Comparison of Side Channel Analysis Measurement Setups

Alvin Cai Kunming

Master thesis

Department of Mathematics and Computer Science
Technische Universiteit Eindhoven

Supervisors:
Rafael Boix Carpi (Riscure BV)
Lejla Batina (Radboud University)
Boris Skoric (Technische Universiteit Eindhoven)

2015

Abstract

The fair evaluation of side channel attacks and countermeasures is currently an active research topic with several state of the art evaluation metrics proposed recently. However, these metrics have mostly been designed to compare Side Channel Analysis (SCA) distinguishers or countermeasures but not measurement setups. In this thesis, we assess the suitability of existing metrics for evaluating SCA measurement setups. Through practical experiments, we determine that the Signal to Noise Ratio (SNR) metric is the most suitable. To investigate the effect of the measurement setup on the practicality of an SCA attack, we apply the SNR metric on setups at different ends of the price spectrum: the high-end professional Inspector SCA tool and the low-budget, open source ChipWhisperer setup. We conclude that for the low budget setup, practical limitations (such as the sampling size buffer, sampling speed or bandwidth), rather than the noise introduced by the measurement setup are the main factors limiting SCA success.

Acknowledgements

Many people have made this thesis possible. At the top of the list are my supervisors Rafael Boix Carpi and Lejla Batina whose help and guidance were instrumental in shaping this thesis. I would also like to thank Boris Skoric who greatly improved the delivery of this thesis with his careful reading and insightful comments.

Furthermore, I would like to thank Riscure B.V. for providing the hardware and software used in this thesis and all the Riscure employees for being ever so ready to share your knowledge. I have learnt a lot from our wide ranging discussions on SCA, Verilog, soldering (the list goes on). I am also very thankful to Colin O'Flynn, the developer and founder of ChipWhisperer for his technical support, pre-production prototypes of ChipWhisperer-Lite and thesis comments. In addition, many thanks to Kostas Papagiannopoulos for our discussions on the mutual information metric.

I'd also like to thank the assessment committee made up of Boris Skoric, Berry Schoenmakers, Lejla Batina and Rafael Boix Carpi. I am extremely grateful for the time you have all invested to read this thesis and attend my final presentation.

I would like to thank my sponsor from Singapore, the Defence Science & Technology Agency (DSTA), for the opportunity to further my education at the Technische Universiteit Eindhoven and the Kerckhoffs Institute for Computer Security. Finally, I am grateful to my family for their love, patience and support.

Contents

List of Figures

List of Tables

List of abbreviations

ADC	Analog to Digital Conversion
AES	Advanced Encryption Standard
APDU	Application Protocol Data Unit
AVR	Alf and Vegard's RISC processor
CDF	Cumulative Distribution Function
CPA	Correlation Power Analysis
DCM	Digital Clock Management
DES	Data Encryption Standard
DOM	Distance of Means
DPA	Differential Power Analysis
EM	Electromagnetic
EN	Experiment Number
FIFO	First In, First Out
HDL	Hardware Description Language
IO	Input/Output
MIA	Mutual Information Analysis
PC	Personal Computer
PCB	Printed Circuit Board
RDM	Relative Distinguishing Margin
S	Samples
S-Box	Substitution Box
SCA	Side Channel Analysis
SNR	Signal to Noise Ratio
SPA	Simple Power Analysis
USB	Universal Serial Bus

Chapter 1

Introduction

1.1 Cryptography

Many modern-day electronic devices use cryptography to provide confidentiality, integrity or authenticity. An example of a cryptographic service is encryption which can be viewed as a mathematical function that takes as input a cryptographic key and plaintext message and outputs a ciphertext message. The security basis of an encryption algorithm is in the cryptographic key; without the cryptographic key, an adversary is allowed to know the encryption algorithm and ciphertext but still not learn any information about the plaintext message. This enables the confidentiality of messages exchanged over insecure channels.

Other than encryption, other types of cryptographic services such as digital signatures and hashing also exist. There are many practical applications for these services including bank cards that are used at ATMs or payment terminals, digital signatures on software upgrades, secure internet messaging on smart phones, smart meters and sensor nodes.

Regardless of the type of cryptographic service or application, a common goal for attackers is the retrieval of the cryptographic key. This can be achieved through exploiting weaknesses in the cryptographic algorithm, or logical and physical attacks on the implementation such as buffer overflows and micro-probing respectively.

1.2 Side Channel Analysis

In the mid 1990s, it was discovered that the time taken [Koc96] and power consumed [KJJ99] during a cryptographic operation in a physical device reveals some additional information that simplifies cryptographic key retrieval. The usage of these physical observations (whether it be timing, power, optical or

electromagnetic emissions) to retrieve the cryptographic key, is known as side channel analysis and is a cryptanalysis technique.

Such attacks have proven to be very effective, can be conducted easily and are hard to detect as they are non-invasive. Indeed, these attacks had been secretly utilised in classified military programs since the mid 1900s. A well known example of this is the TEMPEST program in the USA which covers methods to both exploit and protect against the risk of compromising emissions [NSA14].

Unlike classical cryptanalysis that uses a black box model of the cryptographic implementation, side channel analysis utilises a grey box model. The black box model is so named because it assumes that the adversary cannot see the internal variables of the implementation but only the plaintext input and ciphertext output. Correspondingly, in the grey-box model *some* of the internal variables are revealed.

The leakage of internal variables can arise due to instruction dependency or data dependency when processing side channel information. Considering power consumption as the side channel, instruction dependency occurs because different sequences of CPU instructions have different power consumption features. By performing simple power analysis on a single power measurement, it is possible to observe the different rounds of a cryptographic operation and also conditional jumps based on intermediate key bits [KJJ99]. For the same series of instructions, different data values being processed also result in power consumption variations, albeit much smaller. It is typically assumed that processing a bit value zero uses a different amount of power than processing a bit value one. This is known as data dependency and can be exploited with differential power analysis.

Side channel attacks are continuously improving and so too are countermeasures. These countermeasures can be classified into hardware, software and application level countermeasures [Wit02]. Hardware countermeasures typically involve shielding or noise introduction to reduce the signal to noise ratio. Software countermeasures include constant time implementations (to eliminate branching on sensitive data), blinding of intermediate values and again noise introduction through random time delays or ordering. Finally, application level countermeasures reduce the number of measurements an attacker is able to or allowed to make e.g. through PIN verification, or limit visibility and control of the inputs and outputs.

Today, some high security products, e.g. smartcards, implement a combination of some of these countermeasures and are highly resistant to side channel analysis. Nevertheless, many cryptographic devices still remain vulnerable chiefly due to the cost of implementing countermeasures and the varying ma-

turity levels of security designers. Moreover, even if a product is secure today, technology advancement might mean that a new attack might render it vulnerable tomorrow.

1.3 Fair Evaluation Metrics of SCA Attacks and Countermeasures

Comparing among SCA attacks can be complicated due to the many factors involved such as the accuracy of the chosen leakage model. Nevertheless, it is beneficial as it enables attackers to optimise their attack strategy to bring about practical benefits such as key recovery with a smaller number of leakages, or lesser computation. Correspondingly, comparing between SCA countermeasures helps security designers to optimise their defence strategy.

1.4 Contribution of this Thesis

Most of the existing SCA metrics are used to compare distinguishers or countermeasures and not measurement setups. In this thesis, we review the existing metrics and assess through practical experiments which of them is the most suitable for evaluating SCA measurement setups. Measurement setups are interesting as they influence the practicality of an SCA attack. For example, the noise introduced by a measurement setup ultimately impacts the number of leakages required for a successful attack. The cost price, technical expertise or engineering time required for a particular measurement setup also determines its accessibility to attackers.

A metric for evaluating SCA measurement setups is hence useful as it will help the attacker to optimise his resources by choosing the most cost effective setup. It also enables SCA tool developers to quantifiably assess the effectiveness of their products so that they can be improved. Through practical experiments, we determined that the SNR metric is the most suitable metric for measurement setups.

