Chapter 6
Testing of Side-Channel Leakage of Cryptographic Intellectual Properties: Metrics and Evaluations

Debapriya Basu Roy, Shivam Bhasin, Sikhar Patranabis, and Debdeep Mukhopadhyay

6.1 Introduction

Semiconductor industry has really flourished under the intellectual property (IP) based business model, where proven IPs are widely reused to meet complexity and time constraints. Since security is emerging as the latest design parameter, along with performance, area, and power consumption, these IPs must be hardened for security. However, it is not always possible to modify each proven IP and add security features. To overcome this issue, required security targets are met at the system level by embedding cryptographic IPs in the system on chip (SoC). These cryptographic IPs ideally encrypt and decrypt all the sensitive data that is communicated among various IPs on the SoC and thus protect it from potential adversaries. The underlying cryptographic algorithms are rigorously tested against any theoretical weaknesses and often standardized.

D.B. Roy (✉) • S. Patranabis
Secured Embedded Architecture Laboratory (SEAL), Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, Kharagpur, India
e-mail: deb.basu.roy@cse.iitkgp.ernet.in; dbroy24@gmail.com; sikhar.patranabis@cse.iitkgp.ernet.in

S. Bhasin
Temasek Laboratories NTU, Singapore, Singapore

Embedding Security and Privacy (ESP) IIT Kharagpur, Kharagpur, India
e-mail: sbhasin@ntu.edu.sg

D. Mukhopadhyay
Secured Embedded Architecture Laboratory (SEAL) Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, Kharagpur, India

Embedding Security and Privacy (ESP) IIT Kharagpur, Kharagpur, India
e-mail: debdeep@cse.iitkgp.ernet.in

P. Mishra et al. (eds.), *Hardware IP Security and Trust*,
DOI 10.1007/978-3-319-49025-0_6

Since the used algorithms are provably secure, adversaries try to exploit other vulnerabilities. A commonly known vulnerability is side-channel attack (SCA [1]). SCA exploits unintentional physical leakage from semiconductor circuits in the form of power consumption, electromagnetic emanation, sound, timing, temperature, etc. The most commonly used physical channel is power consumption. It derives its basic from the behavior of a basic CMOS cell. It is well known that a CMOS gate draws non-negligible current whenever there is a transition ($1 \rightarrow 0$ or $0 \rightarrow 1$) on the input. When the input remains unchanged ($0 \rightarrow 0$ or $1 \rightarrow 1$), the current drawn is negligible or zero. Therefore just by observing the power consumption of a single CMOS gate, an adversary can retrieve some information about the inputs. Since the security of whole SoC relies on the embedded cryptographic IP, it should be tested and protected against SCA. SCA are now also tested by certification labs following Common Criteria [2] or FIPS [3] standards.

The simplest and the most common way to test for side-channel leakage is to perform an attack. The attack is based on intuition or characterization of the leakage model of the target device or IP. The actual power consumption is statistically compared against a set of key hypothesis, under the chosen leakage model. The efficiency of the attack can then be measured in terms of standard metrics like success rate and guessing entropy [4]. However, the attack directly relies on correctness of the leakage model and the quality of acquired side-channel measurements. To overcome these dependencies, alternative techniques like leakage detection and leakage assessment are used. Statistical test can also affect the efficiency of the attack but it was previously shown that all applied statistical tests are asymptotically equivalent [5].

Leakage detection is a preprocessing step in SCA testing. It serves two basic purpose. The first and the most obvious is quality assessment of the acquired side-channel measurement. In other words, leakage detection test checks if the collected measurement carries any relevant information as a quality check. This is even more important when the measurements are not synchronized and the adversary is not sure to capture good section of the measurement. The other and an important application of leakage detection is discovering zone of leakage or relevant point-of-interest (PoI). A side-channel trace or measurement can easily have millions of samples and the number of measurements can also grow in millions for protected implementations. To process such amount of data, one needs immense computing power. Therefore it is important to select the relevant PoI to improve the attack's efficiency.

Leakage assessment, on the other hand, is a generic evaluation technique. Normal SCA testing is highly dependent on the attack parameters (leakage model, statistical tool, and measurements). For evaluation and certification purpose, this scenario is not ideal as the number of parameters keep on increasing, which hinder the evaluation process to be comprehensive. To overcome this problem, evaluators rely on leakage assessment. These leakage assessment techniques are primarily global and parameter independent. It tests for any potential side-channel leakage in the

acquired measurement, without actually performing the attack. If a minimum level of leakage is found, the target is rendered insecure.

This chapter focusses on various aspect of side-channel testing on cryptographic IPs. It aims at establishing basic concepts of side-channel attack, leakage detection, and leakage assessment. Side-channel attack is the simplest and straightforward technique for testing cryptographic IC. In this context, we provide a formal description of an SCA along with its evaluation metrics, i.e., success rate and guessing entropy. Thereafter we describe the notion of leakage detection with a specific leakage detection tool known as normalized inter-class variance (NICV [6]). Relationship of NICV with standard parameters like signal-to-noise ratio (SNR), correlation, and other detection technique is also discussed. Practical case studies on hardware and software platforms are included to help readers understand the basic concepts. Lastly, leakage assessment is discussed, in particular test vector leakage assessment (TVLA [7]). Along with the basic concepts of TVLA, we derive relationship between TVLA and NICV. TVLA is also supported with a practical case study with direct application to SCA protected implementation.

The rest of the chapter is organized as follows. Section 6.2 discusses general background on statistical testing and hypothesis testing. This is followed with a formal description of side-channel attack with emphasis on evaluation metric, i.e., success rate and guessing entropy in Sect. 6.3. Leakage detection methods like NICV and SNR are dealt in Sect. 6.4 along with practical case studies on AES implementation. Section 6.5 describes leakage assessment methodology, in particular TVLA. The relationship between NICV and TVLA is derived in Sect. 6.6 followed by its extension to higher statistical orders in Sect. 6.7. and practical case study in Sect. 6.8. Finally conclusions are derived in Sect. 6.9.

6.2 Preliminaries on Statistical Testing and Testing of Hypothesis

Attack based evaluation strategies are difficult in the practical world as the number of attacks have been steadily increasing. Hence, it is desirable to have a black-box approach, the objective of which will be to quantify *any* side-channel information leakage through the power consumption of the device. The objective of the test strategy is thus to detect whether there is any information leakage, and not to precisely quantify how much of the information leakage is exploitable. Thus the purpose is to develop an *estimation* technique, which will rely on the *Theory of Statistical Hypothesis* and *Theory of Estimation*. The objective of this subsection is to provide a quick overview on the topic.

6.2.1 Sampling and Estimation

Sampling denotes the selection of a part of the aggregate statistical material with a view to obtaining information of the whole. This totality of statistical information on a particular character of all the members relevant to an investigation is called population. The selected part which is used to ascertain the characteristics of population is called *Sample*. The main objective of a good sampling is to obtain maximum information about the population with minimum effort, and to state the limits of accuracy of estimates based on samples.

Any statistical measure calculated on the basis of sample observations is called a *Statistic*, e.g., sample mean, sample variance. On the other hand, statistical measures based on the entire population are called a *Parameter*, e.g., population mean, population variance. The sample depends on chance, and hence the value of a statistic will vary from sample to sample, while the parameter remains a constant. The probability distribution of a statistic is called *sampling distribution*, and the standard deviation of the sampling distribution is called *standard error*, denoted as SE.

In the sequel, we assume simple random sampling, which implies that in the process of selecting a sample, every unit of the population has an equal probability of being included. Furthermore, we consider that the sampling is such that the probability of selection of any particular member remains constant throughout the selection process, irrespective of the fact that the member has been selected earlier or not. Such a sampling can be obtained by performing a simple random sampling with replacement (SRSWR), and is commonly called as simple sampling. It may be mentioned here that when the population size is infinite, even performing a simple random sampling without replacement (SRSWOR) will also result in simple sampling. Thus in cases where the population size is extremely large (say the plaintext pool in case of a 128 bit block cipher) both SRSWR and SRSWOR will result in simple sampling. Thus, we can assume that each sample members has the same probability distribution as the variable x in the population. Hence, the expectation $\mathbb{E}[x_i] = \mu$, where μ is the population mean. Likewise, the variance $\mathsf{Var}[x_i] = \mathbb{E}\left[(x_i - \mu)^2\right] = \sigma^2$, which is the population variance.

It can be proved that standard error for the mean, $\mathrm{SE}(\overline{(x)}) = \frac{\sigma}{\sqrt{n}}$, where n is the size of the sample. We state formally a subsequent result on standard errors which will be useful to understand the subsequent discussion on the detection test.

Theorem 1. *Consider two independent simple samples of sizes n_1 and n_2, with means $\overline{x_1}$ and $\overline{x_2}$, and standard deviations σ_1 and σ_2, respectively, then:*

$$\mathrm{SE}(\overline{x_1} - \overline{x_2}) = \sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}} \tag{6.1}$$

Different probability distributions are used in sampling theory, and they are all derived from the Normal distribution. They are (1) Standard Normal Distribution,

(2) Chi-square (ψ^2) Distribution, (3) Student's t Distribution, and (4) Snedecor's F Distribution.

In the following we provide a quick refresher to the Standard Normal Distribution, and the Student's t Distribution which shall be useful for understanding the remaining part of the sequel and in introducing the T-test.

6.2.2 Some Statistical Distributions

If a random variable x is normally distributed with mean μ and standard deviation σ, then the variable $z = \frac{x-\mu}{\sigma}$ is called a standard normal variate. The probability distribution of z is called Standard Normal Distribution, and is defined by the probability density function (pdf), $p(z) = (1/\sqrt{2\pi})e^{-z^2/2}$, where $-\infty < z < +\infty$.

An important result is if $\overline{x}$ denotes the mean of a random sample of size n, drawn from a normal population with mean μ and standard deviation (s.d.) σ, then, $z = \frac{\overline{x}-\mu}{\sigma/\sqrt{n}}$ follows standard normal distribution.

Likewise, a random variable is said to follow Student's t-distribution, or simply t-distribution, if its pdf is of the form: $f(t) = K(1 + \frac{t^2}{n})^{-(n+1)/1}$, where K is a constant and $-\infty < t < +\infty$. The parameter n is called the number of degrees of freedom (df.).

In statistics, the number of degrees of freedom is the number of values in the final calculation of a statistic that are free to vary. Estimates of statistical parameters can be based upon different amounts of information or data. The number of independent pieces of information that go into the estimate of a parameter is called the degrees of freedom. In general, the degrees of freedom of an estimate of a parameter are equal to the number of independent scores that go into the estimate minus the number of parameters used as intermediate steps in the estimation of the parameter itself (i.e., the sample variance has $N - 1$ degrees of freedom, since it is computed from N random scores minus the only one parameter estimated as intermediate step, which is the sample mean).

