Side-Channel Watchdog: Run-Time Evaluation of Side-Channel Vulnerability in FPGA-Based Crypto-systems

Souvik Sonar, Debapriya Basu Roy, Rajat Subhra Chakraborty and Debdeep Mukhopadhyay

Secured Embedded Architecture Laboratory,
Indian Institute of Technology, Kharagpur, India,
souvik.sonar@iitkgp.ac.in,
{deb.basu.roy, rschakraborty, debdeep}@cse.iitkgp.ernet.in

Abstract. Besides security against classical cryptanalysis, its important for cryptographic implementations to have sufficient robustness against side-channel attacks. Many countermeasures have been proposed to thwart side channel attacks, especially power trace measurement based side channel attacks. Additionally, researchers have proposed several evaluation metrics to evaluate side channel security of crypto-system. However, evaluation of any crypto-system is done during the testing phase and is not part of the actual hardware. In our approach, we propose to implement such evaluation metrics on-chip for run-time side channel vulnerability estimation of a cryptosystem. The objective is to create a watchdog on the hardware which will monitor the side channel leakage of the device, and will alert the user if that leakage crosses a pre-determined threshold, beyond which the system might be considered vulnerable. Once such alert signal is activated, proactive countermeasures can be activated either at the device level or at the protocol level, to prevent the impending side channel attack. A FPGA based prototype designed by us show low hardware overhead, and is an effective option that avoids the use of bulky and inconvenient on-field measurement setup.

1 Introduction

Traditionally a crypto-implementation is evaluated against side channel vulnerability solely during the testing of the device, and once deployed in field, such evaluation is not always possible. However, it might happen that the user may need to update the underlying crypto-algorithm and perform the side channel vulnerability assessment again, which requires costly and sophisticated side channel information acquisition equipments to be brought to the location of deployment.

In this paper, we explore an alternative approach where such side channel vulnerability evaluation is done on-chip and is integrated with the crypto-implementation itself. Additionally, this design strategy allows the user to have a constant monitoring of side channel leakage, and could act as an alternative for

traditional side channel countermeasures which have large overhead for resource-constrained devices. Although our hardware implementation was performed on FPGA platform, the same principle can be applied to ASIC and SoC implementations also. In literatures, various metrics for side channel evaluation have been proposed and is used for the side channel vulnerability testing of the crypto-devices. *Test Vector Leakage Assessment Methodology (TVLA)* [2] is one of such metrics that is becoming increasingly popular due to its simplicity and recent international standardization. The proposed evaluation of side channel vulnerability on the device itself is achieved by integrating the TVLA hardware with the crypto-implementation.

The contribution of the present paper can be tabulated as follow:

- In this paper we consider the widely used (TVLA) [2] as the metric of choice for side channel vulnerability calculation. TVLA is a standard metric for evaluation of a crypto-implementation against first order correlation power attack, and can be easily extended against higher order attacks also [19]. The on-chip side channel vulnerability evaluator constantly monitors the side channel information leakage of the crypto- implementation, and raises an alarm whenever the TVLA value goes beyond safe threshold, making the system susceptible to correlation power attack, hence acting as a *watchdog*.
- Implementation of TVLA involves complex floating point operations and can contribute significantly to the area overhead of the design. In this paper, we have implemented a lightweight fixed-point TVLA computation module, which offers excellent accuracy when compared with an implementation of a floating-point computation module, at a fraction of the hardware overhead.
- Various corrective decision can be made after being alerted by the side channel evaluator to thwart the adversary attacks. In this paper, we outline few of such corrective measures. For example, decision like changing the secret key information or manipulating the algorithm parameters can be made to prevent the impending attack.

We provide a brief overview of TVLA and side channel simulation strategies in Section 2. An efficient simulation of side channel traces is necessary as we need to incorporate this into the implementation for correct computation of TVLA. In Section 3, we discuss the side channel simulation strategy with detailed analysis and results. This is followed by comprehensive description of on-chip TVLA computation hardware in Section 4, where we rearrange the steps for easy integration to the crypto-hardware. Next, in Section 5, we provide the architectural description and resource requirement of the developed hardware module. In Section 6, we give an experimental validation of the proposed methodology by performing a comparative analysis of TVLA computation on actual and predicted side channel traces. Finally in Section 7, we discuss the corrective measures to prevent side channel adversary from getting access to the secret key value followed by conclusion in Section 8.

2 Preliminaries

In this section, we will introduce TVLA and side channel simulation methodology briefly.