Next, we applied the SNR metric on setups at different ends of the price spectrum. We used Inspector SCA to represent a high end tool that is only accessible to large and professional organisations and the ChipWhisperer to represent the low-budget tool accessible to the masses. We concluded that for the low budget setup, practical limitations (such as the sampling size buffer, sampling speed or bandwidth) and not the measurement setup noise are the main factors limiting SCA success.

1.5 Thesis structure

The remaining thesis is structured as follows:

Chapter 2 describes the model used in side channel analysis and provides an overview of selected side channel attacks. This provides the necessary background for the discussion of SCA metrics in Chapter 4.

Chapter 3 describes the measurement setups that are used in the experiments.

Chapter 4 first provides an overview of the major SCA metrics that have been proposed in academia and then describes the reason why signal to noise ratio was determined to be the most suitable metric for measurement setups.

Chapter 5 describes two theoretical methods to calculate the success rate that are more efficient than the empirical method. These findings further encourages the use of SNR as a metric.

Chapter 6 describes the different experiments, presents the results and provides an analysis of the results.

Chapter 7 concludes this thesis with a summary and suggestions for future research.

Chapter 2

Background

2.1 Side Channel Model

In this section, we describe the side channel model shown in Figure 2.1 and mathematical notation that will be used consistently throughout this thesis. Our side channel model is based on the model first proposed in [MR04] using a similar notation to [SMY09].

Here, a physical device (e.g. smartcard) carries out a cryptographic operation (e.g. Advanced Encryption Standard (AES)) that depends on a secret key, K. The physical device leaks some information about its internal operating state through side channels such as power. The output of this side channel is a physical observable $O(W(T))$ that is some function of the word W being processed by the physical device at that specific time instance T.

The adversary then tries to measure the physical observable O using a measurement setup to obtain the leakage L. As explained in Section 3, the power consumption of the circuit O might not be directly measurable and we thus need to derive it through a proxy, e.g. by measuring the voltage across a resistor connected in series. The measurement process also introduces noise and furthermore, might involve some hardware signal processing e.g. filtering and amplification. Finally, the adversary attempts to recover the key using a set of SCA algorithms that are described in Section 2.2.

Above, the upper case letters K, W, O, T and L represent random variables. These random variables can take on some concrete value which is denoted by its lowercase counterpart k, w, o, t or l respectively.

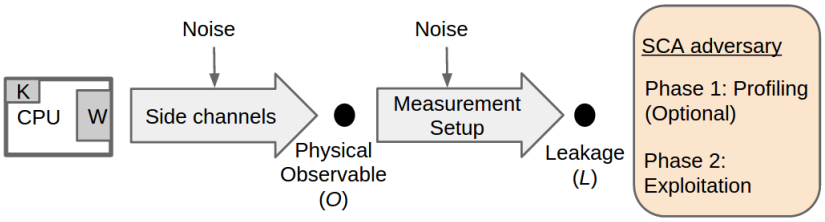

Figure 2.1: SCA Model.

2.2 Differential Power Analysis (DPA)

In this thesis, we focus only on DPA and not Simple Power Analysis (SPA). The broad principles of DPA are first described in Section 2.2.1. Subsequently, we also describe several non-profiled and profiled DPA techniques which are later relevant to the comparison of metrics in Section 4.

Non-profiled attacks such as Distance of Means (DOM) and Correlation Power Analysis (CPA) consist only of an attack phase. Profiled attacks consist of an additional profiling phase that builds a model of the target device with the aim of reducing the number of measurements required later in the attack phase. The template attack is one example of a profiling attack.

2.2.1 Preliminaries

As mentioned in Section 1.2, DPA utilises the physical phenomena that different data values being processed in a physical device result in different power leakages. Power leakages are sometimes also referred to as power traces. In Figure 2.2 the power leakage of a smartcard Data Encryption Standard (DES) Substitution Box (S-box) computation is plotted as a function of time for different S-box output Hamming weights. In this smartcard, the height of the power leakage increases proportionally with the Hamming weight of the S-Box output from 0 to 55 ns.

Each leakage trace displayed in Figure 2.2 was obtained by averaging 500 leakage signals to reduce the noise using the same method proposed in [MDS99]. This large number of leakages is required because the leakage signal from data dependencies is usually small compared to the noise. The side channel adversary hence needs to obtain a sufficiently large number (q) of measurements ($M = l_1, ..., l_q$) of the leakage (L) to counter these noise effects. Each leakage trace (l_i) is made up of r samples measured at fixed time intervals. For non-profiled DPA, we also need either known inputs or outputs ($b_1, ..., b_q$) and the measurements should all be encrypted under a single key (k).

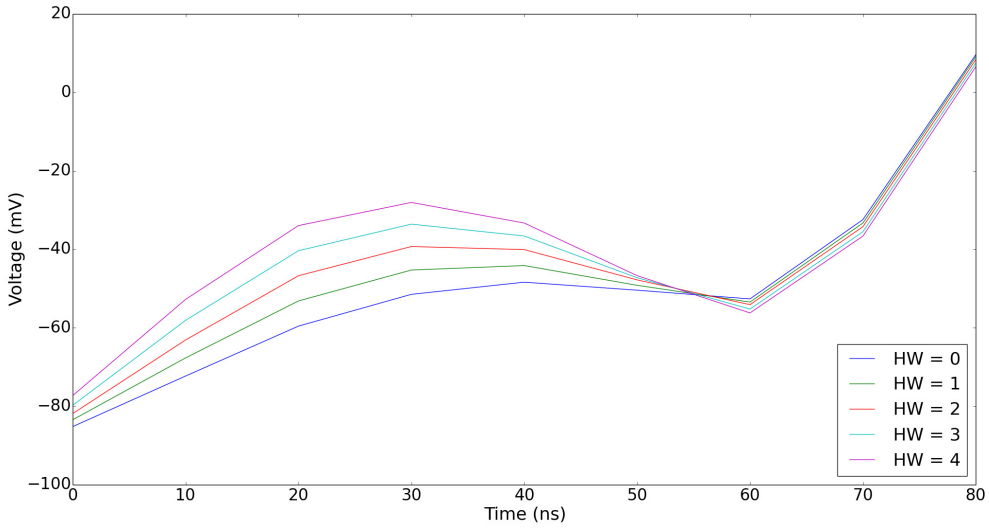

Figure 2.2: Power Trace for different Hamming weight values of the S-box output are plotted as a function of time.

Divide and Conquer

In symmetric cryptographic algorithms, key sizes are usually large enough (e.g. 128 bits) to prevent brute force attacks. However, for reasons such as computational efficiency, many cryptographic algorithms split this large key into many smaller chunks (subkeys) that are processed at different time intervals. We denote the subkey as S and the number of subkey bits as n, so that $s \in \{0,1\}^n$. For AES, $n = 8$ while for DES, $n = 6$. An SCA attack usually employs a divide and conquer strategy to independently retrieve these subkeys. Once all subkeys are known, the correct key can be easily reconstructed using the key schedule.

The S-box of symmetric algorithms is a common target of this strategy due to its non-linear properties and relation to the subkey. In AES, each S-box operation (ϕ), takes an input (X) which is the exclusive-OR of the 8 subkey bits (S) with 8-bits of the S-Box input state (P). P is a function of the plaintext input. We denote the S-box output as Y where $y = \phi(p, s)$.

Leakage Model

In many SCA attacks, we do not use the word W being processed by the physical device directly in the SCA attack. Instead, we use a physical leakage model to transform W to the predicted leakage V which has a more direct relation to the power being consumed. In the situation described for Figure 2.2, the leakage model is the Hamming weight of the S-box output and thus, $V = \mathrm{HW}(\phi(p, s))$ where HW is the Hamming weight function. When using the Hamming weight

model, we make the assumption that processing a bit values of '1' requires different power than processing a bit value of '0'.

The Hamming distance model is another popular leakage model that is bsed on the assumption that the dynamic power consumption of a circuit depends on the number of $0 \rightarrow 1$ and $1 \rightarrow 0$ transitions [MOP08]. The Hamming distance is calculated as the Hamming weight of the old data value (e.g. of a data bus or a register) exclusive-ORed with its new value. The correct model depends on the hardware realisation of the physical device and the accuracy of the chosen leakage model strongly impacts the efficacy of the SCA attack.

