. Our framework performs real-time Test Vector Leakage Assessment (TVLA), Correlation Power Analysis (CPA), and Deep Learning-based Leakage Assessment (DL-LA) [12] directly on-chip, eliminating the need for external oscilloscopes and specialized hardware. Our contribution includes

- **Low-Cost On-Chip Evaluation Framework**: We propose *RAVEN*, a fully on-chip side-channel evaluation platform utilizing FPGA-based sensors, specifically RDS [14], for capturing side-channel leakage. This setup eliminates the need for external devices like oscilloscopes or customized FPGA boards, streamlining the analysis process and significantly reducing costs and time requirements to perform side-channel leakage assessment.
- **Hardware Implementation of TVLA, CPA and DL-LA**: Resource efficient run-time TVLA is designed in pipeline and faster than its software counterpart. To the best of our knowledge, this is the first work to integrate

Correlation Power Analysis (CPA) directly into hardware alongside the RDS sensor. The CPA architecture is lightweight, pipelined, and optimized for performance, while the DL-LA [12] module allows for leakage detection.
- **Validation and Performance**: We have validated our proposed methodology on an FPGA implementation of AES-128, executed on the AMD-Xilinx PYNQ board. The results for TVLA, CPA, and DL-LA obtained from our proposed on-chip side-channel evaluation framework match closely with the software implementations of these algorithms, proving its' effectiveness.

The rest of the paper is constructed as below. Section II discusses the state-of-the-art works on on-chip leakage measurement and side-channel analysis, while section III focuses on the required background studies. In section IV we have proposed our hardware architecture for computing TVLA, CPA and DL-LA. Section V shows the experimental results and finally we conclude the paper in section VI.

II. Related Work

In this section, we briefly review related works on FPGA-based sensors and on-chip side-channel measurement frameworks. These sensor circuits were designed to capture side-channel information from cryptographic algorithms without requiring physical access to the implementation. The earliest sensors used for this were Ring Oscillator (RO)-based sensors and Time-to-Digital Converter (TDC) sensors, which have been employed in power-based side-channel attacks [7], [6], [18] and covert channels [15], [4]. Additionally, TDC and RO-based sensors have been utilized to detect hardware Trojans [8], [17]. However, RO-based sensors are relatively easy to detect in hardware [9], [10], limiting their effectiveness in side-channel attacks [11]. TDC sensors, constructed using fast carry chains, are highly effective for capturing side-channel information but require careful placement near the target cryptographic circuit, and improper placement can significantly degrade their effectiveness. To address these limitations, the Routing Delay Sensor (RDS) was introduced in [14]. Comparative results from correlation power analysis (CPA) attacks involving RDS variants and TDC demonstrated that RDS consistently outperforms other sensors. Given its clear advantages in side-channel analysis, we have built the proposed *RAVEN* framework using RDS.

In [13], the authors proposed a hardware implementation of TVLA using online computation for mean and variance, with a parallel design where each sample point required a separate module. However, they used a linear regression model to generate traces instead of an on-chip sensor. Later, [5] introduced a pipelined version of TVLA, allowing mean and variance updates after every k traces from a TDC sensor. In our work, we adopt this pipelined approach but optimize the architecture for reduced resource utilization.

CPA has been found quite effective in exploiting side-channel leakages to recover the secret keys of cryptographic algorithms. Several studies have focused on enhancing CPA's efficiency. One approach involves dimensionality reduction of side-channel traces to improve the signal-to-noise ratio (SNR). For instance, [3] used Linear Discriminant Analysis (LDA) for this purpose, showing that an optimal attack on multivariate traces is equivalent to an attack on a transformed univariate trace. In our work, we adopt LDA to compress traces from the RDS, ensuring efficient processing while retaining attack efficacy. This paper presents the first hardware-based CPA design, advancing side-channel analysis.

Deep Learning Leakage Assessment (DL-LA) was introduced in [12], where a convolutional neural network (CNN) is used to detect side-channel vulnerabilities instead of traditional TVLA. The key advantage of DL-LA is that it eliminates the need for pre-processing side-channel traces for higher-order analysis. In this paper, we propose a hardware-based implementation of DL-LA, which sets our proposed framework apart from existing state-of-the-art solutions.