2.1 TVLA

The methodology involves measuring side channel information (e.g. power traces, EM traces) while performing cryptographic operation (e.g. AES) with a pre-specified set of input vectors viz. plaintext and fixed key and performing *Welch t-test* on side channel measurements [8]. Let S_A and S_B be the two sets of measurements drawn for a DUT at any instant. Let μ_A and μ_B denote their sample means, σ_A^2 and σ_B^2 denote their sample variance, N_A and N_B denote the number of measurements in each set. The *t-test statistic* is given by:

$$t = \frac{\mu_A - \mu_B}{\sqrt{\frac{\sigma_A^2}{N_A} + \frac{\sigma_B^2}{N_B}}} \tag{1}$$

The null hypothesis is that the two samples come from the same population, implying that their means μ_A and μ_B are the same. If the power traces in the two subsets are statistically different with high confidence, then information leakage is present and the device is vulnerable to side-channel attacks. The null hypothesis is rejected when the computed t-value exceeds the threshold value of 4.5, i.e., $\mid t \mid > 4.5$. This threshold leads to a confidence level > 0.99999. The computation of TVLA can be performed in two ways [2] [8]. In our approach we are using *Non-Specific Leakage Test* of the first order for computing on-chip TVLA value. For *non-specific t-test*, S_A is collected with a fixed plaintext and S_B is collected with random plaintexts drawn from uniform distribution. The *t-test statistic* is then calculated by computing the mean and variances for both the data sets. If at any point of time the t-test statistic exceeds the safe threshold, i.e. if $t_{SAFE} > |4.5|$, the hardware is said to be under threat of side channel analysis. The details of the test are illustrated in [2] [19].

2.2 Simulating Side Channel Traces

The success of any side channel attack depends upon the Signal-to-Noise ratio (SNR) of the acquired side channel traces. Hence, the most important part of a successful side channel attack is accurate and precise side channel trace acquisition setup, requiring costly equipments like high-end oscilloscopes and sophisticated probes. Trace acquisition from a physical device using these costly equipments in real time is time consuming, as the trace acquisition speed is bounded by the speed of the crypto-algorithm execution in the physical device. To tackle this issue, one possible alternative is to develop a software which will generate the side channel traces using a simulation model. This approach makes the side channel evaluation procedure faster, allowing the evaluator more freedom to carry out side channel evaluation for different leakage models and attack methodologies.

To estimate power consumption of a chip, different EDA vendors provide different CAD software tools like *PrimeTime* [6], *NC-Sim* [4], *ModelSim* [10]. However, these tools are designed to aid the designer in reducing the power consumption of the chip, hence not suitable side channel analysis. This argument holds true for *Nano-Sim* [15] also which provides estimation of power consumption of the CMOS circuits in the design. *PINPAS* [6] can generate power based side channel traces for different crypto-algorithms on different hardware implementation. But it requires the architectural information of the implementation which may not be always available. Another tool *SCARD* [1] provides similar estimation but only a platform-independent one; consequently, its estimation might be of varying accuracy from platform to platform.

The authors in [18] and [5] have proposed a methodology to generate simulated power based side channel traces without the knowledge of the underlying hardware. In [5], authors have used a stochastic model to build the model for power traces generation and used linear regression to generate the simulated power traces. The process comprises of two phases: profiling or training and prediction as shown in Fig. 1. According to this model, the power traces are expressed as below:

$$L(m) = \theta_0(m) + \sum_{i=1}^{i=n} \theta_i(m) x_i \tag{2}$$

where $L(m)$ denotes the value of the power trace at the sample point m, x_is are the parameter which correlates with the power consumption of the device. For example, let us consider the power consumption during the execution of a particular round of a block cipher where a register R_0 is getting updated from value v_0 to v_1. Here x_i can be the Hamming weight of $v_0 \oplus v_1$, in that case n is equal to 1. On the other hand x_i can be the i^{th} bit of $v_0 \oplus v_1$. In that case n is equal to the length of the target register R_0. The value θ_i is the *weight-factor* of x_i and θ_0 is an additional weight-factor popularly known as the *intercept* [14].

As shown in Fig. 1, in the profiling phase, the training side channel traces along with their corresponding plain-text and key value are passed to the model builder. The model builder computes the weight-factors for each point along with the intercepts, which are then used by the predictor to generate the simulated power traces for given plain-text and key.

In this paper, we will mainly consider the above discussed linear regression model [5] for side channel simulation generation. The main motivations behind this choice is that it is relatively easy to design and implement hardware for the linear regression model computation, and for correct TVLA prediction we need to have on-chip power trace generator module. Next we provide more details of the power trace generator module with more focus on the quality of generated power traces.