Attack Output

With the appropriate leakage model, leakage traces and known inputs or outputs, the SCA attacker can compute the predicted leakage, e.g. $(V' = \mathrm{HW}(\phi(p, s')))$, for all candidate subkeys s'. With a statistical test, the attacker then compares the predicted leakage for all candidate subkeys against the measured leakages to try to determine which subkey is most likely to be correct. In non-profiled attacks, this statistical test is usually termed a 'Distinguisher'. For example, the distinguisher described in Section 2.2.2 and Section 2.2.3 are the distance of means and Pearson's correlation coefficient respectievly.

We use the definition from [SMY09] that the output from the SCA attack is a guess vector $\mathbf{g} = [g_1, g_2, ..., g_{2^n}]$. Here, the different subkey candidates are sorted according to their likelihood of being correct, in descending order; The most likely subkey candidate is g_1 and the least likely subkey candidate is g_{2^n}.

2.2.2 Distance of Means (DOM)

In the first published paper on DPA in 1998 [KJJ99], Kocher et al proposed to use the '1-bit distance of means' as an SCA distinguisher. In this method, one bit (e.g. the Least Significant Bit (LSB)) of the intermediate value Y is used as the select function for each subkey guess. The attacker partitions the traces M into two sets M_0 and M_1 depending on whether the select function returns '1' or '0'. This is effectively a 1-bit Hamming weight leakage model.

The difference between the average of both sets, $E(M_0) - E(M_1)$ would result in spikes only for the correct key guess s^*. The reasoning behind this deduction was that if the key guess was wrong, the two sets M_0 and M_1 would be uncorrelated and the difference between the averages should approach zero as the number of measurements q increase. However, when the key guess was correct, the signals become correlated and if we plot the average of M_0 against the average of M_1, we should observe a pattern similar to Figure 2.2 but with just 2 waveforms.

2.2.3 Pearson's Correlation Coefficient

The CPA attack in [BCO04] makes use of Pearson's correlation coefficient from classical statistics to detect linear correlation between the predicted leakage (V') calculated for every candidate subkey, against the set of leakage traces M. We develop the necessary notation and detail the process of calculating the Pearson's correlation coefficient for a single DES S-box in the following steps:

1. We obtain a measurement set of q leakages $M = [l_1, l_2, ..., l_q]$. Each leakage is made up of r samples so that $l_i = [l_{i,1}, l_{i,2}, ..., l_{i,r}]$. We denote the measurement set that is made up of only the sample j from all leakages as $M(j) = [l_{1,j}, l_{2,j}, ..., l_{q,j}]$.

2. We also obtain the plaintext corresponding to each leakage. With the plaintext, we can calculate the input state p of our target S-box. The set of input states that corresponds to the leakages in the measurement set is $[p_1, p_2, ..., p_q]$.

3. For each leakage in the measurement set M, we can calculate the predicted leakage for a given subkey candidate s'. We denote the set of predicted leakages for the subkey candidate s' as $\mathbf{v}(\mathbf{s}') = [v_1(s'), v_2(s'), ..., v_q(s')]$. Assuming a Hamming weight leakage model, $v_i(s') = \mathrm{HW}(\phi(p_i, s'))$.

4. The Pearson's correlation coefficient between $M(j)$ and $\mathbf{v}(\mathbf{s}')$ is calculated with Equation (2.1). It is defined as the covariance between $M(j)$ and $\mathbf{v}(\mathbf{s}')$, normalised by the product of their standard deviation. This value is the Pearson's correlation coefficient for a single sample. Typically, the Pearson's correlation coefficient is calculated for all samples in the leakage so that we can obtain plots of the Pearson's correlation coefficient against time as shown in Figure 2.3. The correlation coefficient in this figure was calculated with the same leakages that were used to generate Figure 2.2.

5. Finally, Step (4) is repeated for the different candidate subkeys. Similar to the distance of means, the correct subkey guess should provide a higher correlation peak value compared to incorrect subkey guesses. As such, we rank the candidate subkey that returns the highest absolute value of the Pearson's correlation coefficient as the most likely subkey. As shown in Figure 2.3, the Pearson's correlation coefficent approaches 1 for the correct subkey (in this case $s^* = 0\text{x}1\text{f}$) and is much lower for a wrong subkey ($s' = 0\text{x}22$).

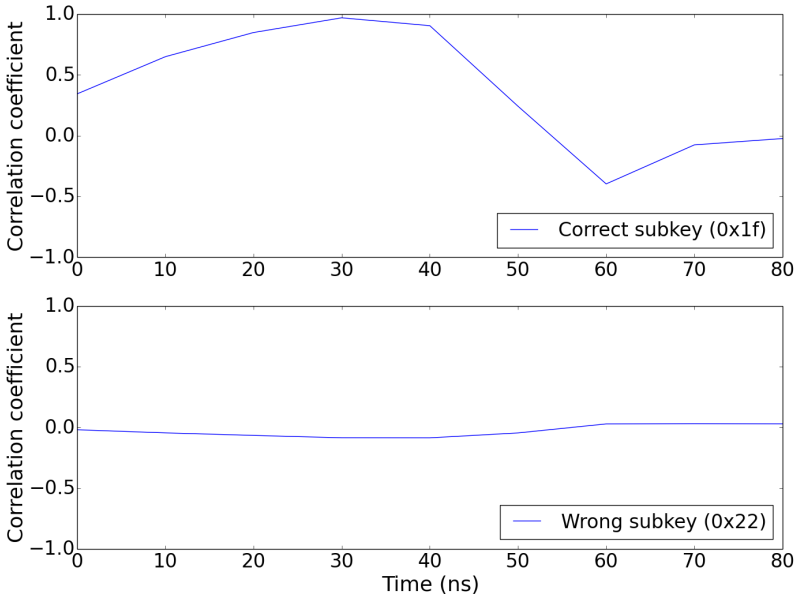

Figure 2.3: Pearson's Correlation Coefficient for the correct subkey and a wrong subkey.

$$\begin{aligned}\rho(\mathbf{v}(\mathbf{s}'), M(j)) &= \frac{\operatorname{cov}(\mathbf{v}(\mathbf{s}'), M(j))}{\sigma_{\mathbf{v}(\mathbf{s}')}\sigma_{M(j)}} \\ &= \frac{\mathrm{E}[(\mathbf{v}(\mathbf{s}') - \mathrm{E}[\mathbf{v}(\mathbf{s}')])(M(j) - \mathrm{E}[M(j)])]}{\sqrt{\mathrm{E}[(\mathbf{v}(\mathbf{s}') - \mathrm{E}[\mathbf{v}(\mathbf{s}')])^2]} \cdot \sqrt{\mathrm{E}[(M(j) - \mathrm{E}[M(j)])^2]}}.\end{aligned} \tag{2.1}$$

The formula for covariance is

$$\operatorname{cov}(X, Y) = \mathrm{E}[(X - \mathrm{E}[X])(Y - \mathrm{E}[Y])], \tag{2.2}$$

and that for standard deviation is

$$\sigma(X) = \sqrt{\mathrm{E}[(X - \mathrm{E}[X])^2}. \tag{2.3}$$

2.2.4 Template Attacks

The template attack was introduced in [CRR03] as the strongest form of side channel attack possible in the information theoretic sense. Interestingly, while other attacks aim to eliminate noise, the template attack uses it as an additional source of information which optimises its effectiveness.

The principle behind this attack is that for a given operation (i.e. the same CPU instruction and data being processed), the measured leakage is a combi-

nation of an intrinsic signal generated by that operation and some noise which is both intrinsically generated and also from the environment. For a symmetric cipher such as DES or AES, we typically assume 2^n different operations each associated with the 2^n different S-box inputs so that $x \in \{0,1\}^n$. The signal component μ_i associated with S-box input $(x = i)$ is always the same for repeated invocations of the same operation and is estimated by taking the mean of all leakages with that same S-box input such that $\mu_i = \mathrm{E}_j[l_j], \forall x_j = i$.