III. Background

In this section, we are going to provide a brief background on the RDS circuit, along with the necessary mathematical details for understanding TVLA, CPA, and DL-LA.

A. RDS: Routing Delay Sensor

TDC and RO-based sensors are sensitive to placement in the circuit. To address this, the authors of [14] proposed a Routing Delay Sensor (RDS) that offers high accuracy without placement constraints. The RDS comprises two main components: *initial delay* (calibrating the sensor) and *routing delay* (capturing side-channel information). The *initial delay* is further divided into *coarse delay* (LUT-based) and *fine delay* (fast carry chain), while the routing resources connects these outputs to sampling registers which causes *routing delay*. Unlike TDC and RO sensors, RDS measures signal routing delays, indirectly revealing power consumption during cryptographic operations. In [14], RDS was validated on an AES-128 implementation, where CPA was successfully applied to extract the secret key, demonstrating RDS's potential for side-channel analysis.

B. CPA: Correlation Power analysis

Correlation Power Analysis (CPA) exploits the relationship between a cryptographic device's power consumption and the key-dependent data it processes. By measuring power during cryptographic operations and correlating it with predicted power models, an attacker can deduce secret keys. The success of CPA relies on an accurate power model, with the Hamming Distance (HD) model being effective for hardware implementations. Pearson's correlation coefficient is commonly used to distinguish between potential keys.

To retrieve the secret key, we compute Pearson's correlation coefficient ρ between every column of the power trace matrix ($trace$) with every column of the matrix generated by the hypothetical power model ($hypo$). The value of ρ between

i^{th} column of matrix $trace$ and j^{th} column of matrix $hypo$ is given by

$$\rho = \frac{cov(trace[i], hypo[j])}{\sqrt{var(tarce[i]) \times var(hypo[j])}} \tag{1}$$

Here cov indicates covariance and var denotes variance. If the attack is successfully administered, the correct key will have higher correlation value compared to wrong key guesses and will be accurately classified.

C. TVLA: Test Vector Leakage Assessment

Test Vector Leakage Assessment (TVLA) is a statistical method used to detect side-channel leakage in cryptographic implementations. It works by comparing power traces recorded during cryptographic operations executed under different conditions. There are two common approaches:

1) Fixed vs. Random Test: In this test, the cryptographic algorithm is executed repeatedly with a fixed plaintext and with random plaintext, whereas the secret key is kept same for both cases. The goal is to compare the d^{th} order statistical moment of these two sets to determine whether the traces have d^{th} order leakage or not.
2) Fixed vs. Fixed Test: This variant compares the side-channel traces from two fixed inputs. The rest of the method remains the same.

Welch's t-test: Welch's t-test is a statistical tool that compares the means of two groups or the difference between one group's mean and a standard value. It's also known as a t-statistic or t-distribution. The Welch's t-test statistic is calculated as:

$$t = \frac{\mu_A - \mu_B}{\sqrt{\frac{\sigma_A^2}{N_A} + \frac{\sigma_B^2}{N_B}}} \tag{2}$$

where μ and σ^2 denote their sample mean and variance and N denotes no. of measurements in each set. If $|t| > 4.5$, it suggests that, with 0.99999 confidence, the two sets have different mean, and hence side channel leakage is detected.

D. Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis (LDA) is a statistical method used for dimensionality reduction and classification. LDA reduces multi-dimensional traces to a single data point, as shown in [3], where the optimal attack on multivariate traces is equivalent to one on LDA-transformed traces. By maximizing variance between classes and minimizing within-class variance, LDA enhances the signal-to-noise ratio, thus maximizing accuracy with smaller dimension. In equation $X_{d,q} = \alpha_d Y_q(k^*) + N_{d,q}$, d is the sample index, q is the trace index, $X_{d,q}$ represents the d^{th} sample of q^{th} recorded power trace, $Y_q(k^*)$ represents the hypothetical power trace computed for k^* key byte guess. $N_{d,q}$ is random measurement noise. The monovariate trace used for the attack is derived from the multivariate trace by the following way

$$\tilde{x}_q = \frac{(\alpha^D)^T \Sigma^{-1}}{(\alpha^D)^T \Sigma^{-1} \alpha^D} \times x_q^D \qquad (q = 1, ..., Q)$$

where Σ is the covariance matrix of noise N. For the rest of the paper, we will term the left multiplicand as the LDA coefficients.