6.2.3 Estimation and Test of Significance

The objective of sampling is to infer the features of the population on the basis of sample observations. Statistical inference has two different ways: (1) Point Estimation and (2) Interval Estimation. In point estimation, the estimated value is given by a single quantity, which is a function of the given observations. In interval estimation, an interval within which the parameter is expected to lie is given by using two quantities based on sample values. This is known as Confidence Interval, and the two quantities which are used to specify the interval are known as Confidence Limits.

Let $x_1, x_2, \ldots, x_n$ be a random sample from a population of a known mathematical form which involves an unknown parameter θ. The confidence intervals specify two functions t_1 and t_2 based on sample observations such that the probability of θ being included in the interval (t_1, t_2) has a given value, say c: i.e., $P(t_1 \leq \theta \leq t_2) = c$. The probability c with which the confidence interval will include the true value of the parameter is known as Confidence Coefficient of the interval.

Let us illustrate this using an example. Let us consider a random sample of size n from a Normal population $N(\mu, \sigma^2)$, where the variance σ^2 is known. It is required to find confidence intervals for the mean, μ. We know that the sample mean $\overline{x}$ follows approximately a normal distribution with mean μ and variance σ^2/n. Thus, the statistic $z = (\overline{x} - \mu)/(\sigma/\sqrt{n})$ has a standard normal distribution. From the properties of the standard normal curve, 95 % of the area under the standard normal curve lies between the ordinates at $z = \pm 1.96$. Thus, we have

$$P[1.96 \leq (\overline{x} - \mu)/(\sigma/\sqrt{n}) \leq 1.96] = 0.95 \tag{6.2}$$

Thus arranging terms, the interval $(\overline{x} - 1.96\frac{\sigma}{\sqrt{n}}, \overline{x} + 1.96\frac{\sigma}{\sqrt{n}})$ is known as the 95 % confidence interval for μ. For almost sure limits, we replace the value 1.96 by the value 3.

In some cases, the population may not be truly a normal distribution, but the sample distributions of statistics based on large samples are approximately normal.

6.2.4 Test of Significance: Statistical Hypothesis Testing

Statistical tests often require to make decisions about a statistical population on the basis of sample observations. For example, given a random sample, it may be required to decide whether the population from which the sample has been obtained is a normal distribution with a specific mean and standard deviation. Any statement or assertion about a statistical population or its parameters is called a Statistical Hypothesis. The procedure which enables us to decide whether a certain hypothesis is true or not is called Test of Significance or Statistical Hypothesis Testing.

A statistical hypothesis which is set up (i.e., assumed) and whose validity is tested for possible rejection on the basis of sample observations is called Null Hypothesis. It is denoted as H_0 and tested for acceptance or rejection. On the other hand, an Alternative Hypothesis is a statistical hypothesis which differs from the null hypothesis, and is denoted as H_1. This hypothesis is not tested, its acceptance (or rejection) depends on the rejection (or acceptance) of that of the null hypothesis. For example, the null hypothesis may be that the population mean is 40, denoted as $H_0(\mu = 40)$. The alternative hypothesis could be $H_1(\mu \neq 40)$.

The sample is then analyzed to decide whether to reject or accept the null hypothesis. For this purpose, a suitable statistic, called Test Statistic is chosen. Its sampling distribution is determined, assuming that the null hypothesis is true.

The observed value of the statistic would be in general different from the expected value because of sampling fluctuations. However if the difference is very large, then the null hypothesis is rejected, Whereas, if the differences is less than a tolerable limit, then H_0 is not rejected. Thus it is necessary to formally determine these limits.

Assuming the null hypothesis to be true, the probability of obtaining a difference equal to or greater than the observed difference is computed. If this probability is found to be small, say less than 0.05, the conclusion is that the observed value of the statistic is rather unusual, and has arisen because the underlying assumption, i.e., the null hypothesis is not true. We say that the observed difference is significant at 5 % level of significance, and hence the null hypothesis is rejected at 5 % level of significance. The level of significance, say α also corresponds to a $(1 - \alpha)$ level of confidence. If, however, this probability is not very small, say more than 0.05, the observed difference cannot be considered unusual and is attributed to sampling fluctuations only. The difference now is not significant at 5 % level of significance.

The probability is inferred from the sampling distribution of the statistic. We find from the sampling distribution of the statistic the maximum difference which is exceeded say 5 % of cases. If the observed difference is larger than this value, the null hypothesis is rejected. If it is less, there is no reason to reject the null hypothesis.

Let us illustrate this with an example. Suppose, the sampling distribution of the statistic is a normal distribution. Since, the area under the normal curve under the ordinates at mean $\pm$ 1.96 (standard deviation) is only 5 %, the probability that the observed value of the statistic differs from the expected value by 1.96 times or more the standard error of the statistic (which is the standard deviation of the sampling distribution of the statistic) is 0.05. The probability of a larger difference will be still smaller.

If, therefore the statistic $z = \frac{\text{(Observed Value)}-\text{(Expected Value)}}{\text{Standard Error (SE)}}$ is either greater than 1.96 or less than -1.96, the null hypothesis H_0 is rejected at 5 % level of significance. The set of values $z \geq 1.96$, or $z \leq -1.96$ constitutes what is called the Critical Region of the test.

Thus the steps in Test of Significance can be summarized as follows:

1. Set up the Null Hypothesis H_0 and the Alternative Hypothesis, H_1. The null hypothesis usually specifies some parameter of the population: $H_0(\theta = \theta_0)$.
2. State the appropriate test statistic T and also its sampling distribution, when the null hypothesis is true. In large sample tests, the statistic $z = (T - \theta_0)/\text{SE}(T)$, which approximately follows standard Normal distribution, is often used. In small sample tests, the population is assumed to be normal and various test statistics are used which follow Standard Normal, Chi-Square, t distribution.
3. Select the level of significance, α of the test, or equivalently $(1 - \alpha)$ as the level of confidence.
4. Determine the critical region of the test for the chosen level of significance.
5. Compute the value of the test statistic z on the basis of sample data and the null hypothesis.
6. Check whether the computed value of the test statistic lies in the critical region. If it lies, then reject H_0, else H_0 is not rejected.

With this background, we eventually arrive at the proposed Test Vector Leakage assessment test, which is essentially a test of equality of two moments drawn independently and randomly from two populations. The starting point is the first moment, where equality of two means from the two samples are tested for equality. Thus, the statistic $T = \overline{x_1} - \overline{x_2}$, and thus the statistic $z = \frac{\overline{x_1} - \overline{x_2}}{\mathrm{SE}(\overline{x_1} - \overline{x_2})}$ is chosen for testing the null hypothesis: $H_0(\mu_1 = \mu_2)$, where μ_1 and μ_2 are the two means for the two independent samples. As discussed, the standard error of the difference of means $\overline{x_1} - \overline{x_2}$ is $\mathrm{SE}(\overline{x_1} - \overline{x_2}) = \sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}$.

For a large distribution, the test statistic $z = \frac{\overline{x_1} - \overline{x_2}}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}}$ follows standard normal distribution. However, for tests with any sample sizes, a more exact sampling distribution for z is the t-distribution, and this gives rise to the Welch's t-test. The statistic z then follows the t-distribution with degrees of freedom calculated according to Welch–Satterthwaite, as $v = \frac{\mathrm{SE}(\overline{x_1} - \overline{x_2})}{\frac{(\sigma_1^2/n_1)}{n_1 - 1} + \frac{(\sigma_2^2/n_2)}{n_2 - 1}}$. The null hypothesis of two equal means is rejected when the test statistic $|z|$ exceeds a threshold of 4.5, which ensures with degrees of freedom > 1000, $P[|z| > 4.5] < 0.00001$, this threshold leads to a confidence of 0.99999.

6.3 Formalizing SCA and the Success Rate of Side-Channel Adversary: Guessing Entropy

In order to evaluate the several attack methods and also to compare the several cryptographic designs wrt these attacks several formal metrics have been developed. It is important to have an understanding of these security metrics which are based on formalizing these side-channel analysis techniques. In this section, we provide an overview on some of the fundamental metrics.

6.3.1 Success Rate of a Side-Channel Adversary

Side-channel adversaries work on a divide and conquer strategy, where the key space is divided into several equivalent classes. The attack is normally unable to distinguish keys which belong to the same class or partition, but is capable to distinguish between two keys which fall in different partitions. Thus one can formalize the adversary as an algorithm which targets a given implementation of a cryptographic algorithm, formally denoted as E_K, where K denotes the key space. The adversary also assumes a leakage model for the key denoted as L. The leakage model provides some information of the key or some other desirable information. Also the adversary is bounded in terms of computational resources and thus the adversary $A_{E_K,L}$ is an algorithm with time complexity τ, memory complexity m,

Algorithm 6.1: Formal definition of success rate of a side-channel attack

Input: K, L, $\mathsf{E_K}$
Output: 0 (failure), 1 (success)

$k \in_R K$
$s = \gamma(k)$
$\mathbf{g} = [g_1, \ldots, g_o] \leftarrow A_{E_k,L}$
if $s \in g$ **then**
  return 0
else
  return 1

making q queries to the target implementation of the cryptographic algorithm. Note that the leakage function is not capable of distinguishing certain keys, hence inducing a partition S on the entire key space K. The mapping from K to S can be captured by a function γ such that $s = \gamma(k)$, where $s \in S$ and $k \in K$, and $|S| << |K|$. The objective of the adversary is to determine the corresponding equivalence class to which a chosen key k belongs denoted as $s = \gamma(k)$ with a non-negligible probability.

As an analogy consider the Hamming Weight or Hamming-Distance leakage model, which divides the entire key space into equivalence classes or partitions, which the attacker tries to distinguish with sufficient observations. The output of the adversary is based on the ciphertext (black-box information) and the leakage (side-channel information) outputs a *guess vector*, which are the key classes sorted in terms of descending order of being a likely candidate. We thus define an order-o ($o \leq |S|$) adversary when the adversary produces the guessing vector as $\mathsf{g} = [g_1, \ldots, g_o]$, where g_1 is the most likely candidate, and so on. Now having defined the adversary let us try to understand formally the side-channel experiment to define the metrics.

More precisely, the side-channel attack is defined as an experiment $\mathrm{Exp}_{\mathsf{A_{E_K,L}}}$ where $A_{E_K,L}$ is an adversary with time complexity τ, memory complexity m, and making q queries to the target implementation of the cryptographic algorithm. In the experiment for any k chosen randomly from K, when the adversary $A_{E_k,L}$ outputs the guessing vector g, the attack is considered as a success if the corresponding key class denoted as $s = \gamma(k)$ is such that $s \in g$. More formally the experiment for a side-channel attack of order-o is as in Algorithm 6.1. The experiment returns 0 or 1 indicating a success or failure of the attack.