The noise is best modelled as a multivariate random variable drawn from a noise probability distribution. The noise vector u_j for a leakage l_j with S-box input $(x_j = i)$ is calculated by subtracting μ_i from that leakage so that $u_j = l_j - \mu_i$. We group the set of noise vectors for S-box input $x = i$ together to form $\mathbf{u_i} = \{u_j : \forall j \text{ such that } x_j = i\}$. Similar to the leakages l, the noise vector u_j is made up of r samples such that $u_j = [u_{j,1}, u_{j,2}, ..., u_{j,r}]$.

We denote the set of noise vectors for S-box input $x = i$,with only the single sample point h, as $\mathbf{u_i}(\mathbf{h}) = \{u_{j,h} : \forall j \text{ such that } x_j = i\}$. The (r x r) noise covariance matrix C_i can be computed from the set of noise vectors so that $C_i[h_1][h_2] = \mathrm{cov}(\mathbf{u_i}(\mathbf{h_1}), \mathbf{u_i}(\mathbf{h_2}))$.

The template is made up of the intrinsic signal μ_x and noise covariance matrix C_x for each S-box input. In practice, each leakage contains a large number of samples r that renders it infeasible to compute the 2^n, (r x r) noise covariance matrices. To reduce the computational and storage requirements, a reduced set of r' more 'interesting' sample points from each leakage are used. The method of choosing these r' points has been described by the original authors in [CRR03] and improved upon in [RO05] but will not be described here as it is not relevant to this thesis. This is because in Chapter 4, we use the Template attack's profiling method for only a *single* sample point when computing the mutual information metric.

In the attack phase, a maximum likelihood approach is used. For a given leakage trace l_a from the attack measurement set, the attacker calculates the probability of observing the noise vector for each of the hypothetical operation (which we have defined in this example to be the different S-box inputs). For the hypothetical S-box input $x = i$, the attacker can calculate the noise vector u_a from the leakage trace l_a by subtracting the intrinsic signal corresponding to that S-box input so that $(u_a|x = i) = l_a - \mu_i$. The probability of observing the noise vector $(u_a|x = i)$ can then be calculated using

$$\Pr[u_a|x = i] = \frac{1}{\sqrt{(2\pi)^{r'} \cdot |C_i|}} \exp(-\frac{1}{2} \cdot (u_a|x = i)^T \cdot (C_i)^{-1} \cdot (u_a|x = i)). \quad (2.4)$$

Finally, the most likely operation is the one with a noise vector which has

the highest probability of being observed $x^* = \text{argmax}_i(\Pr[u_a|x = i])$. From x^* it is possible to directly derive the most likely subkey s^* as $s^* = x^* \oplus p_a$

2.2.5 Mutual Information Analysis (MIA)

The mutual information between the hypothetical intermediates and the leakage trace l was introduced as an information theoretic SCA attack in [GBTP08]. The benefits of this attack were that it did not require knowledge of the leakage model and could detect any kind of statistical dependencies, not just linear dependencies.

The mutual information between leakage trace L and predicted leakages V is defined as

$$I(L; V(s)) = H(L) - H(L|V(s)). \tag{2.5}$$

Unfortunately, we cannot know the true entropy or conditional entropy of the leakage trace and need to model the probability distributions of $P(L|V)$ and $P(L)$. It is explained in [VCS09] that estimating the probability distribution can be done with both parametric or non-parametric methods. Non-parametric methods do not require the attacker to make any assumptions about the leakages e.g. that it follows a gaussian distribution. This is one of the tenants of MIA.

The histogram method is an example of a non-parametric method. To simplify the description, we consider leakages with only a single sample point (instead of r samples). To estimate the probability of the leakages L using the histogram method, each leakage is partitioned into a bin according to its value. Each bin contains leakages that fall within a certain predetermined range of values and the range of all bins have an equal width. Finally, the probability of that sample taking a certain value can be approximated by dividing the number of samples that fall within a bin by the total number of samples in all bins.

To estimate the probability of $L|V(s)$, leakages with the same values of $V(s)$ are grouped together. The probability of $L|(V(s) = i)$, is estimated for that group of leakages using the same method described for L. The calculation of conditional entropy also requires the probability distribution of $V(s)$. $\Pr[V(s) = i]$ can be empirically estimated by dividing the number of occurrences of $V(s) = i$ by the total number of traces q. In practice, it is typical to assume a random distribution of plaintext inputs and estimate $\Pr[V(s) = i] = \frac{\binom{n}{i}}{\sum_k \binom{n}{k}}$.

Parametric methods can also be useful due to their efficiency. An example of a parametric method is to assume a gaussian distribution of leakages and estimate $\Pr(L|(V(s))$ using the template attack's profiling phase.

Finally, like in all other attacks, a maximum likelihood method is used and the best guess is the subkey which maximises $I(L; V(s))$ so that $s^* = \text{argmax}_s(I(L; V(s))$. The underlying rationale is because the mutual informa-

tion of two independent variables is zero (i.e. for wrong key guesses), but non-zero when they are dependent.

Chapter 3

Power Measurement Setups

3.1 Typical Power Measurement Setups

A measurement setup to conduct practical power analysis usually consists of several components that interact with each other in a fixed sequence [MOP08]. These components are:

- The **cryptographic device** under attack. This device usually interfaces with a Personal Computer (PC) controlled by the SCA attacker which issues commands to it and receives its response. A typical command from the PC would be to 'encrypt a plaintext block, with a specified cryptographic key'. The cryptographic device would process the command and respond with the ciphertext. It is also commonly known as the 'target', TOE (Target of Evaluation) or DUT (Device Under Test).
- The cryptographic device usually requires some form of **power supply and clock generator** which may be internal or external to the cryptographic device depending on its implementation.
- A **Power Measurement Circuit** is needed to extract a voltage signal that can be measured by a digital sampling oscilloscope, which conveys as much information as possible about the power consumed during the cryptographic operation. In the simplest scenario, a resistor with low resistance is placed between the power supply and the cryptographic device. The voltage across this resistor is proportional to the current drawn by the device and also to its power consumption if the voltage of the power supply is constant. Measurement circuits may also be contactless and instead measure electric or magnetic fields emitted by the cryptographic device.

- The voltage signal from the power measurement circuit is acquired by a **Digital Sampling Oscilloscope**. The oscilloscope is controlled by the PC and transfers the voltage measurements to the PC for storage. The oscilloscope usually requires an analog trigger signal to start the acquisition.
- The Personal Computer controls the whole measurement setup and stores the measured power traces for further processing (e.g. CPA attacks).

3.2 Quality of a Measurement Setup

The quality of a measurement setup is described in [MOP08] as the amount of noise that is present in the measured power traces. Considering the noise introduced by the cryptographic device as a given, and constant across different measurement setups, let's consider only the noise introduced by the rest of the measurement circuit. According to [MOP08], the following noise sources lead to noise in the power traces:

- **Noise of the power supplied** to the cryptographic device.
- **Noise of the clock generator**. Furthermore, an unstable clock frequency will also lead to misalignment across different traces which reduces the signal to noise ratio.
- **Conducted emissions** from components directly connected to the cryptographic device e.g. the communication bus to the PC.
- **Radiated emissions** from electrical components which are not directly connected.
- **Quantization Noise** caused by the round-off error during analog to digital conversion by the oscilloscope.

3.3 Cost of a Measurement Setup

The cost of a setup is derived not only in monetary terms but also in other intangibles such as development time and knowledge. Specialized SCA setups such as Riscure's Inspector SCA [BV15b] and Cryptography Research's DPA Workstation [Res15] consist of a tightly integrated package of hardware and software that provides high performance, reproducible results and a smooth user interface. Such setups cost more in monetary terms but save the user significant engineering cost and are typically used by security evaluation laboratories

or government organizations. We also have more "home brew" setups that involve lower hardware cost but higher hardware/software development time and electronics knowledge which might be the case for setups used in academic laboratories.

The ChipWhisperer, an open source and low cost SCA platform that was released in 2013, potentially reduces the entrance cost for SCA in all dimensions (price, time and knowledge). In this thesis, we use Riscure's Inspector to represent the high-end measurement setup and ChipWhisperer to represent the low-budget measurement setup.