E. Deep learning for leakage assessment

The use of deep learning models for detecting side-channel leakage was introduced in [12], where classifiers like Multi-Layer Perceptron (MLP) and Convolutional Neural Networks (CNN) were applied to differentiate between two sets of power traces similar to TVLA. The ability of the models to distinguish between the two datasets indicates the presence of leakage. This reduces the need for pre-processing where SCA traces are misaligned or when the leakage is multivariate. DL-LA was tested against the PRESENT block cipher with and without SCA countermeasures. Their results shows that DL-LA can detect leakages with multivariate nature with lesser number of traces than required by traditional leakage detection methods.

IV. Hardware implementations

In this section, we will discuss the implementation of CPA, TVLA and DL-LA in hardware and the necessary optimizations needed for efficient hardware implementations. All the hardware architectures have been implemented on the PYNQ board of AMD-Xilinx FPGA and the overhead results are obtained after post-place and route implementations. Our case study is built on an RDS circuit that monitors the side-channel leakage of an AES-128 architecture. The RDS requires 32 LUTs and 512 registers whereas the AES-128 requires 1478 LUTs and 516 registers. The width of the sensors in our implementation is 512 and the leakage is calculated by calculating the Hamming weight of the sensor output.

The workflow of the *RAVEN* architecture is as below.

- The user selects in which mode the RAVEN should operate. At this moment, RAVEN can be configured to compute TVLA, DL-LA, or CPA.
- The DUT gets executed once and the sensor records its side channel information and stores it in BRAMs. At this point, clock to *RAVEN* is disabled so it does not impact the sensor outputs.
- After each execution of the DUT, clock to *RAVEN* is enabled and it does run-time computation for either TVLA, CPA or DL-LA depending upon the user choice.
- This process continues until sufficient information is recorded regarding the side-channel vulnerability of the design under test.

A. TVLA in hardware

In this section, we will discuss our methodology behind the proposed TVLA/t-test hardware architecture. We will be using fixed vs random test for our t-test implementation. Let set A denote the random plaintext traces and set B denote the fixed plaintext traces. As the fixed plaintext traces will not be varying much, the assumption is $\sigma_B^2 = 0$. So we have

$$t = \frac{\mu_A - \mu_B}{\sqrt{\frac{\sigma_A^2}{N}}} \tag{3}$$

As implementing division is expensive in hardware, the following way is used for calculating leakage vulnerability. For leaky implementation, t-value needs to be greater than 4.5. Therefore for leaky implementation , the following relation will hold. This is found by squaring both sides of Eq.(3).

$$20.25 \times \sigma_A^2 > (\mu_A - \mu_B)^2 \times N \tag{4}$$

We can use this inequality condition to achieve our goal. Answer is 1 when condition is satisfied (Leakage detected), 0 when condition fails (no leakage detected). Same method is used in [13], although their margin of error was ± 20, whereas our results have no such errors due to the following optimizations.

We need to compute μ_A, μ_B, σ_A to check the inequality of Eq. (4).

$$\mu_N = \sum_{k=1}^{N} \frac{x_k}{N} \ , \ \sigma_N^2 = \sum_{k=1}^{N} \frac{x_k^2}{N} - \left(\sum_{k=1}^{N} \frac{x_k}{N} \right)^2 \tag{5}$$

The value of $(1/N)$ will be provided from the software side to remove the costly division operation. Our proposed architecture is completely pipelined where each sample point is passed to the t-test hardware module one by one and the values are updated for each sample point. The pipeline starts with computing M_1 and M_2 which are defined as follows:

$$M_1 = \sum_{k=1}^{N} \frac{x_k}{N} \ , \ M_2 = \sum_{k=1}^{N} \frac{x_k^2}{N} \tag{6}$$