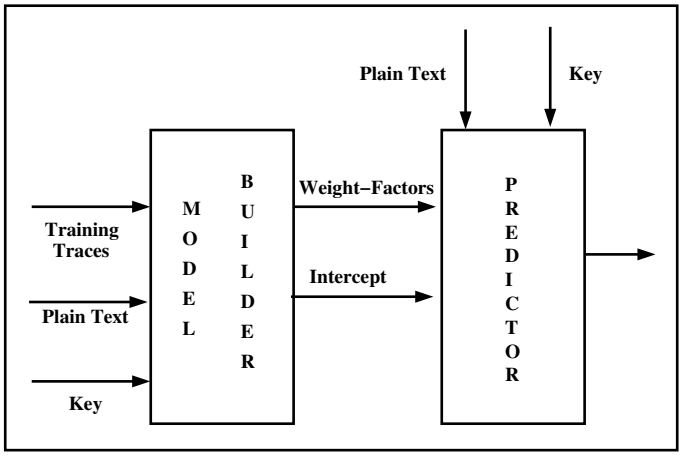

Fig. 1: Simulated Trace Generation Methodology

3 On-Chip Generation of Side Channel traces

As mentioned in Section 2.2, the leakage model of the crypto-implementation is given by following equation:

$$L(m) = \theta_0(m) + \sum_{i=1}^{i=n} \theta_i(m) x_i \tag{3}$$

To build the model we need to calculate the value of $\theta_i(m)$ and $\theta_0(m)$. There are several approaches which can be followed to obtain the values of $\theta_i(m)$ and $\theta_0(m)$. We have used the *Least Mean Square* (LMS) algorithm to compute the values of $\theta_i(m)$ and $\theta_0(m)$. We are not going to provide details of LMS algorithm execution as it is a well known algorithm for supervised learning. Interested readers are encouraged to go through [14] and [9] for further details of LMS algorithm. Instead, we will focus on the quality of the training generation. Accuracy of the model is estimated as the difference of the variances of the actual power traces $C(m)$ and the predicted power traces $P(m)$ as follows:

$$Q(m) = Var(C(m)) - Var(P(m)) \tag{4}$$

The most important factor in this trace generation module is the choice of x_i. We can choose x_i as the Hamming distance of a register, which gets updated during the execution of a particular round of a block cipher. On the other hand, we can also choose x_i as the i^{th} bit of the updated register value. In that case we will have N number of x_i values for N bits of the target register. Both of these were considered and analyzed in [8] [2]. Our experimental results (described in Section 6) indicate that both these choices are equally effective in practice, as the model built by them generate similar quality of side channel

traces. Additionally, we also consider the trace count as another parameter for model building and generate the power traces. Finally, we develop another model using ANN (*Artificial Neural Network*) and compare it with the developed linear models (LMs).

Fig. 2 shows the comparison between different models and actual power traces. This figure clearly exhibits the practical equivalence between different models and allows us to choose a simpler lightweight model comprising only *Hamming distance* (HD) as parameter for the on-chip trace generator. The equation used in the on-chip power trace generator is given below:

$$P(t) = \theta_0 + \theta_1 \times HD \tag{5}$$

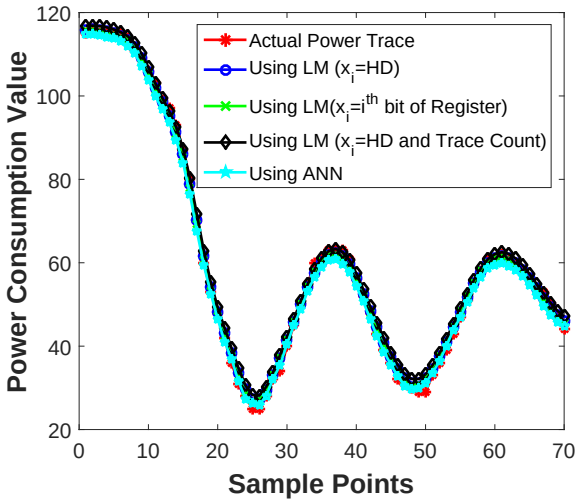

Fig. 2: Comparison Between Different Model for Power Trace Generation

Additionally, this developed model only exhibits the leakage information of the side channel traces. Actual side channel signals contain this leakage information but are contaminated with noise. This noise signal can be broadly classified into two different classes viz. algorithmic noise and device specific noise. Different crypto-algorithms will have different algorithmic noise on the same device. On the other hand, each device will have different noise level for same algorithm due to process variation. Hence, to make the side channel trace generation more accurate, we need to incorporate this noise into the simulation of side channel traces. Keeping the crypto-algorithm fixed, we have computed the device specific noise and incorporated it in the developed model to generate accurate side channel traces. The histogram of the device specific noise is shown in Fig. 3. This figure clearly shows that this device specific noise is can be well-approximated by a Gaussian model.