The oth order success rate of the side-channel attack $\mathsf{A_{E_K,L}}$ against the key classes or partitions S is defined as:

$$\mathbf{Succ}^{\mathbf{o}}_{\mathbf{A_{E_K,L}}}(\tau, m, k) = \mathsf{Pr}[\mathrm{Exp}_{A_{E_K,L}} = 1]$$

Algorithm 6.2: Formal definition of guessing entropy

Input: K, L, E_{K}
Output: Key class i

1 $k \in_R K$
2 $s = \gamma(k)$
3 $\mathbf{g} = [g_1, \ldots, g_o] \leftarrow A_{E_k,L}$
4 **return** i such that $g_i = s$

6.3.2 Guessing Entropy of an Adversary

The above metric for an oth order attack implies the success rate for an attack where the remaining workload is o-key classes. Thus the attacker has a maximum of o-key classes to which the required k may belong. While the above definition for a given order is fixed wrt the remaining work load, the following definition of guessing entropy provides a more flexible definition for the remaining work load. It measures the average number of key candidates to test after the attack. We formally state the definition, based on the adversaries experiment as in Algorithm 6.2.

The Guessing Entropy of the adversary $A_{E_k,L}$ against a key class variable S is defined as:

$$\mathbf{GE}_{\mathbf{A}_{\mathbf{E}_\mathbf{K},\mathbf{L}}}(\tau, m, k) = \mathsf{E}[\mathrm{Exp}_{\mathbf{A}_{\mathbf{E}_\mathbf{K},\mathbf{L}}}]$$

6.4 Leakage Detection in SCA Traces: NICV and SNR

A formal framework for side-channel analysis and the methodology to evaluate its success was discussed in the previous section. An effective SCA stands on three parameters: measurements, leakage model, and distinguisher. If any of these parameters are sub-optimal, then it has a direct impact on SCA and its success. SCA countermeasures are also designed on the same principle such that it is hard to estimate the optimal leakage model, distinguisher, or acquire quality measurements. This section focuses on assessing the quality of side-channel measurements. A practical problem in SCA is that the measured traces can be huge with multimillion sample points. Processing traces of such size requires time and computing power. In order to optimize the attack, it is important to identify a small set of the so-called zone of leakage or relevant point-of-interest (PoI). This process of finding PoI is known as leakage detection. Moreover as a first test, leakage detection also informs if the traces carry any relevant information or not, thus aiding in assessing the quality of measurements. In the following, we provide the theoretical background and rationale to normalized inter-class variance (NICV) as a leakage detection tool. The relation of NICV to signal-to-noise ratio (SNR) is also discussed followed by case study on practical implementations of AES.

6.4.1 Normalized Inter-Class Variance

Normalized inter-class variance (NICV) is a technique which is designed to detect relevant PoI in an SCA trace [6]. Detection of PoI is used to compress the trace which results in acceleration of SCA. NICV can be computed based on public parameters like plaintext or ciphertext and doesn't need a profiling or pre-characterization of the target device. Moreover, NICV is leakage model agnostic which means that it can be applied without the knowledge of the target implementation. These properties make NICV an ideal candidate for accessing quality of measurements. The technical background of NICV is discussed in the following.

A side-channel adversary acquired leakage measurement $Y \in \mathbb{R}$ corresponding to a public parameter X (let's say a byte of plaintext or ciphertext, i.e., $\mathcal{X} = \mathbb{F}_2^8$). In general, Y can be continuous, but X must be discrete (and $\mathcal{X}$ must be of finite cardinality). Then, for all leakage prediction function L of the leakage knowing the value of x taken by X (as per Proposition 5 in [8]), we have

$$\rho^2\left[L(X);Y\right] = \underbrace{\rho^2\left[L(X);\mathbb{E}\left[Y|X\right]\right]}_{0\leq\,\cdot\,\leq 1} \times \rho^2\left[\mathbb{E}\left[Y|X\right];Y\right] \quad . \tag{6.3}$$

Here, $\mathbb{E}$ and Var denote the expectation and the variance, respectively, whereas ρ represents correlation. Equation (6.3) was further simplified in Corollary 8 of [8] to derive:

$$\rho^2\left[\mathbb{E}\left[Y|X\right];Y\right] = \frac{\mathsf{Var}\left[\mathbb{E}\left[Y|X\right]\right]}{\mathsf{Var}\left[Y\right]} \quad , \tag{6.4}$$

The term in Eq. (6.4) is further called as the *normalized inter-class variance* (NICV). It is an ANOVA (ANalysis Of VAriance) F-test, as a ratio between the explained variance and the total variance.

From Eqs. (6.3) and (6.4), we can derive that for all prediction function $L : \mathbb{F}_2^8 \to \mathbb{R}$:

$$0 \leq \rho^2\left[L(X);Y\right] \leq \frac{\mathsf{Var}\left[\mathbb{E}\left[Y|X\right]\right]}{\mathsf{Var}\left[Y\right]} = \text{NICV} \leq 1 \quad . \tag{6.5}$$

Thus, NICV is the *envelop* or maximum of all possible correlations computable from X with Y or NICV denotes the best-case results. Although Eq. (6.5) contains an equality, however, it is almost impossible to achieve in practical scenario. The equality can occur if and only if $L(x) = \mathbb{E}\left[Y|X = x\right]$, i.e., L is the optimal prediction function. The difference can come from various practical reasons like:

– The adversary knows the exact prediction function, but not the correct key. For instance, let us assume the traces can be written as $Y = w_H(S(X \oplus k^\star)) + N$, where $k^\star \in \mathbb{F}_2^8$ is the correct key, $S : \mathbb{F}_2^8 \to \mathbb{F}_2^8$ is a substitution box, w_H is

the Hamming weight function, and N is some measurement noise, that typically follows a centered normal distribution $N \sim \mathcal{N}(0, \sigma^2)$. In this case, the optimal prediction function $L(x) = \mathbb{E}[Y|X = x]$ is equal to: $L(x) = w_H(S(X \oplus k^\star))$ (the only hypothesis on the noise is that it is *centered* and *mixed additively* with the sensitive variable). This argument is at the base of the soundness of CPA: $\forall k \neq k^\star, \rho[w_H(S(X \oplus k)); Y] \leq \rho[w_H(S(X \oplus k^\star)); Y] \leq \sqrt{\mathsf{Var}[\mathbb{E}[Y|X]] / \mathsf{Var}[Y]}$.

– The leakage model is incorrect, for instance, $L(x) = w_H(x \oplus k^\star)$, when $Y = w_H(S(x \oplus k^\star)) + N$.
– The leakage model is a close approximation of the actual leakage model, for instance, $L(x) = w_H(S(x \oplus k^\star))$, whereas actually $Y = \sum_{i=1}^{8} \beta_i \cdot S_i(x \oplus k^\star) + N$, where $\beta_i \approx 1$, but slightly deviate from one.

The distance between CPA and NICV, in non-information theoretic attacks (i.e., attacks in the proportional / ordinal scale, as opposed to the nominal scale [9]) is similar to the distance between perceived information (PI) and mutual information (MI) [10].

In practice, the (square) CPA value does not attain the NICV value, owing to noise and other imperfections. This is illustrated in Fig. 6.1.

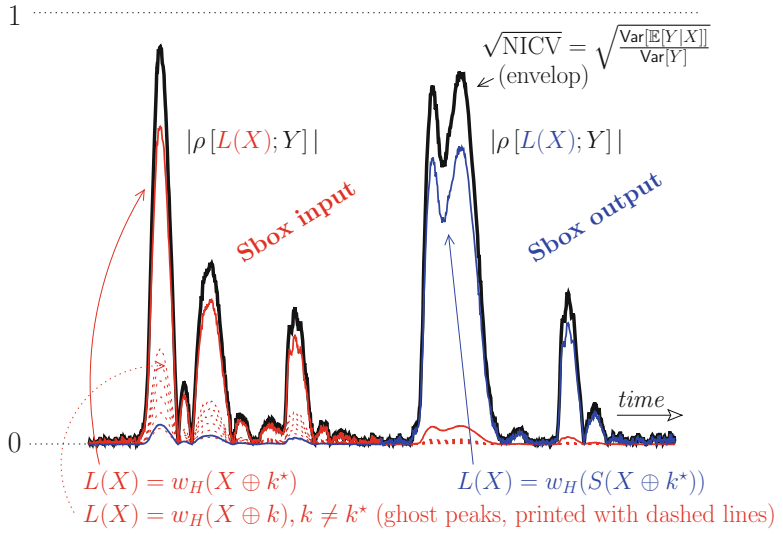

Fig. 6.1 NICV metric when $Y = \sum_{i=1}^{8} \beta_i \cdot S_i(x \oplus k^\star) + N$, and attack results for some prediction functions

6.4.2 NICV and SNR

NICV can be extended to establish relationship with signal-to-noise ratio (SNR), which is a widely used metric across domains. In the case of SCA, the signal refers to that part of the measurement which is directly related to sensitive data manipulation. When power measurements are considered, the signal is the current drawn for sensitive data computation. For instance, the measurement $Y = w_H(S(X \oplus k^\star)) + N$. The signal here directly refers to computation related to sensitive key $k^\star$ only. Note that a typical cryptographic algorithm will have several key bytes manipulated independently (16 for AES). The current drawn by computation related to other key bytes except $k^\star$ is considered noise and known as algorithmic noise. The noise N is the sum of algorithmic noise and measurement noise.

Now we derive the relation between NICV and SNR. It should be noted that we are considering only a byte of the ciphertext which has 2^8 number of possible values. If the public parameter X is uniformly distributed, the NICV in itself *is not* a distinguisher. Indeed, if we assume that $Y = L(X) + N = w_H(S(X \oplus k^\star)) + N$, then:

$$\begin{aligned}
\mathsf{Var}\left[\mathbb{E}\left[Y|X\right]\right] &= \textstyle\sum_{x\in\mathcal{X}} \mathsf{P}[X = x]\mathbb{E}\left[Y|X = x\right]^2 - \mathbb{E}\left[Y\right]^2 \\
&= \tfrac{1}{2^8} \textstyle\sum_{x\in\mathcal{X}} \mathbb{E}\left[w_H(S(x \oplus k^\star)) + N\right]^2 \\
&\quad - \left(\textstyle\sum_{x\in\mathcal{X}} \mathbb{E}\left[w_H(S(x \oplus k^\star)) + N\right]\right)^2 \\
&= \tfrac{1}{2^8} \textstyle\sum_{x'=x\oplus k^\star\in\mathcal{X}} \mathbb{E}\left[w_H(S(x'))\right]^2 \\
&\quad - \left(\textstyle\sum_{x'=x\oplus k^\star\in\mathcal{X}} \mathbb{E}\left[w_H(S(x'))\right]\right)^2 \\
&= \mathsf{Var}\left[w_H(S(X))\right].
\end{aligned}$$

Further elaborating denominator of NICV, $\mathsf{Var}\left[Y\right] = \mathsf{Var}\left[w_H(S(X))\right] + \mathsf{Var}\left[N\right]$.