The hardware architecture of the TVLA module is shown in Fig. 2. We use BRAMs to keep track of M_1, M_2 for fixed and random sets. For every new trace that is generated by the RDS and stored in the sensor BRAM, we first multiply each sample point with $\frac{1}{N}$ and then they are added with the previous value to generate M_1 and M_2. Note that with this method we get the correct value of mean and variance after every N trace where N is chosen by the user. At the end of N traces, $\mu_N = M_1$ and $\sigma_N^2 = M_2 - (M_1)^2$.

As we are not computing the running mean and variance, updating the means and variance values for different values of N is challenging. We resolve this issue by following strategy:

- Suppose we have computed M_1 and M_2 for N_1 number of traces and now we want to compute M_1 and M_2 for $N_1 + N_2$ number of traces.
- Update the M_1 and M_2 by computing $M_1 = M_1 \times \frac{N_1}{N_1+N_2}$ and $M_2 = M_2 \times \frac{N_1}{N_1+N_2}$.
- Then keep updating M_1 and M_2 for every new trace by multiplying the summation result by $\frac{1}{N_1+N_2}$.

Once we have the M_1 and M_2 value for the random set and the M_1 value for the fixed set after N traces, we can easily compute Eq. (4) for each sample point and detect whether any sample is leaking or not. With the updating strategy described above, we can also detect leaky samples after any arbitrary $N + k$ traces where k is chosen by the user.

For accurate results, fixed point arithmetic with 18 bits for the fraction part is used. The main idea behind the computation of M_1 and M_2 is that the correction value for the M_1 and M_2 for every new trace (i.e) $\frac{x_k}{n}$ and $\frac{x_k^2}{n}$ are always less than 1.

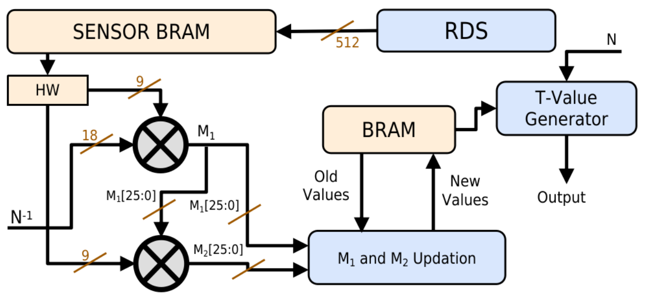


Fig. 2: Hardware Implementation of TVLA Architecture

This is true under the assumption that the N is greater than the sample values produced by the sensors which will be true in most scenarios. For example, for our implementation with an RDS sensor with a width of 512, the maximum value that each sample point can have is at most 512. This phenomenon allows us to give more bits for fractional parts thus increasing the accuracy.

TABLE I: Resource utilisation of TVLA module

	LUT	Reg.	Slices	BRAM	DSP	Freq.(MHz)
[5]	1580	1430	-	8	14	-
[13]	199	222	110	2	24	72
Our impl.	1881	308	506	1.5	10	100

After each execution of the cryptographic algorithms, we will require $s + 5$ cycles (s being no, of sample points to compute M_1 and M_2. The updation of M_1 and M_2 for different values of N will also require $s + 5$ cycles due to the pipeline nature of the architecture. Once the leakage detection is requested by the user for a given sample point, the tool returns 1 or 0 as per Eq. 4 within 2 clock cycles.

In Table I, the resource utilisation by our design, [5] and [13] are compared. We require fewer DSPs and BRAMs compared to existing designs while slight increase in LUTs when compared with [5]. Even though [13] requires fewer LUTs than us, they will require s such modules to detect the leakage for s sample points and therefore the total overhead will be significantly high. Note that our implementation will remain same for different cryptographic algorithms which shows the generic nature of the architecture.