3.4 ChipWhisperer

ChipWhisperer [OC14] was designed by Colin O'Flynn to be an open source SCA platform that contains all the necessary components required for SCA. Two versions of the measurement hardware are currently available and these are the 'ChipWhisperer Capture Rev2' and the 'ChipWhisperer-Lite' which are described in Section 3.4.1 and 3.4.2 respectively. The SCA software that runs on the attacker's PC is described in Section 3.4.3. Finally, there is also a set of reference cryptographic devices to perform attacks against, mounted on the 'Multi Target Victim Board' and described in Section 3.6.1. The full setup which contains the attacker's PC, Multi Target Victim Board and ChipWhisperer Capture Rev2 hardware is shown in Figure 3.1.

3.4.1 ChipWhisperer Capture Rev2 Hardware

The main ChipWhisperer version used in this research is the ChipWhisperer Capture Rev2. It is pictured within a metal enclosure in Figure 3.1. Within this enclosure are three Printed Circuit Boards (PCBs); A carrier board mounted with a commercially available ZTEX Field Programmable Gate Array (FPGA) module[1] and an Analog to Digital Converter (ADC) which is named the OpenADC. Together, these three boards make up the ChipWhisperer Capture Rev2 and are shown in Figure 3.2. The ChipWhisperer Capture Rev2 setup costs approximately USD $1500 [new].

Carrier Board

The carrier board, provides the underlying connectivity between the different hardware components and also input/output connections to facilitate communication with external chips (e.g. the multi-target victim board). Other ancillary

[1]This is sold separately by the company ZTEX. More information can be found at http://www.ztex.de/usb-fpga-1/usb-fpga-1.15.e.html

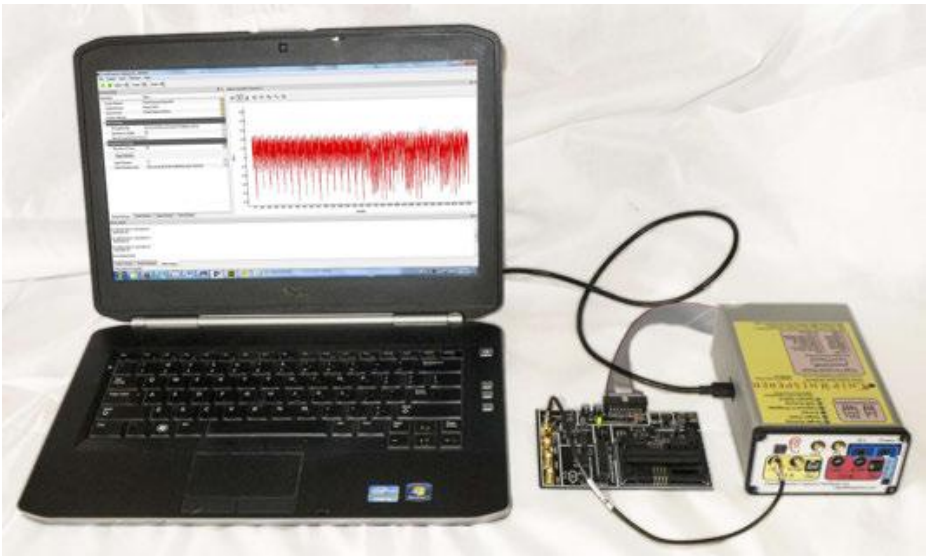

Figure 3.1: The full ChipWhisperer setup containing from left to right, the attacker's PC running the ChipWhisperer capture software, the Multi Target Victim Board and the ChipWhisperer Capture Rev2 hardware. Image from [OC14].

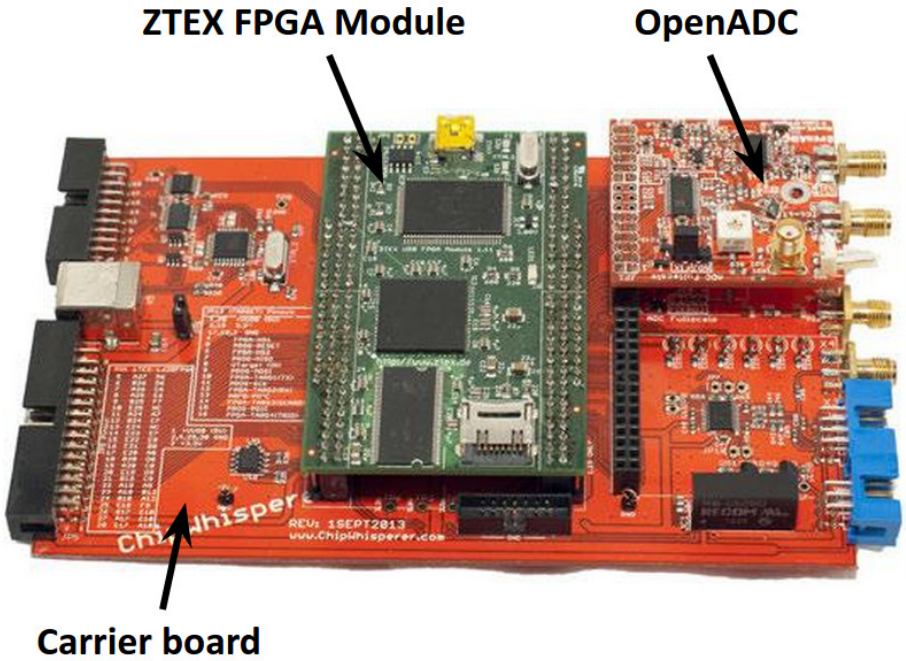

Figure 3.2: The ChipWhisperer Capture Rev2 is made up of the carrier board, a ZTEX FPGA module and an OpenADC. Image from [OC14].

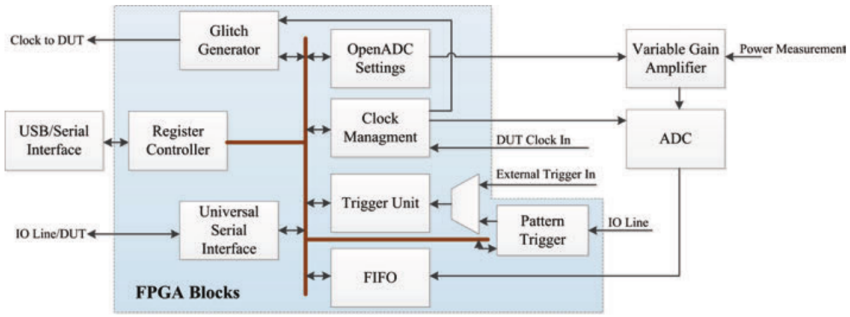

Figure 3.3: The base design of the ChipWhisperer FPGA with several hardware modules connected via an internal communication bus. Image from [OC14].

functions provided by the carrier board are voltage conversions and a programmer for the ATMEL AVR[2] 8-bit microcontroller. The AVR microcontroller itself is located on the Multi-Target Victim Board.

ZTEX FPGA 1.15a module

The ZTEX module consists of a Cypress micro-controller and a Spartan 6 LX45 FPGA. The micro-controller facilitates communication with the ChipWhisperer software that is running onboard a PC. The FPGA contains all the control logic required to operate the OpenADC such as configuration of its gain settings, sampling clock, trigger signal and also implements the memory queue that stores the acquisition samples. The FPGA also contains other logic such as glitch generation and pattern based triggering.

The FPGA is implemented in a modular manner with each function performed by a dedicated hardware module. All hardware modules are connected to an internal communication bus and centred around the register controller. This design is shown in Figure 3.3.

Apart from the register controller which is central to the functioning of the ChipWhisperer hardware, the other hardware modules, e.g. glitch generator and OpenADC settings, operate independently of each other. These modules may be removed if they are not required, or customised by the user as the Verilog source code is open source. New modules may also be added. Furthermore, the physical ZTEX FPGA module can be replaced with a different ZTEX FPGA module with less or more resources as per the user's budget and requirements.