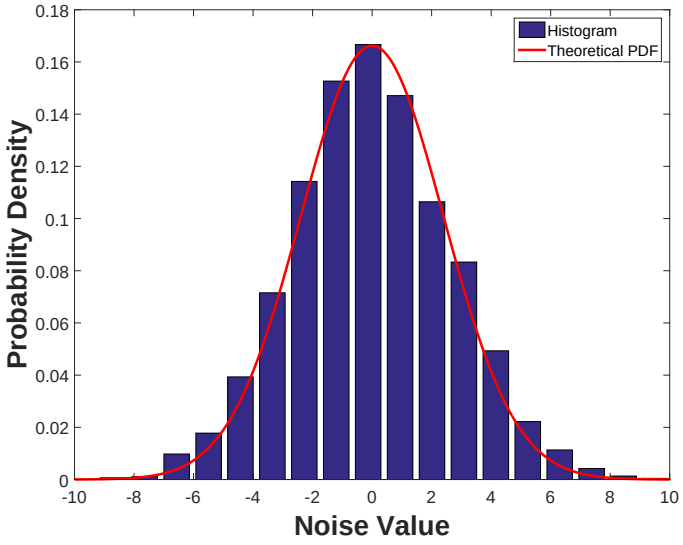

Fig. 3: Theoretical PDF and Simulated Histogram of Gaussian Noise

4 TVLA Computation on Hardware

As described in Section 3, the computation of TVLA demands a significant number of side channel traces acquired from a given crypto-implementation hardware. This approach can be suitably applied for the analysis of crypto-algorithm in an online mode. The same scheme can also be adapted for online analysis where we will be requiring to make decisions based on real time acquisition of side channel information.

In Eq.(1), we impose an assumption that the second data set will have almost constant mean and negligible variance as it consists of traces from a fixed input. Let us assume $\mu_b \approx C$ and $\sigma_b^2 \approx 0$. We can rewrite Eq.(1) as

$$t \approx \frac{\mu_a - C}{\sqrt{\sigma_a^2/N_a}} \tag{6}$$

Let x_n be the n^{th} side channel trace value at any sample point m. Also we consider $\mu_{a_{n,m}}$, $CM2_{a_{n,m}}$ and $\sigma^2_{a_{n,m}}$ be the mean, second central moment and variance respectively after n traces.
The t-statistic value is computed using the following equations:

$$t_{n,m} \approx \frac{\mu_{a_{n,m}} - C}{\sqrt{\sigma^2_{a_{n,m}}/n}} \qquad n = 1, 2, 3....., N_a \tag{7}$$

where the μ and CM values can be calculated iteratively as follows:

$$\mu_{a_{n,m}} = \mu_{a_{n-1,m}} + \frac{x_{n,m} - \mu_{a_{n-1,m}}}{n} \tag{8}$$

$$CM_{2_{a_{n,m}}} = CM_{2_{a_{n-1,m}}} + \frac{(x_{n,m} - \mu_{a_{n-1,m}})^2}{n} \times (n-1) \tag{9}$$

$$\sigma^2_{a_{n,m}} = \frac{CM_{2_{a_{n,m}}}}{(n-1)} \tag{10}$$

From the above equations we note that for computing TVLA value we require three divisions and one square root operations per new side channel trace per data point, which increases area as well as performance overheads. In order to implement the TVLA module more optimally in hardware, we have incorporated certain modifications whose details are mentioned as follows

1. **Minimizing hardware critical operations**: For efficient implementation of TVLA module in hardware we need to minimize the hardware-intensive operations such as divisions and square roots. In a way we need to compute the TVLA value such that we can systematically exploit the hardware resources. To achieve this objective, the Eq.(7) is modified accordingly such that we calculate the TVLA values only with additions, subtractions and multiplications. We introduce a variable $\Delta_{n,m}$ which is defined as:

$$\Delta_{n,m} = \frac{(x_{n,m} - \mu_{a_{n-1,m}})}{n} \tag{11}$$

Eq.(11) for computation of $\Delta_{n,m}$ can be viewed as a multiplication of $(x_{n,m} - \mu_{a_{n-1,m}})$ with the inverse of n for the n^{th} side channel trace value. In our module the inverse is computed offline for some fixed number of traces and stored in on-chip memory (e.g. Block RAM in FPGA), such that the value can be queried on demand during the computation of TVLA.
Eqns.(8) and (9) can be rewritten in terms of $\Delta_{n,m}$ as

$$\mu_{a_{n,m}} = \mu_{a_{n-1,m}} + \Delta_{n,m} \tag{12}$$

and

$$CM_{2_{a_{n,m}}} = CM_{2_{a_{n-1,m}}} + \Delta_{n,m} \times (x_{n,m} - \mu_{a_{n-1,m}}) \times (n-1) \tag{13}$$