B. Optimizations for CPA Architecture

In our proposed framework, we have used LDA [3] on the original traces to convert them into monovariate traces to reduce the complexity and increase the efficiency of CPA implementation. For these, the traces are first trained to compute LDA coefficients [3] of dimension $1 \times s$ which when multiplied by a trace of dimension $s \times 1$ produces a monovariate traces. To find the correct key, we compare the correlation value and find the key byte with maximum correlation. The correlation value for key guess k_1 is given by

$$Correlation(k_1) = \frac{cov(A, B_{k_1})}{\sqrt{var(A) \times var(B_{k1})}} \tag{7}$$

where A is the mono variate power trace from LDA and B_{k_1} is the hypothetical power trace for key guess k_1. As A

is the monovariate recorded power trace, it will remain the same for all the key guesses. So we can neglect $var(A)$ in the equation as it is the same for all key guesses. We define a term Pseudo-correlation which will be used instead of real correlation. Neglecting $var(A)$ and rearranging the terms in Eq.(7) brings us to the pseudo-correlation for key byte k_1 as

$$PC(k_1) = \frac{Q\left(\sum_{i=1}^{Q} a_i b_i\right) - \sum_{i=1}^{Q} a_i \sum_{i=1}^{Q} b_i}{\sqrt{Q\left(\sum_{i=1}^{Q} b_i^2\right) - \left(\sum_{i=1}^{Q} b_i\right)^2}} \tag{8}$$

where a_i is mono variate trace from LDA and b_i is a hypothetical power trace. This expression of PC is the final value we will be using instead of Pearson's correlation coefficient. We introduce PC_{num} and PC_{den}, which are computed in the hardware.

$$PC_{num}(k_1) = Q\left(\sum_{i=1}^{Q} a_i b_i\right) - \sum_{i=1}^{Q} a_i \sum_{i=1}^{Q} b_i \tag{9}$$

$$PC_{den}(k_1) = Q\left(\sum_{i=1}^{Q} b_i^2\right) - \left(\sum_{i=1}^{Q} b_i\right)^2 \tag{10}$$

These values are returned to the software and can be used for pseudo-correlation computation and used for finding the best key guess. The user can stop the computation at any number of traces and can compute the pseudo correlation. This way avoids any storage of traces and saves time in computation.

C. CPA hardware architecture

The hardware architecture for the CPA is shown in Fig. 3. As our case study is built on AES-128, recovery of each key byte involves 256 key guesses. For CPA, we propose a pipelined design to handle all 256 key guesses efficiently, as shown in Fig. 3. The pipeline begins with the calculation of the hypothetical power leakage of the Design-under-Test (DUT). Hamming distance model is used for hypothetical power leakage computation. Once the hypothetical power leakage is computed, the next stage of the pipeline handles two key operations: calculating the square of the hypothetical power trace and multiplying it with the mono variate power trace, coming from the LDA module. These are performed using two parallel multipliers. To store intermediate values for all 256 key guesses, three BRAMs are used. Assume that A represents the collected power traces and B the computed hypothetical power leakage. We track sums of $\sum a_i$, $\sum b_i$, $\sum b_i^2$, and $\sum a_i b_i$. Since $\sum a_i$ is the same for all key guesses, it doesn't require BRAM storage, leaving the other three sums to be stored in the BRAMs. At this stage, the values of b_i^2 (the square of the hypothetical power trace) and $a_i b_i$ (the product of LDA output and hypothetical power trace) are computed. For each key guess, the previously stored values in BRAMs are retrieved, updated with the new results, and written back to the same address. These stored values are used to compute PC_{num} and PC_{den}. To reduce the overhead, fixed arithmetic is used with 14 bits for fraction part. The overhead of the different modules involved in CPA is shown in Table II. To use CPA for other cryptographic tools, the only change required is the hypothetical power trace generator. Rest of the CPA architecture remains the same.