[2]ATMEL Corporation states that the term AVR is not an acronym and does not stand for anything. However it is commonly accepted that it AVR stands for Alf and Vegard's RISC processor.

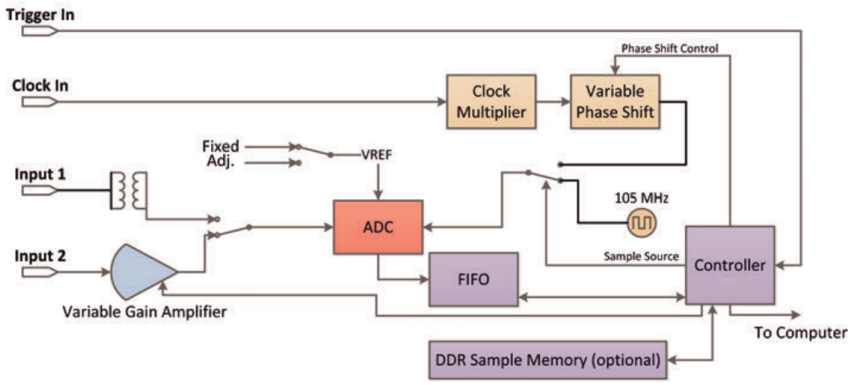

Figure 3.4: The design of the OpenADC. Image from [OC13a].

OpenADC

The OpenADC is ChipWhisperer's equivalent of an oscilloscope. Two distinguishing features of the OpenADC are synchronous sampling as well as input signal amplification of up to 55dB. Synchronous sampling means to synchronise the sampling of the input signal to the clock of the cryptographic device under test. We could for example sample only on the rising edge of the target clock which is hypothetically when the CPU processes the instruction and thus also where interesting leakages occur. In contrast, a typical oscilloscope samples asynchronously based on an internal clock. This clock is continuously running and even though we could time the trigger to the oscilloscope at a fixed and repeatable time instant relative to the cryptographic device, the oscilloscope only starts measuring on the rising edge of *its* own internal clock resulting in some random jitter. The benefit of the synchronous sampling approach is that it can provide comparable performance at a much lower sampling frequency compared to asynchronous sampling [OC13a].

The OpenADC hardware consists of only a variable gain amplifier and an ADC as shown in Figure 3.4 as well as the requisite input and output connections and power supply. The other components in this figure are implemented as digital logic within the ZTEX FPGA. Using the native Digital Clock Management (DCM) IP core of the Xilinx FPGA, the ChipWhisperer is able to adjust the phase of the output clock which is important as the point of interest for power analysis may not lie directly on the rising edge but sometime after the clock edge. The DCM is also able to multiply or divide the frequency of the target's input clock so that output-clock-frequency $= \frac{\text{Multiply}}{\text{Divide}} \cdot$ (input-clock-frequency)

Finally, the OpenADC has a 10-bit resolution, a maximum sampling speed of 105 Million Samples/s (MS/s) and a 50 Ohm input impedance. Its bandwidth

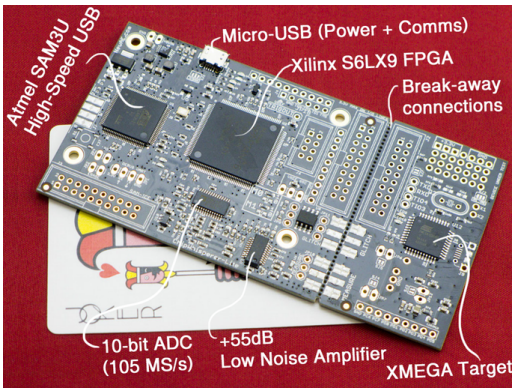

Figure 3.5: The ChipWhisperer-Lite is significantly more compact than the ChipWhisperer Capture Rev2. Image from [kic].

(which is defined throughout this thesis as the highest frequency at which the amplitude has fallen to 0.707 times the maximum amplitude) is 120MHz.

3.4.2 ChipWhisperer-Lite Hardware

Recently, the production of a new ChipWhisperer version called the ChipWhisperer-Lite has been successfully funded via the website Kickstarter [kic] and will be available from August 2015. The ChipWhisperer-Lite contains all the features of the ChipWhisperer Capture Rev2 mentioned in Section 3.4.1 but is more tightly integrated, with all components on a single PCB instead of three separate boards. As such, the ChipWhisperer-Lite is significantly more compact as shown in Figure 3.5 and also costs less at only USD $200.

For this thesis, the ChipWhisperer developer Colin O' Flynn was kind enough to provide us with two pre-production prototypes for our experimental use. Due to time constraints, limited experiments were conducted on the ChipWhisperer-Lite. Another limitation was that we were unable to interface the ChipWhisperer-Lite with the smartcard socket of the multi target victim board and this is explained in Section 3.4.6.

3.4.3 Software

The ChipWhisperer SCA software running on the attacker's PC is implemented entirely in Python and separated into two programs each with a distinct function. The 'Capture' program serves to control the ChipWhisperer hardware to capture the power traces and save them to the local file system of the PC. The 'Analysis' program runs SCA attack algorithms such as profiling and CPA on the stored traces.

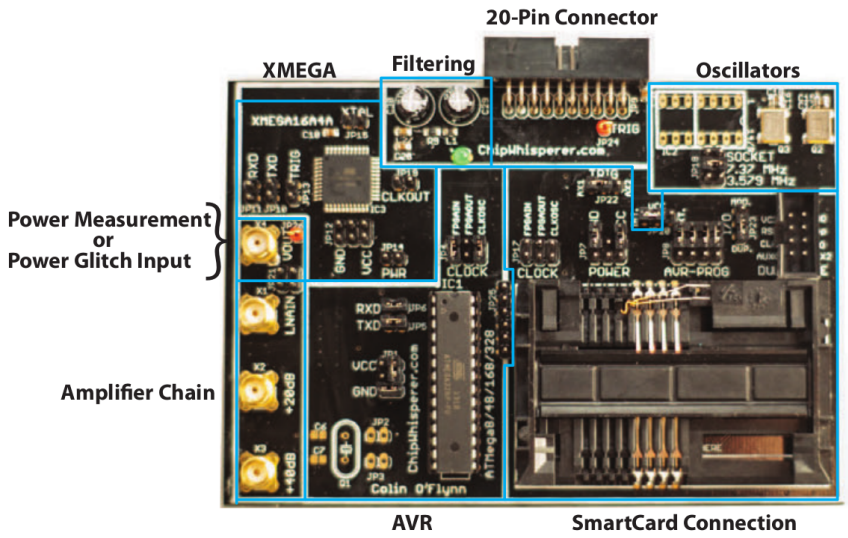

Figure 3.6: The ChipWhisperer Multi Target Victim Board. Image from [OC14].

In this thesis, both the ChipWhisperer SCA 'Capture' and 'Analysis' software were *not* utilised. Instead, the 'Capture' function was implemented in the Inspector software suite so that controlling the ChipWhisperer hardware to perform acquisition of power traces could be done through the Inspector software interface. The reason for this is explained in Section 3.4.5.

3.4.4 Cryptographic device under attack

ChipWhisperer has a 'Multi-Target Victim Board' as an optional add-on. This board is shown in Figure 3.6 and consists of three different targets (an AVR, an XMEGA and a smartcard socket) that serves to demonstrate basic attacks through online tutorials. Only the smartcard socket is used for this thesis.

3.4.5 Software Customisation

Other than controlling the measurement hardware (i.e. the ChipWhisperer Capture Rev2 or ChipWhisperer-Lite), the ChipWhisperer software on the attacker's PC also needs to control the cryptographic target. The target might be directly connected to the PC, e.g. via a Universal Serial Bus (USB), or connected via the ChipWhisperer hardware. In the latter case, the ChipWhisperer hardware then acts as a relay for serial communications between the target and the PC.