To put both the equations together, we get

$$\boxed{\text{NICV} = \frac{\mathsf{Var}\left[\mathbb{E}\left[Y|X\right]\right]}{\mathsf{Var}\left[Y\right]} = \frac{1}{1 + \frac{1}{\text{SNR}}}} \,, \tag{6.6}$$

where the SNR is the ratio between:

- the signal, i.e., the variance of the informative part, namely $\mathsf{Var}\left[w_H(S(X \oplus k^\star))\right]$, and
- the noise, considered as the variance $\mathsf{Var}\left[N\right]$. An approximate estimation of noise is the intra-class variance, i.e., $\mathbb{E}\left[\mathsf{Var}\left[Y|X\right]\right]$.

Clearly, Eq. (6.6) does not depend on the secret key $k^\star$ as both Y and X are public parameters known to the attacker. Moreover, this expression is free from the leakage

model $L(X)$, which means that NICV does not depend on the implementation. Thus, NICV searches for all linear dependencies of public parameter X with available leakage traces Y independent of the implementation.

There are several advantages of computing NICV as compared to SNR. The main reason for using NICV over SNR is that, NICV is a bounded quantity. As shown in Eq. (6.5), the value of NICV lies in the range [0,1]. Thus the quality of the trace is measured directly as a percentage. Owing to this property, it is possible to compare the quality of measurements directly.

Next, the computation of SNR requires much more traces than NICV. The noise component $\mathsf{Var}[N]$ of SNR is mainly composed of the intra-class variance $\mathbb{E}[\mathsf{Var}[Y|X]]$. In order to properly estimate $\mathbb{E}[\mathsf{Var}[Y|X]]$, the adversary must acquire at least two measurements per class in order to compute the variance. Of course, it is desirable to have much more traces per class to compute the noise component of SNR. Since NICV depends on global variance of the measurements, it takes fewer traces to estimate.

6.4.3 Related Work in Leakage Detection

Initial methods for leakage detection were based on templates [11]. In this method, PoI are those which maximizes $\sum_{i,j=1}^{n}(T_i - T_j)$, for T templates for n subkey values. T_i is the average of the traces when the sensitive variable belongs to the class i. It was improved to Sum Of Squared pairwise Differences (SOSD [12]) as $\sum_{i,j=1}^{n}(T_i - T_j)^2$, to avoid cancellation of leakage of opposite polarity. SOSD was then extended to Sum Of Squared pairwise T-differences (SOST [12]) by normalizing it with variances as $\sum_{i,j=1}^{n}\left(\frac{T_i - T_j}{\sqrt{\frac{\sigma_i^2}{m_i} + \frac{\sigma_j^2}{m_j}}}\right)^2$. Here σ_i is the variance of T in class i, and m_i is the number of samples in class i. However, template based method needs an access to clones of a device.

6.4.4 Case Study: Application on AES

The main application of NICV is to find the interesting time samples for accelerating SCA. An SCA trace can easily have millions of points, however, only few of these points qualify as PoI. It is of interest of the adversary or evaluator to detect these PoI and compress the final measurement.

The NICV metric is first tested on a software implementation of AES-256 running on an ATMEL AVR microcontroller. A standard trace of this implementation contains seven million points and needs roughly 5.3 Mbytes of disk space when stored in the most compressed format. These details are in respect to a LeCroy wavescanner 6100A oscilloscope with a bandwidth of 1 GHz. The NICV is then

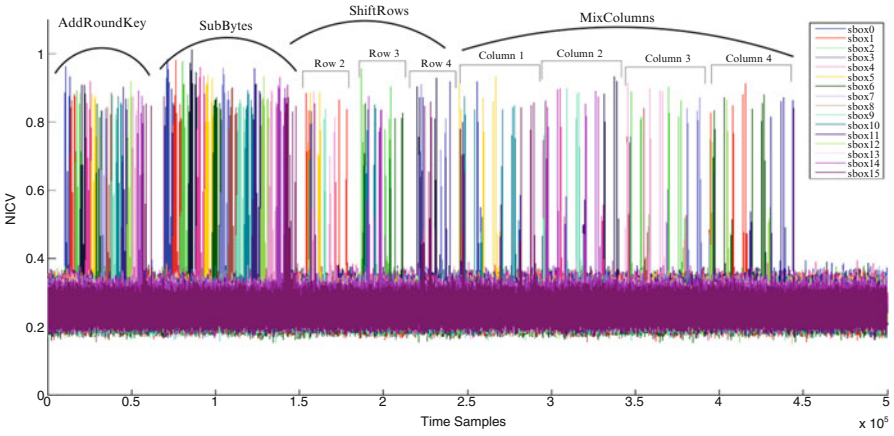

Fig. 6.2 NICV computed for an AES-128 software implementation to detect each round operation

applied to identify operations related to each byte of sensitive key $k_0^\star - k_{15}{}^\star$. Figure 6.2 shows the computation of NICV on the first round. The computations of $k_0^\star$ for round 1 take only $\approx$ 1000 time samples. Since each $k^\star$ is associated with a different substitution box (Sbox), the bytes are labeled as the Sbox number, i.e., Sbox0 to Sbox15. Once the interesting time samples corresponding to each executed operation is known, the trace size is compressed from 7,000,000 to 1000, i.e., a gain of roughly 7000×. NICV also enables us to reverse engineer the underlying software implementation. We computed NICV for all the 16 bytes of the plaintext and plotted the 16 NICV curves in Fig. 6.2 (depicted in different colors). By closely observing Fig. 6.2, we can distinguish individual operations from the sequence of byte execution. Each NICV curve (each color) shows all sensitive leakages related to that particular byte. Moreover, with a little knowledge of the algorithm, one can easily follow the execution of the algorithm. For example, the execution of all the bytes in a particular sequence indicates the SubBytes or AddRoundKey operation. Manipulation of bytes in sequence $\{1, 5, 9, 13\}, \{2, 6, 10, 14\}$, and $\{3, 7, 11, 15\}$ indicates the ShiftRows operations. The ShiftRows operation of AES shifts circularly 3 out of 4 rows with different constant. This can be clearly seen in Fig. 6.2: only three rows are manipulated and the bytes in the first row, i.e., $\{0, 4, 8, 12\}$ are not used during this time. Similarly MixColumns can also be identified by just looking at the bytes manipulated together.

Another common problem in SCA is the choice of leakage model which directly affects the efficiency of the attack. As shown in Sect. 6.4.1, the square of the correlation between modeled leakage ($L(X, K)$) and traces ($Y = L(X, K^\star) + N$) is smaller or equal to NICV, where N represents a noise. The equality exists only if the modeled leakage is the same as the traces. A real implementation can have different active components at different point of time. For example, the activity corresponding to Sbox input computation will be at different time than Sbox output computation.

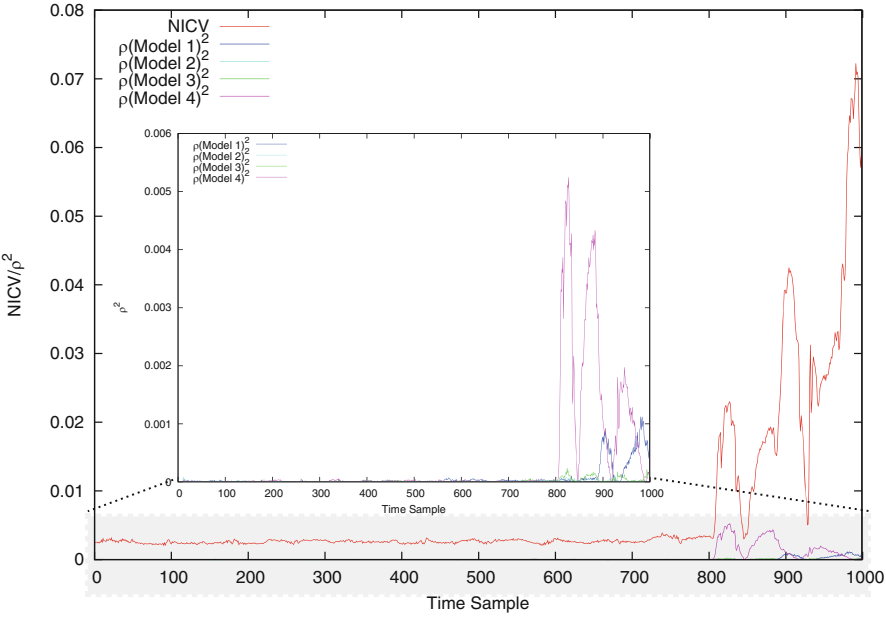

Fig. 6.3 NICV vs ρ^2 of four different models

If we test the implementation against these two models, leakage corresponding to each model shall be visible on the measurement at different times.

We tested the model on an AES-128 hardware implementation using two different leakage models on an FPGA and acquired SCA traces. The first leakage model corresponds to the state register present before the Sbox operation of AES, i.e., $w_H(\text{val}_i \oplus \text{val}_f) \in [\![0, 8]\!]$ (Model 1) and $\text{val}_i \oplus val_f \in [\![0, 255]\!]$ (Model 2). w_H is the Hamming weight function. Similar models are built for another register which is intentionally introduced at the output of the Sbox, i.e., $w_H(S(\text{val}_i) \oplus S(\text{val}_f)) \in [\![0, 8]\!]$ (Model 3) and $S(\text{val}_i) \oplus S(\text{val}_f) \in [\![0, 255]\!]$ (Model 4). Figure 6.3 shows the square of correlation of four different leakage models with the traces against the NICV curve. It can be simply inferred from Fig. 6.3 that Model 4 performs the best while Model 2 is the worst. The gap between NICV and $\rho(\text{Model } 4)^2$ is quite large due to reasons mentioned in Sect. 6.4.1. Thus NICV depicts leakage corresponding to several leakage model in the measurements.

6.5 Test Vector Leakage Assessment Methodology

The previous section discussed NICV as a leakage detection technique. Leakage detection is a pre-processing step in the side-channel analysis, which helps in locating PoI and thus improving attack efficiency. However, it does not provide

conclusive output about security of the IP. In order to evaluate the security, one must rely on leakage assessment.

When an IP has to be certified for security evaluation, a certification lab must conduct series of SCA on the IP under test. The list must be updated regularly to keep track of state-of-the-art vulnerabilities. These attacks might involve applying different distinguishers like correlation, mutual information or linear regression [13], etc. Moreover, several leakage models must be tested. The range of attacks and leakage model makes certification a time-consuming and non-comprehensive task. Moreover, it can be error prone as it highly depends on choice of distinguisher and model.

The aforementioned shortcomings motivate the need of a generic testing method which could effectively evaluate the security of the IP. Leakage assessment is a methodology which checks for potential side-channel leakage in the target without actually performing the attack. The methodology is based on the fact that SCA exploits side-channel information corresponding to sensitive intermediate values. This presence of side-channel information can be statistically detected using hypothesis testing. The seminal work which proposed this methodology named it Test Vector Leakage Assessment (TVLA [7]).