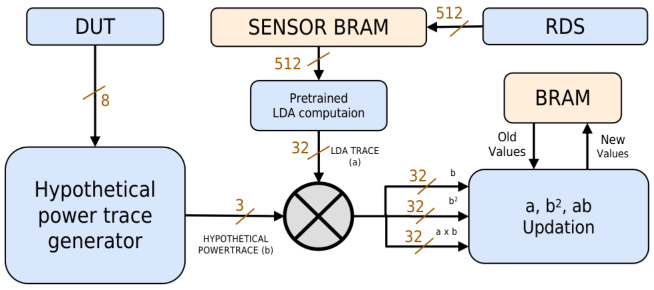


Fig. 3: Hardware Architecture of CPA

TABLE II: Resource utilisation of CPA and LDA module

	LUT	Registers	Slices	BRAM18	Freq.(Mhz)
CPA	330	201	194	4	10
LDA	1405	116	499	1	10
Total	1735	317	693	5	10

D. DL-LA hardware architecture

In deep learning-based leakage assessment [12], we try to distinguish between two different sets of side channel traces with either MLP or CNN model. In our implementation, we have used CNN due to its superior classification capability compared to MLP. We have opted for a lightweight model with a single convolutional layer followed by a dense layer. The below provided model was tested in software using power traces and proved effective for leakage detection. The CNN model is also pre-trained in software with traces acquired from RDS.

```
model = Sequential([
    Reshape((140,1), input_shape = (140,)),
    Conv1D(filters=3, kernel_size=3, \
           strides=1, input_shape=(140,1),\
           activation='relu'),
    Flatten(),
    Dense(1, activation='sigmoid')])
```

As our implementation platform in PYNQ board with limited number of DSP blocks, we need to devise a lightweight architecture for prediction of leakage through the DL-LA method. Thus, in our architecture, a limited number of multipliers are allocated to each layer and the necessary products are computed sequentially. The detailed architecture of each layer is explained in the following sections.

1) Convolution Layer: The convolutional layer applies filters to extract key features from the input data, where each filter has learnable weights. The stride determines the step size for filter movement. In our design, three filters with three weights each are used, requiring nine multipliers. The filters move with a stride of 1, covering the entire input to capture critical patterns. ReLU activation is applied to the output of the convolution layer. $ReLU(x) = max(x, 0)$. The architecture is illustrated in Fig.4.

2) Dense layer: A dense (fully connected) layer connects every output from the previous layer to each node in the next layer. We store the outputs from the 3 filters in separate BRAMs and similarly store the weights of the dense layer in 3 additional BRAMs. We use 3 multipliers to handle weight multiplication, ensuring efficient resource utilization while processing the dense layer.

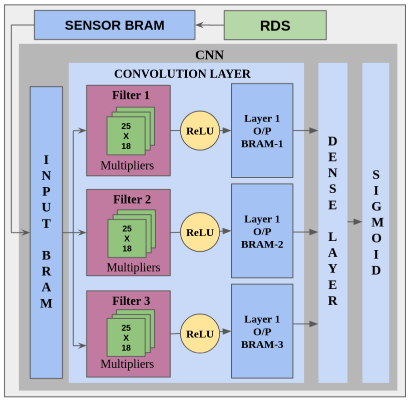


Fig. 4: DL-LA Hardware architecture.

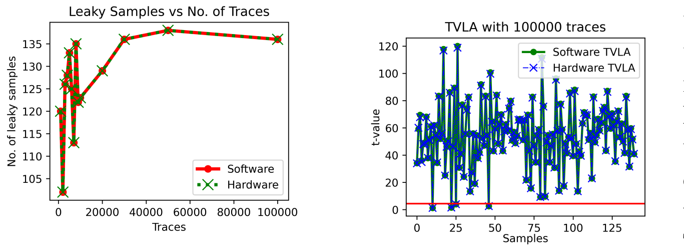
3) Sigmoid activation:: The sigmoid activation function is commonly used in neural networks to map input values to a range between 0 and 1. The sigmoid function is mathematically defined as:

$$\sigma = \frac{1}{1+e^{-x}}$$

In [16], sigmoid function was implemented in hardware using the piece-wise linear approximation method in the following way. If $x < 0$ then $f(-x) = 1 - f(x)$. We calculate for positive x and apply this formula.

$$f(x) = \begin{cases} 1 & |x| \geq 5.0, \\ 0.03125 * |x| + 0.84375, & 2.375 \leq |x| < 5.0, \\ 0.125 * |x| + 0.625, & 1.0 \leq |x| < 2.375, \\ 0.25 * |x| + 0.5, & 0 \leq |x| < 1.0 \end{cases} \quad (11)$$

The implemented CNN hardware requires 1207 LUTs, 417 Registers, 413 Slices, 3.5 BRAMs, 13 DSPs and operates at 100 MHz of frequency.