In a similar manner the variance $\sigma^2_{a_{n,m}}$ can be computed from Eq.(10) by multiplying $CM_{2_{a_{n,m}}}$ with the inverse of $(n-1)$ which is accessible from the Block RAM.
Now, squaring Eq.(7) we get

$$t^2_{n,m} \approx \frac{(\mu_{a_{n,m}} - C)^2 \times n}{\sigma^2_{a_{n,m}}} \qquad n = 1, 2, 3....., N_a \tag{14}$$

Rearranging Eq.(14) we obtain as follows

$$t^2_{n,m} \times \sigma^2_{a_{n,m}} \approx (\mu_{a_{n,m}} - C)^2 \times n \tag{15}$$

Thus instead of calculating the t-value directly we make decisions based on comparison by exploiting the Eq. (15). As mentioned in Section 4 the

safe value, $t_{SAFE} = |4.5|$. By replacing $t^2_{n,m}$ in Eq. (15) with the value of $t^2_{SAFE} = 20.25$, we say the crypto-algorithm is unsafe or not at any m^{th} sample point for the n^{th} trace value if it satisfies the below mentioned condition

$$UNSAFE_{m,n} = \begin{cases} 1, & if \quad 20.25 \times \sigma^2_{a_{n,m}} < (\mu_{a_{n,m}} - C)^2 \times n \\ 0, & if \quad 20.25 \times \sigma^2_{a_{n,m}} \geq (\mu_{a_{n,m}} - C)^2 \times n \end{cases}$$

Consequently with this approach we are able to make real time decisions based on TVLA computation with efficient utilization of hardware resources.

2. **Avoiding Floating Point Operations**: The predicted side channel traces so obtained as described in Section 2.2 are represented in floating point numbers. The arithmetic operations on floating point numbers not only incur area overheads but also make the hardware design more complex. In our method instead of performing floating point operations, we adapt a mechanism to compute the TVLA value using fixed point operations by representing the computed power traces in standard Q-number format and keeping the resolution of fractional parts up to 8-bits. The values of θ_0 and θ_1 computed from Eq.(5) during profiling phase are converted to its Q-number format and given as constants to the on-chip TVLA trace simulator. Also by ignoring the excess bits, the fractional parts are restricted to 8-bits during intermediate operations in TVLA computation. It is observed that the total computation of TVLA can be done using a register of length atmost 50-bits. The fixed point operations are implemented using DSP hard macros available on FPGAs [20].

3. **Minimal number of Sample Points**: As seen from Eq.(7), the computation of TVLA is done on side channel traces for a particular m^{th} sample point. The maximum value of m depends on the number of sample points chosen in order to avoid false alarms. But as we increase the number of sample points (m) and henceforth compute the TVLA for each point, the hardware efficiency is compromised to a great extent. Instead, for a particular crypto-algorithm we primarily focus on those points in side channel information traces that are highly prone to CPA. Henceforth, the TVLA values on entire set of points are calculated on side channel traces that are used during the profiling phase in offline mode. We then identify those sample points that corresponds to high TVLA values. The parameters θ_0 and θ_1 computed in Eq.(5) are then incorporated in each of the TVLA modules. Thereafter the module for each sample points compute the TVLA values real-time in hardware and raise alarms to thwart any attack. In our simulations, we have taken $m = 6$ as a satisfactory trade-off between modeling accuracy and hardware overhead.

5 Architecture and Resource Requirement

In our experimental setup, we have used SASEBO-GII as target evaluation board with AES-128 as crypto-implementation [7]. The TVLA values are computed over a set of 9000 traces. The overall hardware architecture is depicted as a block diagram in Fig. 4.