TVLA partitions the side-channel measurements of traces Y on the basis on plaintext. In order to compute TVLA, one must acquire two sets of traces. While one set corresponds to a fixed key and fixed plaintext as input to the cryptographic IP, the second set collects traces corresponding to fixed key and random plaintext. Thereafter a hypothesis testing performed by assuming a null hypothesis that the two sets of traces have identical means and variance. Background on hypothesis testing was previously discussed in Sect. 6.2. If the null hypothesis is accepted, it signifies that the traces carry no sensitive information. On the other hand, a rejected null hypothesis indicates presence of exploitable leakage. In [7], Welch's t-test is used for statistical testing and the technique is called non-specific t-test. This can be expressed as:

$$\text{TVLA} = \frac{\mathbb{E}\left[Y_r\right] - \mathbb{E}\left[Y_f\right]}{\sqrt{\frac{\text{Var}[Y_r]}{m_r} + \frac{\text{Var}[Y_f]}{m_f}}} = \frac{\mu_r - \mu_f}{\sqrt{\frac{\sigma_r^2}{m_r} + \frac{\sigma_f^2}{m_f}}}\ , \tag{6.7}$$

where Y_r and Y_f correspond to set of traces with random and fixed plaintext, respectively. m_r, m_f signifies the number of traces in set Y_r, Y_f, respectively. The mean and standard deviation of set Y_r is denoted by μ_r and σ_r. Similarly, μ_f and σ_f refer to mean and standard deviation of Y_f. The TVLA value must be contained with the range $\pm R$, to accept the null hypothesis. If the TVLA value exceeds the absolute threshold R in either polarity, the device is considered to leak sensitive side-channel information. The intuitive value generally used is $R = 4.5$, which for large $m(> 5000)$ accepts or rejects the null hypothesis with a confidence of 99.9999 %. So at any time, if the TVLA value of the target IP is either less than -4.5 or more than 4.5, it can be rendered as leaking side-channel information.

The non-specific *t*-test assess the leakage in the target IP without performing an attack and without any hypothesis on the underlying implementation. It finally provides a level of confidence on the presence of exploitable leakage from the target. However no information is provided about the source of leakage and methodology to exploit that leakage. Further on, the author proposes specific *t*-test testing, which is partitioned based on sensitive intermediate value rather than public plaintext, to potentially find the source of leakage.

TVLA was originally demonstrated on first order leakage in a univariate setting [7]. This work was further extended to a multivariate setting in [14] along with performance enhancement for efficient computation of TVLA. Another aspect of *T*-test based assessment was explored in [15]. While [7, 14] used an unpaired *T*-test, the use of paired *T*-test was proposed in [15]. The main argument for switching to paired *T*-test (TVLA_P) is that it is less sensitive to environmental noise and gradual temperature changes. It can be expressed as:

$$\text{TVLA}_P = \frac{\mu_d}{\sqrt{\frac{\sigma_d^2}{m}}} \ , \tag{6.8}$$

μ_d and σ_d are mean and standard deviation of paired difference of Y_r and Y_f. The paired *T*-test was also combined with a moving average based computation, leading to better results for multivariate setting. Nevertheless, the moving average improvement can also be applied to unpaired *T*-Test.

6.6 Equivalence of NICV and TVLA

TVLA and NICV were previously discussed in detail. In this section, we derive the relationship between these two metrics. This allows us to estimate one metric by knowledge of the other. TVLA is a simple *T-test*. Actually if we look closely, NICV is similar to statistical *F-test*. Now, in case of statistical *T-test*, we only consider two different classes whereas in case of statistical *F-test*, we consider multiple different classes. In the scenario of two class statistical *F-test*, the result of statistical *F-test* is found to be proportional to square of the statistical *T-test* result. This shows a clear relationship between NICV and TVLA. Moreover, this also exhibits relationship between TVLA and SNR as from NICV, we can easily calculate the SNR of the underlying attack setup. In the following, we will try to obtain the exact relationship between NICV and TVLA.

Let us assume that we have some side-channel traces which belong to two different classes: class 1 and class 2. Both the classes have same cardinality *N*. The mean of class 1 is μ_1 and mean of class 2 is μ_2. Now, the computation of NICV

is as follows:

$$\text{NICV} = \frac{\frac{1}{2}\sum_{i=1}^{2}(\mu_i - \mu)^2}{\frac{1}{2N}\sum_{i=1}^{2N}(x_i - \mu)^2} \tag{6.9}$$

Similarly we can compute TVLA^2 as follows:

$$\text{TVLA}^2 = \frac{(\mu_1 - \mu_2)^2}{\frac{V_1}{N} + \frac{V_2}{N}} = \frac{K}{\frac{V_1}{N} + \frac{V_2}{N}} \tag{6.10}$$

where V_1, V_2 are variance of class 1 and class 2, respectively, and $K = (\mu_1 - \mu_2)^2$. μ is the mean of the all side-channel traces and is equal to $\mu = \frac{\mu_1 + \mu_2}{2}$. Now we will consider only the numerator part of the NICV formulation which is

$$\begin{aligned}
&\frac{1}{2}\sum_{i=1}^{2}(\mu_i - \mu)^2 \\
&= \frac{1}{2}((\mu_1 - \mu)^2 + (\mu_2 - \mu)^2) \\
&= \frac{1}{2}\left(\left(\mu_1 - \frac{\mu_1 + \mu_2}{2}\right)^2 + \left(\mu_2 - \frac{\mu_1 + \mu_2}{2}\right)^2\right) \\
&= \frac{1}{4}(\mu_1 - \mu_2)^2 = \frac{K}{4}
\end{aligned} \tag{6.11}$$

Next we will consider the denominator part of the NICV computation which is as follows:

$$\begin{aligned}
&\frac{1}{2N}\sum_{i=1}^{i=2N}(x_i - \mu)^2 \\
&= \frac{1}{2N}\sum_{i=1}^{i=2N}(x_i - \frac{\mu_1 + \mu_2}{2})^2 \\
&= \frac{1}{2N}\sum_{i=1}^{i=2N}(x_i^2 - x_i(\mu_1 + \mu_2) + \frac{(\mu_1 + \mu_2)^2}{4}) \\
&= \frac{1}{2N}\sum_{i=1}^{i=2N}(x_i^2 - x_i(\mu_1 + \mu_2)) + \frac{(\mu_1 + \mu_2)^2}{4}
\end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{2N}\sum_{x_i \in class1}(x_i^2 - x_i(\mu_1+\mu_2)) + \frac{1}{2N}\sum_{x_i \in class2}(x_i^2 - x_i(\mu_1+\mu_2)) + \frac{(\mu_1+\mu_2)^2}{4} \\
&= \frac{1}{2N}\sum_{x_i \in class1}((x_i-\mu_1)^2 - \mu_1^2 + \mu_1 x_i - \mu_2 x_i) + \\
&\quad \frac{1}{2N}\sum_{x_i \in class2}((x_i-\mu_2)^2 - \mu_2^2 + \mu_2 x_i - \mu_1 x_i) + \frac{(\mu_1+\mu_2)^2}{4} \\
&= \frac{1}{2N}\sum_{x_i \in class1}(x_i-\mu_1)^2 + \frac{1}{2N}\sum_{x_i \in class2}(x_i-\mu_2)^2 + \frac{1}{2N}\sum_{x_i \in class1} x_i(\mu_1-\mu_2) + \\
&\quad \frac{1}{2N}\sum_{x_i \in class2} x_i(\mu_2-\mu_1) + \frac{(\mu_1+\mu_2)^2}{4} - \frac{\mu_1^2}{2} - \frac{\mu_2^2}{2} \\
&= \frac{V_1}{2} + \frac{V_2}{2} + \frac{1}{2}(\mu_1-\mu_2)\mu_1 + \frac{1}{2}(\mu_2-\mu_1)\mu_2 + \frac{(\mu_1+\mu_2)^2}{4} - \frac{\mu_1^2}{2} - \frac{\mu_2^2}{2} \\
&= \frac{V_1}{2} + \frac{V_2}{2} + \frac{K}{4}
\end{aligned}
\tag{6.12}
$$

Hence the NICV value can be re-written as below

$$
\begin{aligned}
\text{NICV} &= \frac{\frac{K}{4}}{\frac{V_1}{2} + \frac{V_2}{2} + \frac{K}{4}} \\
&= \frac{1}{\frac{2(V_1+V_2)}{K} + 1} \\
&= \frac{1}{\frac{2N}{\text{TVLA}^2} + 1} \\
&\propto \text{TVLA}^2
\end{aligned}
\tag{6.13}
$$

This demonstrates that NICV value is directly proportional to square of TVLA value.

6.7 TVLA on Higher Order Side-Channel Attacks

In this section, we will focus on application of TVLA on higher order side channel analysis. The TVLA methodology, discussed in the previous sections is applicable to first order side-channel analysis where the adversary is allowed to exploit leakage of only one intermediate value. Using this methodology, we can analyze first order

side-channel vulnerability of any crypto-system. Additionally, we can also validate side-channel security of any countermeasure which claims to be resistant against first order attacks. However, any countermeasure which prevents first order attack can be vulnerable against higher order attacks. Hence, it is imperative to analyze these countermeasures against higher order attacks and for this we also need to update the formulation of TVLA metric computation. In [14], authors have proposed a detailed description of how to modify the TVLA metric for higher order attacks. In this section, we will give a brief overview of that formulation. Now, the difference between first order and higher order side-channel analysis is that in case of higher order analysis we need to pre-process the side-channel analysis. For example, in case of second order side-channel analysis, we need to convert the traces into mean free squared traces. To compute TVLA on higher order, we need to estimate mean and variance of these pre-processed side-channel traces.

6.7.1 Estimation of Mean

In this section, we will first describe the estimation of first parameter of TVLA for higher order analysis which is mean in-case of first order of analysis. For higher order analysis, we need to pre-process the side-channel traces and then need to compute the mean of pre-processed side-channel traces. In this subsection, our objective is to estimate the mean of the pre-processed side-channel traces for higher order analysis.

A dth order side-channel attack exploits leakage of d different intermediate variables and tries to get access to the secret key. To formulate the corresponding TVLA metric for dth order side-channel analysis we introduce the following notations:

1. M_d= dth order raw statistical moment of a random variable X. M_d is computed as follows:

$$M_d = \mathbb{E}\left[X^d\right] \tag{6.14}$$

   where $\mathbb{E}[.]$ is the expectation operator. For $d = 1$, $M_1 = \mu$ (mean).
2. CM_d= dth order central moment ($d > 1$) of random variable X. CM_d is computed as follows:

$$\mathrm{CM}_d = \mathbb{E}\left[(X - \mu)^d\right] \tag{6.15}$$

   For $d = 2$, CM_2= variance of the distribution.
3. SM_d= dth order standardized moment ($d > 2$) of random variable X. SM_d is computed as follows:

$$\mathrm{SM}_d = \mathbb{E}\left[\left(\frac{X - \mu}{\sigma}\right)^d\right] \tag{6.16}$$

   where σ is the standard deviation of the distribution.