(a) Number of leaky sample points in hardware TVLA implementation and software TVLA

(b) T-value computed in hardware TVLA and software TVLA

Fig. 5: Comparison between Hardware and Software Implementation of TVLA

V. Experimental Results

In this section, we will provide the results and the performance of the proposed tools. All the experiments are conducted in AMD-Xilinx Pynq platform. The design under test (DUT) is an iterative implementation of AES-128. For TVLA, we have compared the results of the hardware implementation with a software implementation in two-way. First, for the given AES-128 implementation, we have compared the number of leaky samples obtained by the proposed hardware implementation with that of software implementations. This comparative result is shown in Fig. 5a. The results of hardware matches perfectly with the software results. As a second and even strong evaluation, t-values have been computed for 100000 traces in Fig.5b. The results of hardware matches with the software results perfectly. These two evaluations prove the correctness of the TVLA implementation. No sample points are misclassified as leaky or non-leaky even for 100000 traces thus making our implementation fail-proof for significant number of traces. For CPA, the tool ran for 6000 plaintext encryption and the intermediate values are used to calculate the pseudo-correlation which is provided in Fig.6. In Fig.6, the pseudo correlations for all key guesses are plotted for the first 6 key bytes for space constraint in the paper. All key bytes were broken with 6000 traces. The stems marked in red are the correct key guesses. The results were obtained way faster (approximately 10 minutes) than the traditional CPA setup which takes hours just for trace collection. For DL-LA, the traces for two different plaintext were obtained. The CNN model was pre-trained in the software and the pre-trained weights were hard-coded in the hardware. The accuracy of the software model was 94.46%. Now the hardware implementation of DL-LA is evaluated in run time against same two set of plaintexts used for training. The hardware implementation was able to classify the two set of traces with 91.42% accuracy for 4000 traces. The confusion matrix for the predictions is given in Table III. '0' and '1' represent two different sets of plaintexts.

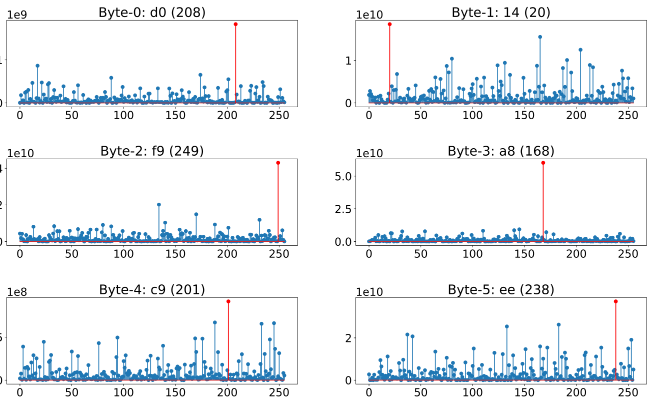


Fig. 6: Pseudo Correlation vs Key guesses for all bytes

TABLE III: Confusion matrix for DL-LA prediction in hardware

	Predicted 0	Predicted 1
Actual 0	1892	108
Actual 1	235	1765

VI. Conclusion

In this paper we have presented a cheap and efficient on-chip laboratory for side channel analysis. It consists of RDS, on-chip sensor for power trace measurement, TVLA, CPA and DL-LA hardware architectures. Results shown by these tools proves that this framework is as accurate as its software counterparts. The proposed *RAVEN* framework allows for a runtime evaluation of the cryptographic tools without any external devices. This work will pave path for future works on a fully independent on-chip laboratory for side channel analysis.