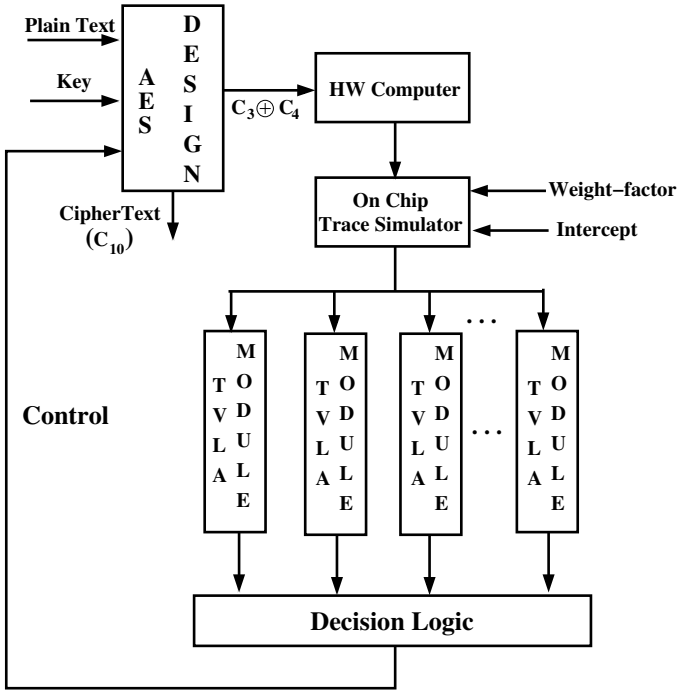

Fig. 4: Overall Architectural Framework

The summary of the resource utilization is highlighted in **Table 1**.

Implementation	LUT's	Slices	Slice Registers	Block RAM	DSP Slices	Critical Path Time(ns)
TVLA with floating point	2490	838	418	0	0	24.026
TVLA with fixed point	199	110	222	2	24	13.929

Table 1: Resource Utilization in SASEBO-GII

From **Table 1** it is quiet evident that the fixed point implementation of TVLA is much faster and hardware efficient as compared to its floating point representation. The latency for the standard implementation of AES-128 module

on SASEBO-GII evaluation board is observed to be approximately 14 ns [17]. Additionally, the fixed point implementation of TVLA module gives an advantage by eliminating the synchronization issues with the standard AES-128 crypto-implementation on SASEBO-GII hardware. Henceforth, we adapt the fixed point approach for the on-chip TVLA module.

6 Experimental Validation

As mentioned in Section 5, the first 4500 traces out of 9000 traces are used for training or profiling in order to obtain the parameters θ_0 and θ_1, which are then applied for prediction of the remaining 4500 traces. The predicted trace is then further added with modeled hardware specific noise on-chip as described in Section 3. We then compute the values for Eq.(15) with $t^2 = t^2_{SAFE} = 20.25$ over a particular sample point on actual power traces acquired using oscilloscopes. The same computation is carried out for predicted traces. Fig. 5 depicts the graphical representation of TVLA computation over actual and predicted power traces for sample points with $m = 6$ using fixed point implementation. From Fig. 5 it is quiet evident that TVLA value for predicted traces shows the same trend when compared with the TVLA values with actual power traces. Further, we carried out a comparative analysis for TVLA computation in floating and fixed point representation by calculating the corresponding errors. It is observed that the error between floating and fixed point estimations of TVLA lie within ± 20 which is quiet acceptable at per with the required hardware constraints.

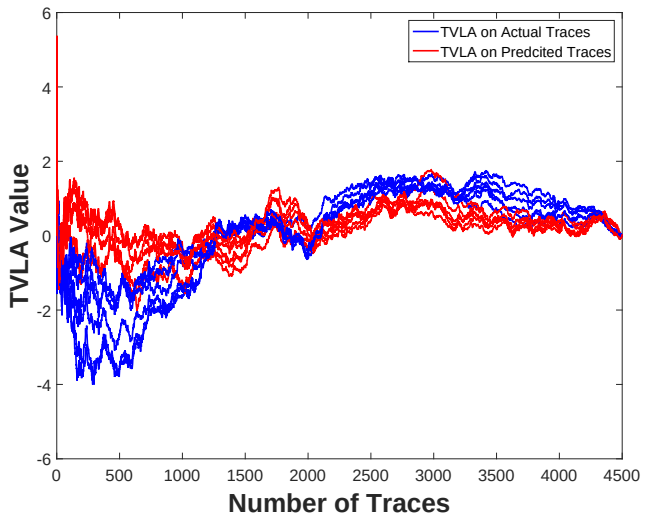

Fig. 5: TVLA Computation of Traces using Fixed Point Implementation for $m = 6$

7 Thwarting SCA through TVLA Alert

Once the integrated TVLA module exhibits TVLA value beyond the safe value, certain preventive steps must be adopted to protect the underlying crypto-algorithm from side channel adversary. In this section, we will briefly outline few examples of such preventive techniques.