In case of first order test, we use mean of the two different classes which is actually raw statistical moment of them. In case of second order, we will use CM_2 which is actually the variance of the side-channel traces. For third and higher order of analysis we will require SM_d of side-channel traces. Now, the most efficient way of calculating these statistical metrics is to use online one pass algorithms which compute the value of these metrics for each new trace, added to the distribution. This actually allows computation of these metrics during the acquisition of side-channel traces and saves storage as well as postprocessing time. There are various ways by which a user can compute the discussed statistical metrics in one pass fashion. But care should be taken to avoid any numerically unstable steps as that will introduce errors in the computation. A stable incremental one pass algorithm was proposed in [14] which we will describe next.

Let us assume that $M_{1,Q}$ denote the raw statistical moment of a given set Q. Additionally. let y be the new acquired side-channel trace added to the set Q. Let $M_{1,Q'}$ be the new raw statistical moment of the enlarged set Q'. This can be computed as follows:

$$M_{1,Q'} = M_{1,Q} + \frac{\Delta}{n} \tag{6.17}$$

where $\Delta = y - M_{1,Q}$ and n denotes the number of side-channel traces in enlarged set Q'. Similarly we can estimate central moment (CM_d) using one pass algorithm. However to do that, we need to estimate another metric central sum (CS_d) which can be computed as follows:

$$\mathrm{CS}_d = \sum (x_i - \mu)^d \tag{6.18}$$

From CS_d, we can calculate CM_d as follows:

$$\mathrm{CM}_d = \frac{\mathrm{CS}_d}{n} \tag{6.19}$$

The one pass incremental computation for CS_d is given below:

$$\mathrm{CS}_{d,Q'} = \mathrm{CS}_{d,Q} + \sum_{k=1}^{d-2} \binom{d}{k} \mathrm{CS}_{d-k,Q} \left(\frac{-\Delta}{n}\right)^k + \left(\frac{n-1 \times \Delta}{n}\right)^d \left[1 - \left(\frac{-1}{n-1}\right)^{d-1}\right] \tag{6.20}$$

Finally, we can compute SM_d by using the following formula:

$$\mathrm{SM}_d = \frac{\mathrm{CM}_d}{(\sqrt{(\mathrm{CM}_2)^d}} \tag{6.21}$$

Using the above described equations, we now have formulations for computing CM_d and SM_d in one pass. This allows us to estimate the mean of the pre-processed side-channel traces for higher order analysis. For second order, it is the variance of the side-channel traces. For higher order, we need to compute SM_d which will act as the mean of the pre-processed side-channel traces.

6.7.2 Estimation of Variance

In this subsection, our focus will be on the estimation of the variance of the pre-process side-channel traces. In case of second order analysis, the side-channel trace set Y needs to be converted into mean free squared, i.e., $(Y-\mu)^2$. Variance of this mean free squared traces can be calculated as follows:

$$\begin{aligned}
\mathsf{Var}\left[(Y-\mu)^2\right] &= \frac{1}{n}\sum\left((y-\mu)^2 - \frac{1}{n}\sum(y-\mu)^2\right)^2 \\
&= \frac{1}{n}\sum\left((y-\mu)^2 - \mathrm{CM}_2\right)^2 \\
&= \frac{1}{n}\sum(y-\mu)^4 - \frac{2}{n}\mathrm{CM}_2\sum(x-\mu)^2 + \mathrm{CM}_2^2 \\
&= \mathrm{CM}_4 - \mathrm{CM}_2^2
\end{aligned} \tag{6.22}$$

For higher order analysis (for example, dth order), where the traces are needed to be standardized, we first compute $Z = \left(\frac{Y-\mu}{\sigma}\right)^d$. The computation of variance of Z can be done as follows:

$$\begin{aligned}
\mathsf{Var}\left[Z^2\right] &= \mathrm{SM}_{2d} - \mathrm{SM}_d^2 \\
&= \frac{\mathrm{CM}_{2d} - \mathrm{CM}_d^2}{\mathrm{CM}_2^d}
\end{aligned} \tag{6.23}$$

Now, with this formulation we can now estimate mean and variance of pre-processed side-channel traces for any arbitrary order side-channel analysis and once we have them, we can compute TVLA very easily. These incremental one pass computations are not only beneficiary for saving storage and computation time, but also can be utilized for efficient computation of TVLA on chip [16]. For more detailed analysis on applying TVLA on higher order side-channel analysis, the readers are encouraged to read [14].

6.8 Case Study: Private Circuit

In the previous sections, we have discussed about different metrics which have been proposed for evaluation of side-channel vulnerability. We have also highlighted how two such metrics, TVLA and NICV, are actually equivalent, and can be used interchangeably. In this section, we will provide a case study where we employ TVLA to asses side-channel security of a side-channel secured crypto-implementation. Our case study will be on the seminal work [17] of Ishai, Sahai, and Wagner where they proposed a theoretical countermeasure against probing attack, which is the strongest form of SCA. Although the proposed countermeasure [17], here after referred to as *ISW scheme* or t-private circuits, is suited for probing attack, it can be used to prevent power or Electromagnetic SCA. Probing attack considers a strong adversary who is capable of observing exact values of one or several internal nets of the circuit including the nets containing sensitive information. In case of SCA, the adversary cannot observe the exact value of the sensitive information (key bits or key-dependent nets) but a linear or non-linear transformation of sensitive values like Hamming weight or Hamming-distance. Thus this countermeasure which prevents much stronger probing attack can also be used to prevent passive side-channel attacks like power or EM leakage based attack. Private circuits also form the basis of much studied countermeasures against SCA known as masking [18]. In particular, Rivain and Prouff [18] had proposed a dth order provably secure masking scheme for AES. This masking scheme was derived by optimizing the hardware-oriented t-private circuits for software implementations. The t-private circuits are secure against an adversary capable of observing any t nets of the circuit at any given time instant. Construction of t-private circuits involves $2t$ number of random bits for each input bit of the circuit. Moreover, it requires $2t + 1$ random bits for each 2-input *AND* gate present in the circuit. The overall complexity of the design is $\mathcal{O}(nt^2)$ where n is the number of gates in the circuit.

The countermeasure, proposed in [17] is based on sound theoretical proof but with some inherent assumptions which may not be valid in practical scenario. In this case study, we will try to identify the practical scenarios in which private circuit may fail to provide us the desired security. We would like to state that we are not making any claim against security of private circuit, but we are pointing out the dangerous situations where expected security can be compromised. For this we have implemented a lightweight block cipher *SIMON* [19] using private circuit methodology on SASEBO-GII board. As we evaluate private circuits in a pure hardware setting, we choose to use the original ISW scheme as presented in [17] to be fair towards various schemes branched out of it. Moreover, the implementation in [20] is primarily proposed for FPGA target. However, we intend to study the security of private circuits in general. The choice of block cipher is motivated by its compact design and simplistic construction using basic gates like *AND* and *XOR*.

Detailed case study on side-channel security of private circuits in practical scenarios can be found in [21]. The implemented private circuits are analyzed against SCA using EM traces and correlation power analysis [22]. We primarily

have implemented three different versions of *SIMON* on FPGA using private circuit, which are as follows:

- *Optimized SIMON*: In this case, SIMON is implemented according to the ISW scheme [17], but the design tool is free to optimize the circuit. As our implementation platform is a Virtex-5 FPGA which has six-input Look-up table (LUT), design tool (in our case Xilinx ISE) will optimize the circuit to reduce the resource requirement of the design. This is an example of *lazy engineering* approach, where designer is allowing the tool to do modification on the design without being aware of its possible impact on the security of private circuits.
- 2-*input LUT based SIMON*: Here, to mimic the private circuit methodology exactly on the FPGA, we have constrained the design tool to map each two-input gate to a single LUT. In other words, though an LUT has six inputs, it is modeled as two-input gate and gate-level optimization is minimized.
- *Synchronized 2-input LUT based SIMON*: This is nearly similar to the previous methodology. The only difference is that each gate or LUT is preceded and followed by flip-flops so that each and every input to the gates is synchronized and glitches are minimized (if not suppressed, see [23]).

We will show that among these three, *Optimized SIMON* can be broken using CPA whereas, 2-*input LUT based SIMON* is resistant against CPA, but fails *TVLA* test. Finally we will show that *Synchronized* 2-*input LUT based SIMON* is not only resistant against CPA, but also passes *TVLA* test [24, 25].

We could have based our study on more popular ciphers like AES or PRESENT, but with the FPGA in consideration, it was not possible to fit protected designs when using constraints *LOCK_PINS* and *KEEP*. In fact, we had to switch from SIMON64/96 to SIMON32/64, to be able to fit all the designs on the FPGA.

6.8.1 Experimental Analysis and Result

In this section, we perform practical evaluation of *private circuit* by executing SCA on block cipher *SIMON*. We start with the experimental setup, followed by the analysis of the obtained result on various implementation of private circuits.

To perform SCA, side-channel traces are acquired using an EM antenna of the HZ–15 kit from Langer, placed over a power decoupling capacitor. The traces are captured on a 54855 Infiniium Agilent oscilloscope at a sampling rate of 2 GSample/s. In the following subsections we will discuss the result of SCA on different variants of *t-private SIMON*.