Some targets can be operated with very simple serial commands and it is easy to customise the ChipWhisperer software to interface with them. Unfor-

tunately, the Training Card 8 target described in Section 3.6.1, communicates with the T=1 protocol from the ISO 7816-3 standard which has not yet been implemented in the ChipWhisperer software. Implementing it from scratch in the ChipWhisperer Capture software would require significant development effort as the protocol involves error handling, communication acknowledgements and also maintenance of the communication state and sequence numbers. Moreover, such functionality already exists in the Inspector software suite. The communication protocols to communicate with the other targets used in our experiments also exist in the Inspector software suite.

As such, we decided to interface the ChipWhisperer Capture Rev2 and ChipWhisperer-Lite hardware with Inspector. Inspector is implemented in the Java programming language so this process involved converting portions of the existing ChipWhisperer code from Python to Java, and making it fit into the Inspector software framework. After we accomplished this successfully, we could operate the ChipWhisperer hardware through the Inspector software interface. Although this choice also required development time, thereafter we could interface the ChipWhisperer measurement hardware with a wider range of Riscure developed targets that includes both smartcards and embedded devices.

The Inspector software also has a more complete set of statistical and SCA analysis tools which we utilise to align the different trace sets. The purpose of alignment is described in Section 3.5.1. The comparison metrics were implemented as standalone Python scripts not bound to either the Inspector or ChipWhisperer software. These metrics are described in Section 4.

3.4.6 Hardware Customisation

Other than interfacing the ChipWhisperer Hardware with the Inspector software, several other customisations were made to the ChipWhisperer hardware, i.e. the Hardware Description Language (HDL) of the FPGA, in order for it to effectively make acquisitions with the targets described in Section 3.6.1.

Porting code to a different version of the ZTEX FPGA module

The ChipWhisperer project contains the C code and Verilog HDL for the microcontroller and FPGA of the ZTEX FPGA module. Unfortunately, this code is for the older and already obsolete ZTEX FPGA module 1.11 series and cannot work directly on the 1.15 series used in this thesis. Modifications to the code were required to remap the physical pin connections as the FPGA and microcontroller are connected differently across different versions of the board. These modifications were complicated by an outdated version of the FPGA constraints

file (.ucf) on the OpenADC git repository[3] (The correct version was located in the ChipWhisperer git repository[4]). This resulted in IO signals being routed to unexpected pins and substantial debugging. The correct porting process is documented in Appendix A.

This process is not required for the ChipWhisperer-Lite as all components comes pre-mounted on a single integrated PCB. Although this limits hardware flexibility, it also simplifies the setup process for users.

Acquisition Buffer

The acquisition buffer is implemented in the FPGA as a First In, First Out (FIFO) memory queue with a maximum storage space for 25k samples. After modifying the FPGA HDL description, this was increased to 100k samples which was required to measure software DES encryption on a smartcard (which was long and slow). In the process, together with the ChipWhisperer developer Colin, we discovered and fixed a logical error in the FPGA register controller that caused the register controller in the FPGA to cease responding when data transfers over 2^{16} bytes were requested.

Smartcard Communication

Smartcard communication is managed by the smartcard hardware module within the FPGA. For the attacker to communicate with the smartcard, he needs to send an appropriate command from the attacker's PC to the ChipWhisperer hardware. The register controller in the FPGA receives the command and forwards it to the smartcard hardware module. Finally, the smartcard hardware module will parse the command and drive the hardware IO pin that is connected to the smartcard IO.

The original smartcard hardware module in the FPGA was the 'serial_scard_hls_iface' module. This module was ISO 7816-3 aware and parses the Application Protocol Data Unit (APDU) commands before sending them. Unfortunately, its implementation resulted in some limitations which were:

- Only payloads of 16 bytes or less are allowed
- Only ISO7816-3 T0 protocols are allowed and not T1 protocols
- The APDU message with zero size payload and non-zero expected response length causes the module to go into an unknown state (crash).

[3]https://www.assembla.com/code/openadc/git/nodes/master
[4]https://www.assembla.com/code/chipwhisperer/git/nodes/master

Midway through the thesis, a new ChipWhisperer FPGA smartcard module 'reg_serialtarget' was developed by Colin. Unlike the previous smartcard module, this new module did not interpret the APDU packet but merely conveyed whatever was sent by the PC, directly to the smartcard and vice versa. This allows the ChipWhisperer user to perform all communication processing using a higher level language on the PC. This was not only more user friendly but also facilitated the integration of the ChipWhisperer hardware with Inspector's ISO7816-3 software stack that was explained in Section 3.4.5

With this integration, the ChipWhisperer user could communicate with smartcards through the Inspector software without bothering about the details of the underlying T0 or T1 communication protocols. The user just needs to define the APDU and the Inspector ISO7816-3 software stack will handle all the underlying communication protocols such as parsing the Answer to Reset signal, negotiating with the smartcard to select the most appropriate communication protocol (T0 or T1) and baud rate, then handling the T0 or T1 requirements appropriately.

Smartcard RESET

The smartcard RESET pin in the Multi Target Victim Board was originally floating and not connected to any physical driver. As such, an extra IO pin from the FPGA had to be used to control the RESET signal. The reset signal provided by the ChipWhisperer Python software also did not follow the ISO 7816 standard as the signal generated was inverted. This issue was subsequently addressed in the ChipWhisperer code and online documentation. Discussions with the ChipWhisperer developer on this issue is documented on the ChipWhisperer forum at `https://www.newae.com/forum/viewtopic.php?t=12`.

Smartcard Trigger

Like most other oscilloscopes, the OpenADC requires a hardware trigger signal to indicate the start of an acquisition. For an embedded target which we are able to program and has an extra IO pin, a common strategy is to insert additional logic in the target to raise the IO pin high just before the start of the encryption. This signal acts as the trigger to the OpenADC or oscilloscope. This strategy was not possible for smartcards because no unused IO pins were available. We also tried to use the pattern based triggering function of the ChipWhisperer which was designed to automatically generate a trigger signal to the OpenADC upon detection of a pre-set analog pattern in the power measurement or the IO line. Unfortunately we were not able to get this working reliably.

The smartcard communication chain between the attacker's PC and smart-

card was described in Section 3.4.6. Before a smartcard cryptographic operation occurs, there is usually a series of APDU exchanges to initialise the smartcard. We are not interested in the power leakages from the smartcard during such operations. After these APDU exchanges, the 'CRYPTO COMMAND' APDU from the attacker's PC instructs the smartcard to perform the encryption. After receiving this command, the smartcard performs an encryption and it is this power leakage which we wish to measure.

As such, we experimented with issuing a software trigger command from the attacker's PC, both before and also after the PC sends the 'CRYPTO COMMAND' APDU to the ChipWhisperer. Unfortunately, this approach suffers from a lot of timing jitter and had to be abandoned. This was likely caused by the non-real time nature of the general purpose OS on our PC, and also the many layers of communications between the start of the software command on the PC and the actual trigger signal to the OpenADC.

To overcome this issue, we amended the FPGA block in the ChipWhisperer that controls smartcard communications to send an internal trigger signal to the OpenADC just before the last byte of the 'CRYPTO COMMAND' APDU is sent to the smartcard target. This strategy leads to very stable and precise triggering and is also the approach taken by Riscure's PowerTracer [BV11].

Unfortunately, this approach did not work with the ChipWhisperer-Lite because it uses the microprocessor's built-in smartcard interface to communicate with the smartcard instead of a module within the FPGA. It might be possible to reprogram the microprocessor to achieve a similar triggering ability but this was not pursued due to time limitations.

3.5 Inspector SCA

Inspector SCA is a high-end commercial suite of tightly integrated hardware and software tools developed by Riscure. The hardware consists of a wide range of SCA and fault injection tools that include pattern triggering devices, laser stations for fault injection and low noise measurement equipment. The software has a wide set of signal processing functions such as filters, spectrum and alignment as well as SCA analysis capability for all the major SCA attacks and crypto algorithms. A lot of research and development has been dedicated to make both the hardware and software tools better and faster. The Inspector SCA setup (with only a subset of the hardware) is shown in Figure 3.7.