Generally, the objective of the side channel adversary is to obtain the key of the cryptographic implementation. Hence to prevent that certain protection mechanism can be employed. We categorized these techniques into two broad categories: 1. Device Level Prevention Mechanisms and 2. Protocol Level Prevention Mechanisms. In case of device level prevention mechanism, we try to reconfigure the some parameter of the crypto-algorithm in such a way so that the side channel adversary does not get any information regarding the key value. Some of such device level protection mechanisms are listed below:

1. **Changing the key:** In [3], the authors have presented *fresh re-keying scheme*, where each encryption is computed with a different key, generated from a master key. This scheme is synchronized between sender and receiver and requires only the sharing of the master key. We can adopt similar methodology once the TVLA alert goes high. However in this case, instead of using different key for each encryption, we will change the key only when the TVLA alert goes high
2. **Changing the S-Box:** Deploying a pool of different S-Boxes and doing encryption by randomly choosing one S-box from this pool of S-boxes to prevent CPA was proposed in [12]. In this case, apart from key, there is another secret value termed *tweak* which determines the choice of S-box and is changed for every encryption. Like the previous example, we can also adopt this mechanism in our case by changing the tweak value only when the TVLA value goes beyond the safe value. Efficient generation architecture of such S-boxes is discussed in [16].
3. **Changing the Mask Value (BMS Scheme):** Masking is probably the most popular countermeasure against power attack with a theoretical security guarantee. However, this security guarantee holds only when the mask value is changed from each encryption. An alternative Block RAM scrambling mechanism for efficient masking was proposed in [11] where we do multiple encryptions with a single mask and changes the mask only after a pre-specified number of encryptions is executed. An alternative LUT based scrambling mechanism is discussed in [16]. Now in this case, instead of changing mask after a pre-specified number of encryption, we can change the mask only when the TVLA alert goes high, making the countermeasure more efficient and smart.
4. **Protocol Level Prevention Strategy:** With advancement of IoT based framework in implementing complex networks, security against potential eavesdropper becomes a primal condition. A model based upon quantitative information flow against side-channel analysis of web traffic has been proposed in [13]. However if an IoT node becomes vulnerable to SCA attacks by

an adversary, the full information flow gets compromised. The on-chip TVLA can be implemented on these resource constrained devices, which will trigger some probable decision strategies at protocol level upon TVLA alert. Once such possibility can be bypassing those nodes whose TVLA value crosses the desired threshold level and hence vulnerable to adversary. Once the node regains the safe TVLA value, the information flow is restored. The overall strategy is pictorially represented in Fig. 6.

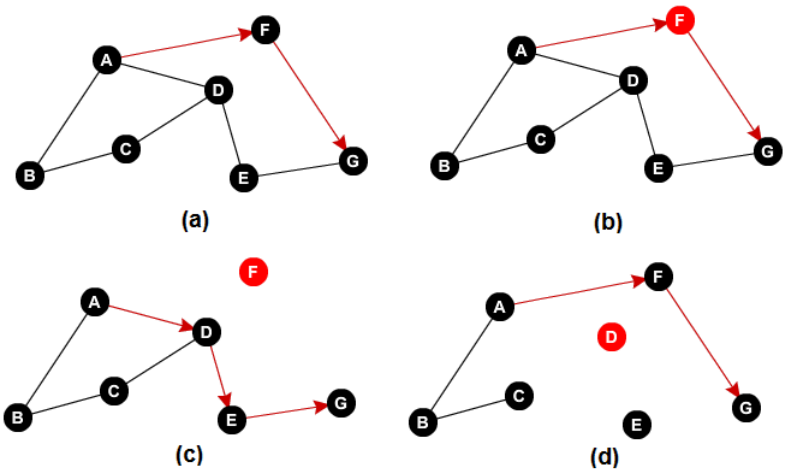

Fig. 6: A plausible Protocol Level Prevention Scheme for IoT based architecture. Each step from (a) to (d) represents the flow of information from IoT node A to node G.
(a) Information flow is initiated from A to G via F.
(b) Node F with high TVLA value is prone to adversary attack.
(c) Node F is isolated from network with information flow bypassed via D, E.
(d) Infomation flow is resumed from A to G via F once node F has safe TVLA value

8 Conclusion

We have described a technique for on-chip estimation of SCA vulnerability of cryptographic hardware. Our scheme utilizes the widely used TVLA metric to perform this estimation. We have demonstrated through experimental results the effectiveness of a low-overhead fixed-point implementation of the TVLA estimation hardware. The scheme can be used to provide proactive, real-time protection against SCA, both at device and protocol level. Our future research would be directed towards demonstrating the effectiveness of the proposed methodology in an IoT framework.