6.8.1.1 Optimized SIMON

In this setting, we implemented SIMON using private circuit methodology with $t = 1$. At this stage, the FPGA tool was not constrained and was allowed to do

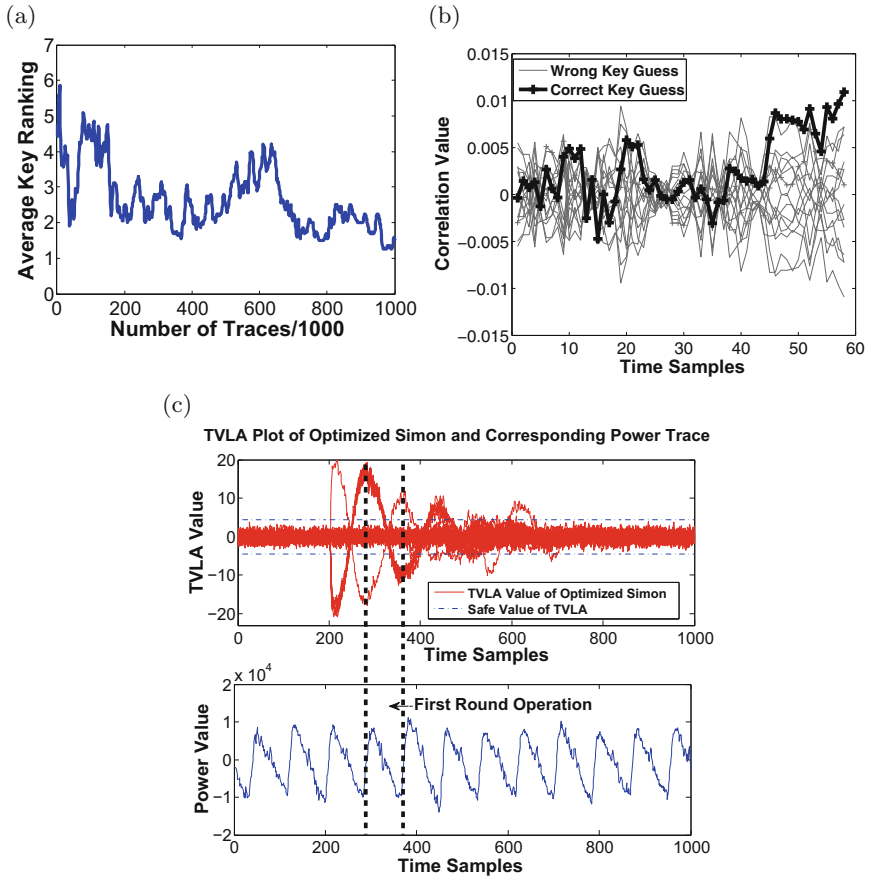

Fig. 6.4 Side-channel analysis of optimized SIMON. (**a**) Average key ranking. (**b**) Correlation Value of Key Guesses. (**c**) TVLA plot

all optimizations if possible. We collected 100,000 EM traces and performed *TVLA* test, along with *CPA*. *TVLA* test is performed on the first round output of *SIMON* for each bit of plaintext. Basically, the traces are partitioned into 2 sets on the basis of value of concerned bit of round output, i.e., 1 or 0. Thereafter, we compute the means (μ_1, μ_0) and standard deviations (σ_1, σ_0) of the two sets and we compute the corresponding TVLA value. The result can be seen in Fig. 6.4c. The first few peaks in the plot are due to the plaintext loading, however, after that it can be seen that for some bits, value of TVLA is much larger than the safe value of TVLA, i.e., ± 4.5 for secure system [24, 25]. Thus this implementation of SIMON is vulnerable to SCA though it is designed by private circuit. The leakage comes due to the optimizations applied by CAD tools. To validate our claim further, we carried out CPA around the sample points where TVLA peaks can be observed. We have chosen

Hamming-distance as our attack model and we targeted first round of *SIMON32/64*. The leakage model can be written as follows:

$$L_{\mathrm{HD}} = w_H(R(x_{i+1}, x_i) \oplus key_i \oplus x_{i+1}),$$

where x_i, x_{i+1} are parts of plaintext, key_i is the round key, and R is the round function. The first round operation of *SIMON32/64* involves 16 bits of key and we try to recover key nibble by nibble. Now, generally for a nibble, key space is 16; however, as SIMON has no non-linear operation on the key in first round, for each key guess there is another key candidate which has exactly same value of correlation coefficient with opposite polarity. Due to this symmetry, total key space reduces from 16 to 8. To measure the success rate, we have calculated *average key ranking* (or equivalently we can calculate guessing entropy) over all the nibbles at intervals of 1000 power traces and the corresponding result is shown in Fig. 6.4a. At the end of 100,000 traces, we have been able to recover the correct key for two nibbles and for the rest of the two nibbles, ranking of correct key is 2, which clearly shows successful attack. Figure 6.4b shows the correlation values of different key guesses. Though the gap between the correct key and wrong key guesses is marginal, it is consistent as indicated by Fig. 6.4b, providing us enough evidence to conclude the attack as successful. The reason for this small nearest rival distance is the Hamming-distance attack model which does not incorporate randomization of private circuit. The key ranking curve is not very smooth. Theoretically addition of traces should lower the key ranking, but the reverse phenomenon can happen in practical scenario, as it depends upon the quality of the acquired traces. CPA is a statistical analysis and this phenomenon is statistical artifact. Moreover, the average key ranking will depend on the combined progression of all the 4 nibbles. Nevertheless, due to the CAD tool optimization, it is possible to successfully retrieve secret information from private circuit (Fig. 6.5).

6.8.1.2 2-Input LUT Based SIMON

In the previous discussion, we provided experimental validations of the disastrous effect of CAD tool optimization on private circuits. Designer can use various attributes and constraints to disable the optimization property of CAD tools, and hence can mimic the private circuit more efficiently (when the input HDL file is described structurally with LUTs). In this implementation, we have constrained the Xilinx ISE tool to treat each LUT as a 2-input gate using `KEEP` and `LOCK_PIN` attributes. *SIMON 32/64* is then designed using this 2-input LUT gates so that no optimizations can be applied by the CAD tools. Similar SCA is carried out for this implementation also and corresponding result is shown in Fig. 6.6. As we can see, TVLA plot still shows occurrence of information leakage, though it is less significant compared to information leakage observed for *Optimized SIMON*. Average key ranking plot (Fig. 6.6b) also shows improvement in terms of more resistance against SCA. However, as there is still information leakage, as shown

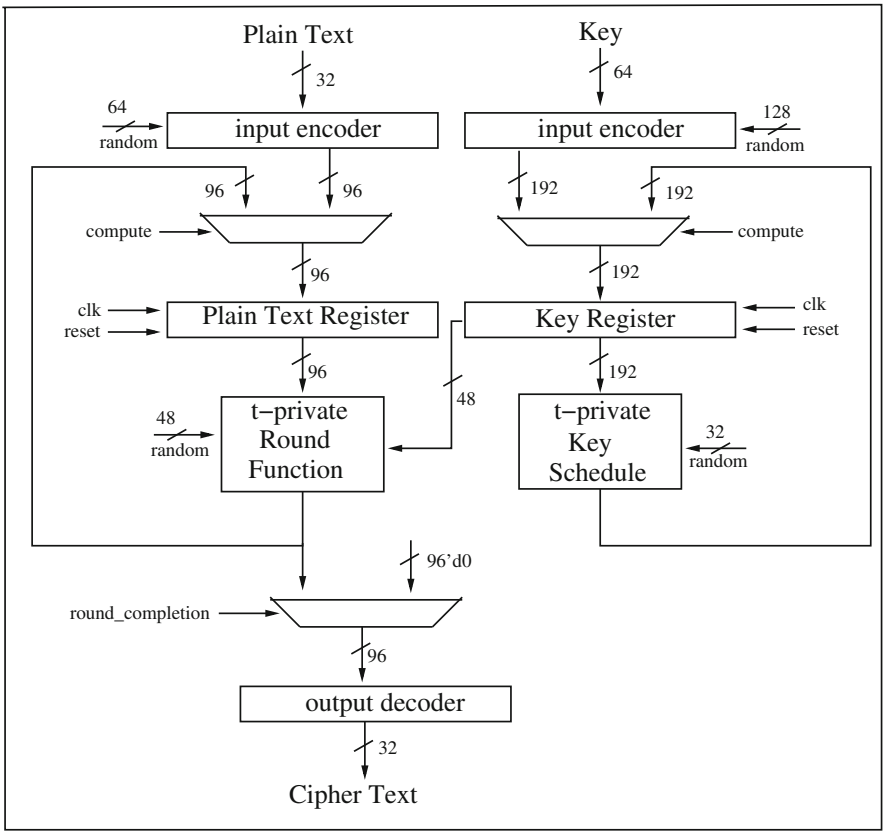

Fig. 6.5 Architectural overview of SIMON

in TVLA plot, the system could be broken by attack which employs better model. In other words, we cannot be absolutely confident about the side-channel resistance of the implementation.

6.8.1.3 Synchronized 2-Input LUT Based SIMON

In the previous discussion, we have shown how side-channel attack is resisted by 2-input LUT based SIMON. However, TVLA plot of 2-input LUT based SIMON still shows some information leakage. In this subsection, we tackle this issue.

It was shown in [21] how delay in random variables can disrupt the security of private circuit. In our implementation, random variables are generated using a maximum length LFSR and we suspect that asynchronous random variables as the reason of the information leakage. To analyze this, we have made a new

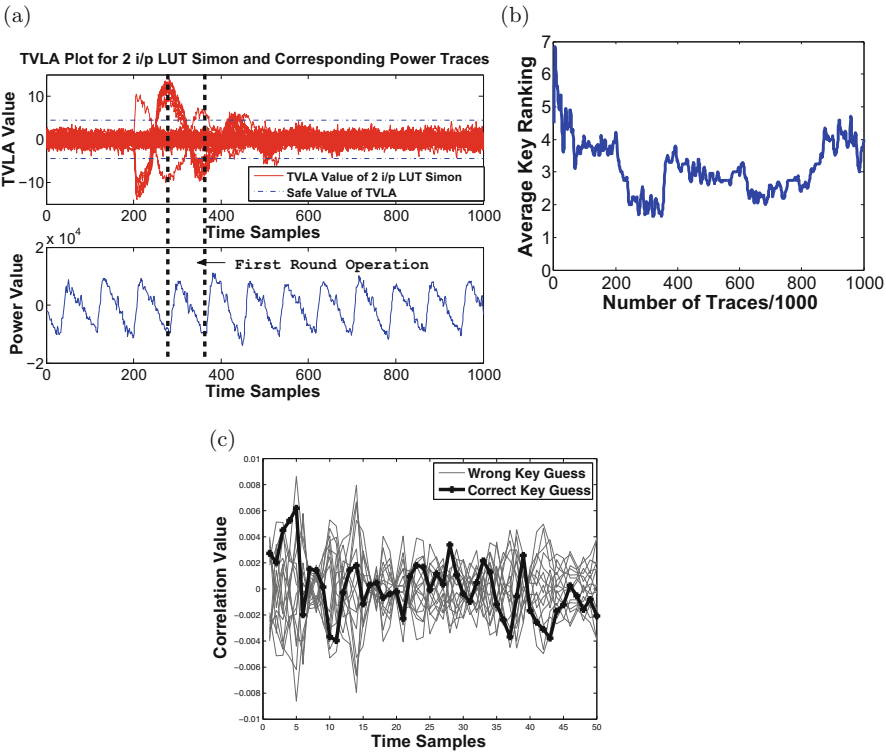

Fig. 6.6 Side-channel analysis of 2-input LUT SIMON. (**a**) TVLA plot. (**b**) Average key ranking. (**c**) Correlation value of key guesses

implementation where each 2-input LUT is followed by a flip-flop,[1] so that if there is any delay or glitches on random variables, it will be handled locally and will not propagate across the circuit. We have carried out similar side-channel analysis on this implementation and the result can be seen in Fig. 6.7. Now before analyzing the result we will like to state that this implementation has more clock cycle requirement compared to previous two implementations due to the presence of flip-flops after each 2-input LUT. For example, in the previous two implementations, first round operation occurs just after the start of encryption, whereas in this implementation first round output is obtained in the 9-th clock cycle after the start of encryption. As we are doing TVLA test on the first round output of SIMON, TVLA peak should appear near the clock cycle where first round output is obtained. As we can see, in TVLA plot (Fig. 6.7a), there are few peaks near the start of encryption

[1] In the *ISW* scheme [17], logic gates of *stateless circuits* are combinational instances, which, by design, evaluate as soon as one input changes. Our addition of the flip-flop after each combinational gate ensures that they evaluate only once, which is the legal mode of operation for *ISW* to be sound.