REFERENCES

[1] Cw305 artix fpga target - newae hardware product documentation. https://rtfm.newae.com/Targets/CW305%20Artix%20FPGA/. (Accessed on 09/12/2024).

[2] Sakura. https://satoh.cs.uec.ac.jp/SAKURA/hardware/SAKURA-G.html. (Accessed on 09/12/2024).

[3] Nicolas Bruneau, Sylvain Guilley, Annelie Heuser, Damien Marion, and Olivier Rioul. Less is more - dimensionality reduction from a theoretical perspective. Cryptology ePrint Archive, Paper 2016/359, 2016.

[4] Ilias Giechaskiel, Kasper Bonne Rasmussen, and Jakub Szefer. C3apsule: Cross-fpga covert-channel attacks through power supply unit leakage. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 1728–1741, 2020.

[5] Ognjen Glamočanin, Louis Coulon, Francesco Regazzoni, and Mirjana Stojilović. Built-in self-evaluation of first-order power side-channel leakage for fpgas. In *Proceedings of the 2020 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 204–210, 2020.

[6] Dennis R. E. Gnad, Cong Dang Khoa Nguyen, Syed Hashim Gillani, and Mehdi B. Tahoori. Voltage-based covert channels using fpgas. *ACM Trans. Des. Autom. Electron. Syst.*, 26(6), jun 2021.

[7] Joseph Gravellier, Jean-Max Dutertre, Yannick Teglia, and Philippe Loubet-Moundi. High-speed ring oscillator based sensors for remote side-channel attacks on fpgas. In *2019 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, pages 1–8, 2019.

[8] Dylan Ismari, Jim Plusquellic, Charles Lamech, Swarup Bhunia, and Fareena Saqib. On detecting delay anomalies introduced by hardware trojans. In *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–7. IEEE, 2016.

[9] Jonas Krautter, Dennis RE Gnad, and Mehdi B Tahoori. Mitigating electrical-level attacks towards secure multi-tenant fpgas in the cloud. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 12(3):1–26, 2019.

[10] Tuan Minh La, Kaspar Matas, Nikola Grunchevski, Khoa Dang Pham, and Dirk Koch. Fpgadefender: Malicious self-oscillator scanning for xilinx ultrascale+ fpgas. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 13(3):1–31, 2020.

[11] Shayan Moini, Xiang Li, Peter Stanwicks, George Provelengios, Wayne Burleson, Russell Tessier, and Daniel Holcomb. Understanding and comparing the capabilities of on-chip voltage sensors against remote power attacks on fpgas. In *2020 IEEE 63rd International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 941–944, 2020.

[12] Thorben Moos, Felix Wegener, and Amir Moradi. Dl-la: Deep learning leakage assessment: A modern roadmap for sca evaluations. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 552–598, 2021.

[13] Souvik Sonar, Debapriya Basu Roy, Rajat Subhra Chakraborty, and Debdeep Mukhopadhyay. Side-channel watchdog: Run-time evaluation of side-channel vulnerability in fpga-based crypto-systems. *Cryptology ePrint Archive*, 2016.

[14] David Spielmann, Ognjen Glamočanin, and Mirjana Stojilović. Rds: Fpga routing delay sensors for effective remote power analysis attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2023(2):543–567, 2023.

[15] Shanquan Tian and Jakub Szefer. Temporal thermal covert channels in cloud fpgas. In *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, FPGA '19, page 298–303, New York, NY, USA, 2019. Association for Computing Machinery.

[16] Ivan Tsmots, Oleksa Skorokhoda, and Vasyl Rabyk. Hardware implementation of sigmoid activation functions using fpga. In *2019 IEEE 15th International Conference on the Experience of Designing and Application of CAD Systems (CADSM)*, pages 34–38. IEEE, 2019.

[17] Xuehui Zhang and Mohammad Tehranipoor. Ron: An on-chip ring oscillator network for hardware trojan detection. In *2011 Design, Automation & Test in Europe*, pages 1–6. IEEE, 2011.

[18] Mark Zhao and G. Suh. Fpga-based remote power side-channel attacks. pages 229–244, 05 2018.