References

1. Martin Aigner, Stefan Mangard, Francesco Menichelli, Renato Menicocci, Mauro Olivieri, Thomas Popp, Giuseppe Scotti, and Alessandro Trifiletti. Side channel analysis resistant design flow. In *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on*, pages 4–pp. IEEE, 2006.
2. G Becker, J Cooper, E DeMulder, G Goodwill, J Jaffe, G Kenworthy, T Kouzminov, A Leiserson, M Marson, P Rohatgi, et al. Test Vector Leakage Assessment (TVLA) methodology in practice.
3. Sonia Belaïd, Fabrizio De Santis, Johann Heyszl, Stefan Mangard, Marcel Medwed, Jörn-Marc Schmidt, François-Xavier Standaert, and Stefan Tillich. Towards fresh re-keying with leakage-resilient PRFs: cipher design principles and analysis. *Journal of Cryptographic Engineering*, 4(3):157–171, 2014.
4. Cadence. Cadence NC-Verilog Simulator Help. `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.433.1841`.
5. Nicolas Debande, Maël Berthier, Yves Bocktaels, and Thanh-Ha Le. Profiled Model Based Power Simulator for Side Channel Evaluation. *IACR Cryptology ePrint Archive*, 2012:703, 2012.
6. Jerry den Hartog, Jan Verschuren, E de Vink, Jaap de Vos, and W Wiersma. PINPAS: a tool for power analysis of smartcards. In *Security and privacy in the age of uncertainty*, pages 453–457. Springer, 2003.
7. Research Center for Information Security (National Institute of Advanced Industrial Science and Technology). Side-Channel Attack Standard Evaluation Board (SASEBO): SASEBO-GII. `http://www.rcis.aist.go.jp/special/SASEBO/SASEBOGII-en.html`.
8. Benjamin Jun Gilbert Goodwill, Josh Jaffe, Pankaj Rohatgi, et al. A testing methodology for side-channel resistance validation. In *NIST Non-invasive attack testing workshop*, 2011.
9. Eleftherios Giovanis. Applications of Least Mean Square (LMS) Algorithm Regression in Time-Series Analysis. *Available at SSRN 1667440*, 2010.
10. Mentor Graphics. ModelSim. `http://www.mentor.com/products/fpga/model`.
11. Tim Güneysu and Amir Moradi. Generic side-channel countermeasures for reconfigurable devices. In *Cryptographic Hardware and Embedded Systems–CHES 2011*, pages 33–48. Springer, 2011.
12. Suvadeep Hajra, Chester Rebeiro, Shivam Bhasin, Gaurav Bajaj, Sahil Sharma, Sylvain Guilley, and Debdeep Mukhopadhyay. DRECON: DPA Resistant Encryption by Construction. In *Progress in Cryptology–AFRICACRYPT 2014*, pages 420–439. Springer, 2014.
13. Michael Backes Goran Doychev Boris Kopf. Preventing side-channel leaks in web traffic: A formal approach. 2013.
14. Sears Merritt and Abigail Jacobs. Lecture Notes for CSCI 5454. 2012.
15. Giuseppe Piro, Luigi Alfredo Grieco, Gennaro Boggia, and Pietro Camarda. Nano-Sim: simulating electromagnetic-based nanonetworks in the network simulator 3. In *Proceedings of the 6th International ICST Conference on Simulation Tools and Techniques*, pages 203–210. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2013.

16. Debapriya Basu Roy, Shivam Bhasin, Sylvain Guilley, Jean-Luc Danger, Debdeep Mukhopadhyay, Xuan Thuy Ngo, and Zakaria Najm. *Security, Privacy, and Applied Cryptography Engineering: 5th International Conference, SPACE 2015, Jaipur, India, October 3-7, 2015, Proceedings*, chapter Reconfigurable LUT: A Double Edged Sword for Security-Critical Applications, pages 248–268. Springer International Publishing, Cham, 2015.
17. Akashi Satoh, Toshihiro Katashita, and Hirofumi Sakane. Secure implementation of cryptographic modules.
18. Werner Schindler, Kerstin Lemke, and Christof Paar. A stochastic model for differential side channel cryptanalysis. In *Cryptographic Hardware and Embedded Systems–CHES 2005*, pages 30–46. Springer, 2005.
19. Tobias Schneider and Amir Moradi. Leakage Assessment Methodology. In Tim Gneysu and Helena Handschuh, editors, *Cryptographic Hardware and Embedded Systems – CHES 2015*, volume 9293 of *Lecture Notes in Computer Science*, pages 495–513. Springer Berlin Heidelberg, 2015.
20. L.F. Stadler. Hard macro-to-user logic interface, August 11 2009. US Patent 7,573,295.