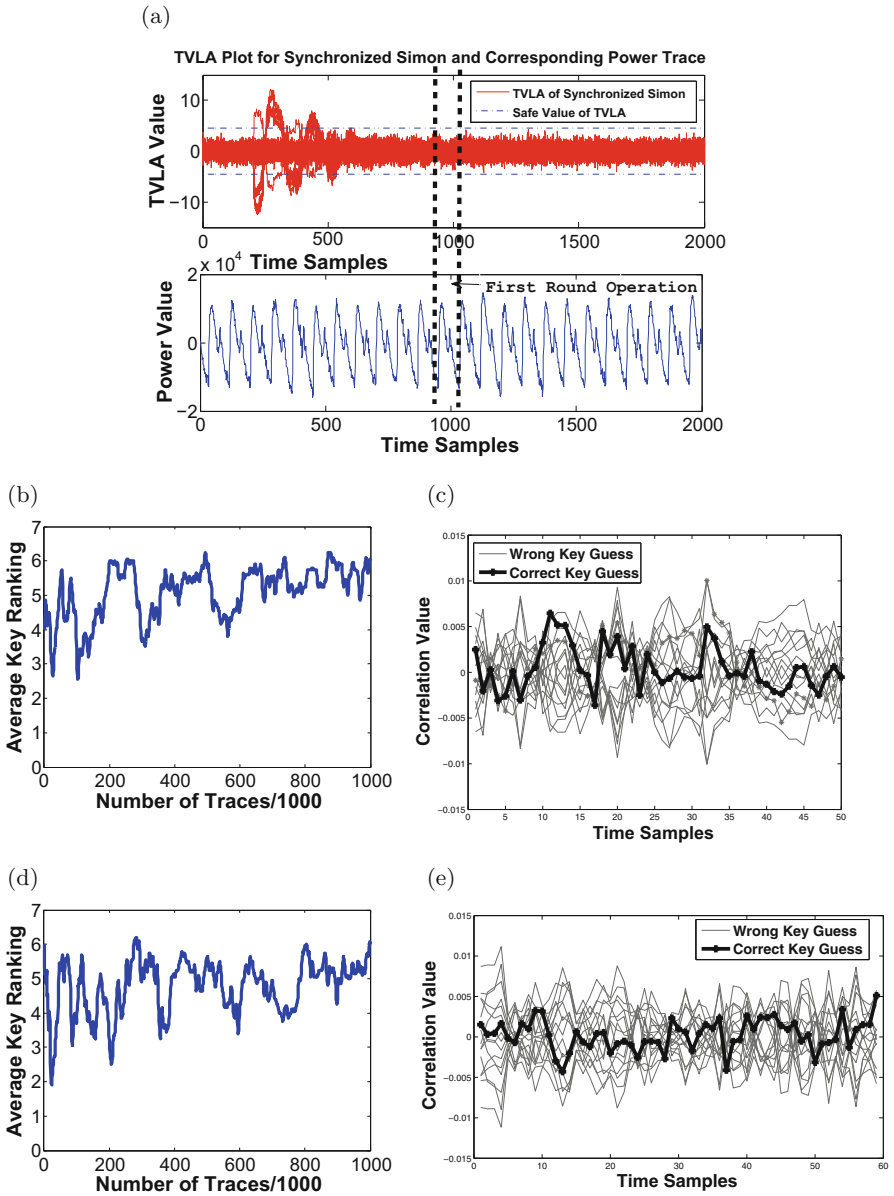

Fig. 6.7 Side-channel analysis of synchronized 2-input LUT SIMON. (**a**) TVLA plot. (**b**) Avg. key rank (TVLA peak). (**c**) Correlation value at TVLA peak. (**d**) Avg. key rank (first round). (**e**) Correlation value at first round

Table 6.1 Summary of side-channel analysis

Design name	TVLA test	Max. TVLA leakage	Avg. key ranking	Remarks
Optimized SIMON	Fails, significant information leakage	18	Key ranking is low, successful attack	Not secure
2-input LUT based SIMON	Fails, but less information leakage	12	Key ranking is high, attack fails	Secure against CPA with HD model
Synchronized 2-input LUT based SIMON	Passes: no leakage at first round	3.5	Key ranking is high, attack fails	Secure

which indicates plaintext loading, but no peak at the first round operation. However, to eliminate any ambiguity, we have performed CPA around both the first round operation and TVLA peak. Result shows that in both cases, side-channel attack is not working and implementation under attack can be considered secure against side-channel attack. Thus in this case, theoretically secure private circuit is translated into practically secure private circuit implementation. The summary of our experimental analysis is shown in Table 6.1.

6.9 Conclusion

This book chapter focuses on the basic concepts of side-channel vulnerability evaluation of cryptographic IPs. In this context we have presented three different metrics: Guessing Entropy, NICV, and TVLA. For each of the metric, we have first described the basic mathematical background, followed by example case studies. We have also highlighted how two different metrics TVLA and NICV are related to each other and how we can modify TVLA formulation to address higher order side-channel analysis. A thorough case study on popular side-channel countermeasure *private circuit* has also been presented where we analyze side-channel security of three different implementation strategies of *private circuits* using TVLA.

Acknowledgements We will like to thank Professor Sylvain Guilley from Telecom Paristech for his valuable inputs on side channel which greatly improved the chapter.

References

1. P. Kocher, J. Jaffe, B. Jun, Differential power analysis, in *Annual International Cryptology Conference* (Springer, Berlin, 1999), pp. 388–397
2. Common Criteria consortium, Application of Attack Potential to Smartcards — v2.7, March (2009)

3. NIST FIPS (Federal Information Processing Standards) publication 140–3, Security requirements for cryptographic modules (Draft, Revised). 09/11 (2009), p. 63
4. F.-X. Standaert, T.G. Malkin, M. Yung, A unified framework for the analysis of side-channel key recovery attacks, in *Annual International Conference on the Theory and Applications of Cryptographic Techniques* (Springer, Berlin, 2009), pp. 443–461
5. S. Mangard, E. Oswald, F.-X. Standaert, One for all; all for one: unifying standard differential power analysis attacks. IET Inf. Secur. **5**(2), 100–110 (2011)
6. S. Bhasin, J.-L. Danger, S. Guilley, Z. Najm, Side-channel leakage and trace compression using normalized inter-class variance, in *Proceedings of the Third Workshop on Hardware and Architectural Support for Security and Privacy* (ACM, New York, 2014), p. 7
7. B.J.G. Goodwill, J. Jaffe, P. Rohatgi, et al, A testing methodology for side-channel resistance validation. NIST Non-invasive Attack Testing Workshop (2011)
8. E. Prouff, M. Rivain, R. Bevan, Statistical analysis of second order differential power analysis. IEEE Trans. Comput. **58**(6), 799–811 (2009)
9. C. Whitnall, E. Oswald, F.-X. Standaert, The myth of generic DPA and the magic of learning, in *Cryptographers Track at the RSA Conference* (Springer, Berlin, 2014), pp. 183–205
10. M. Renauld, F.-X. Standaert, N. Veyrat-Charvillon, D. Kamel, D. Flandre, A formal study of power variability issues and side-channel attacks for nanoscale devices, in *Annual International Conference on the Theory and Applications of Cryptographic Techniques* (Springer, Berlin, 2011), pp. 109–128
11. S. Chari, J.R Rao, P. Rohatgi, Template attacks. In *International Workshop on Cryptographic Hardware and Embedded Systems* (Springer, Berlin, 2002), pp. 13–28
12. B. Gierlichs, K. Lemke-Rust, C. Paar, Templates vs. stochastic methods, in *International Workshop on Cryptographic Hardware and Embedded Systems* (Springer, Berlin, 2006), pp. 15–29
13. J. Doget, E. Prouff, M. Rivain, F.-X. Standaert, Univariate side channel attacks and leakage modeling. J. Cryptogr. Eng. **1**(2), 123–144 (2011)
14. T. Schneider, A. Moradi, Leakage assessment methodology. in *International Workshop on Cryptographic Hardware and Embedded Systems* (Springer, Berlin, 2015), pp. 495–513
15. A.A. Ding, C. Chen, T. Eisenbarth, Simpler, faster, and more robust t-test based leakage detection, 2016
16. S. Sonar, D.B. Roy, R.S. Chakraborty, D. Mukhopadhyay. Side-channel watchdog: run-time evaluation of side-channel vulnerability in FPGA-based crypto-systems. IACR Cryptology ePrint Archive, 2016:182 (2016)
17. Y. Ishai, A. Sahai, D. Wagner, Private circuits: securing hardware against probing attacks, in *In Proceedings of CRYPTO 2003* (Springer, Berlin, 2003). pp. 463–481
18. M. Rivain, E. Prouff, Provably secure higher-order masking of AES. in *12th International Workshop Cryptographic Hardware and Embedded Systems, CHES 2010*, ed. by S. Mangard, F.-X. Standaert , Santa Barbara, CA, Aug 17–20, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6225 (Springer, Berlin, 2010), pp. 413–427
19. R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, L. Wingers, The SIMON and SPECK families of lightweight block ciphers. Cryptology ePrint Archive, Report 2013/404 (2013)
20. J. Park, A. Tyagi, *t*-Private logic synthesis on FPGAs. in *2012 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pp. 63–68, June 2012
21. D.B. Roy, S. Bhasin, S. Guilley, J. Danger, D. Mukhopadhyay, From theory to practice of private circuit: a cautionary note, in *33rd IEEE International Conference on Computer Design, ICCD 2015*, pp. 296–303. New York City, NY, Oct 18–21, 2015. [21]
22. É. Brier, C. Clavier, F. Olivier, Correlation power analysis with a leakage model, in *Cryptographic Hardware and Embedded Systems*. Lecture Notes in Computer Science, vol. 3156 (Springer, Berlin, 2004), pp. 16–29, Aug 11–13, Cambridge, MA
23. A. Moradi, Oliver Mischke. Glitch-free implementation of masking in modern FPGAs, in *HOST* (IEEE Computer Society, New York, 2012), pp. 89–95, June 2–3 2012. Moscone Center, San Francisco, CA. doi:10.1109/HST.2012.6224326

24. G. Goodwill, B. Jun, J. Jaffe, P. Rohatgi, A testing methodology for side-channel resistance validation. NIST Non-Invasive Attack Testing Workshop, Sept 2011
25. J. Cooper, G. Goodwill, J. Jaffe, G. Kenworthy, P. Rohatgi. Test vector leakage assessment (TVLA) methodology in practice, in *International Cryptographic Module Conference*, Holiday Inn Gaithersburg, MD, Sept 24–26, 2013