The only Inspector signal processing function we utilise is alignment. This is because we seek to minimise the software processing performed on the traces (e.g. filtering) to better observe the differences in hardware measurement setups. By applying static alignment, we remove any noise associated with trigger

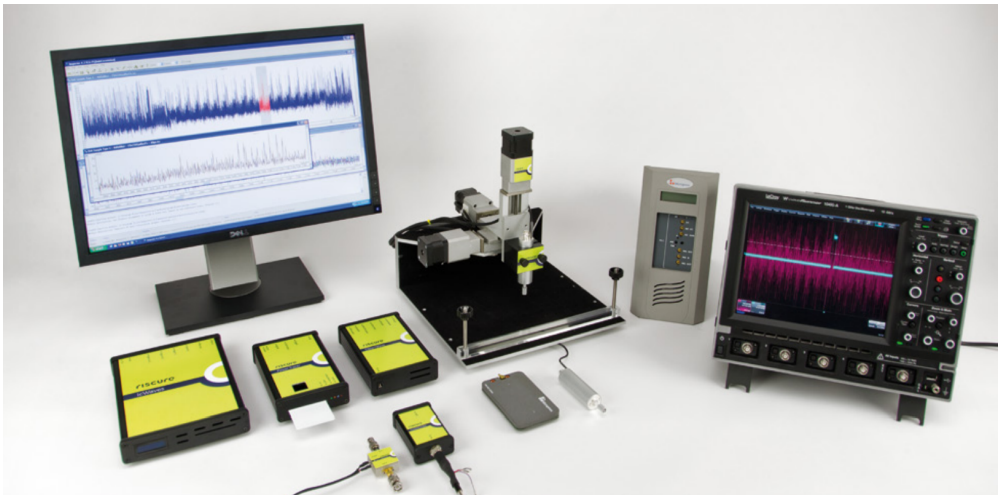

Figure 3.7: Inspector SCA with Power Tracer (power measurement circuit for smartcards), Electromagnetic (EM) Probe Station, CleanWave (analog filter and demodulator for EM measurements), Current Probe, amplifier and Lecroy Oscilloscope. Image from [BV15b].

inaccuracy or trigger delays and so focus only on the hardware acquisition quality of the different setups. The two Inspector measurement hardware used in this thesis are the Power Tracer [BV11] and the Current Probe [BV15a].

3.5.1 Alignment

Alignment is as a signal processing technique to counteract SCA hiding countermeasures such as time shifting or dummy operations that cause power traces to be misaligned from each other in the time domain. In practice, even in the absence of countermeasures, lack of precision in the trigger signal or communication delays can also cause small misalignments. These misalignments increase the noise of each sample point and also the number of traces required for a successful attack. As such, aligning the traces is a common pre-processing step before any SCA analysis is carried out.

We use the Inspector software module to perform the basic alignment technique introduced by Mangard et al in [MOP08] called 'static alignment'. This alignment technique works by finding a distinctive pattern in the first power trace near to the sample point of interest. This pattern is then identified in all the other power traces e.g. by using cross correlation to identify the best match. Finally, all the power traces are shifted so that the patterns occur at the same time instant in the power trace.

Static alignment can effectively correct timing inaccuracies caused by the trigger but is less effective against cryptographic devices that use varying clock

frequencies or delays inserted at random time instances. The elastic alignment module of Inspector SCA was designed to overcome such SCA countermeasures by dynamically 'stretching' and 'shrinking' traces. As none of our targets use such SCA countermeasures, this alignment technique was not used and will not be described. The interested reader can reference [vWWB11] for details.

3.5.2 Power Tracer

The Power Tracer is a power measurement circuit for smartcards. The Power Tracer reduces conducted and radiated emissions (that arise internally and from the attacker's PC) by separating digital and analogue circuitry. It reduces external noise from the power supply by running only on pre-charged capacitors during measurement. It maximises voltage stability and signal bandwidth by measuring power consumption without any measurement resistance [BV11]. Finally, the smart card supply voltage, power signal gain and offset are also configurable from the software interface.

3.5.3 Current Probe

To measure the power consumption of the cryptographic device, one could simply insert a resistor and measure the voltage across that resistor. However, the resistance value that provides the best results may vary significantly depending on the actual cryptographic device; If the resistance value is too small, measuring the voltage drop across that resistor can be difficult. If the resistance value is too large, then it may cause the circuit to malfunction and add thermal noise.

The Riscure approach is to use the Current Probe which senses the current flowing across the circuit and converts it to a voltage for an oscilloscope to measure using a transformer. Because of its very low inductance, it places minimal load on the cryptographic device and is hence versatile enough to be used for both low and high power cryptographic devices.

The Current Probe uses high end analogue components that are designed to introduce very low noise. It also has a high bandwidth of 2.5GHz which enables it to be used for high speed circuits. To use it, the current probe is inserted into a break in the circuit of the cryptographic device as shown in Figure 3.8.

3.6 Commercial Oscilloscopes

The commercial oscilloscopes used in the experiments are the entry level PicoScope 5203 and the more advanced Lecroy HDO6104. Through the Inspector software interface, we are able to configure the oscilloscope settings such as the

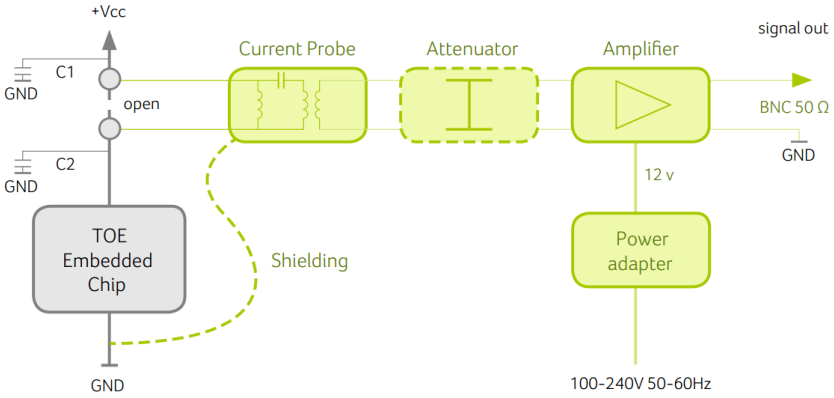

Figure 3.8: Inspector's Current Probe. Image copied from [BV15a].

sampling speed, number of samples and voltage range. The specifications of the oscilloscopes are compared against the OpenADC in Table 3.1.

Table 3.1: Specifications of the Oscilloscopes and OpenADC

	PicoScope 5203	**Lecroy HDO6104**	**OpenADC**
Max Sampling Frequency	1 Giga Samples/s (GS/s)	2.5 GS/s	105 MS/s
Max Resolution	8 bits	12 bits	10 bits
Max Bandwidth	250MHz	1GHz	120 MHz

3.6.1 Cryptographic Device under Attack

We used three different cryptographic devices from Riscure that were meant for customer training and internal R&D. These particular devices were chosen to represent a variety of common targets.

1. **Riscure Training Card 6** is a smartcard with a software DES implementation. It has no SCA countermeasures such as random shifting or masking. Like most smartcards, the smartcard processor uses an external clock signal generated by the smartcard reader. As such, the clock signal is easily accessible and synchronous sampling with the OpenADC is easy to achieve. Moreover, the external clock frequency that was used is 3.579MHz, which is much lower than the limits of the OpenADC. Training Card 6 communicates with the T=0 protocol of ISO 7816-3.

2. **Riscure Training Card 8** is a smartcard with a hardware DES implementation. It has no SCA countermeasures such as random shifting or masking although the implementation has a low signal to noise ratio.

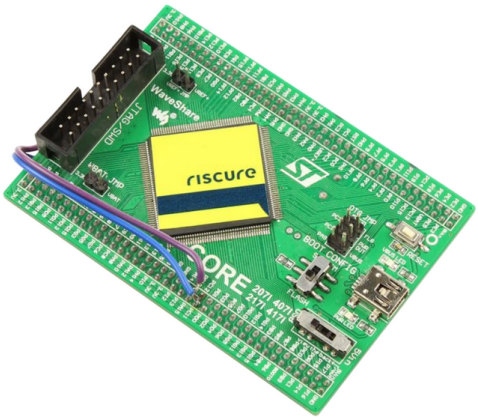

Figure 3.9: Riscure Pinata Board.

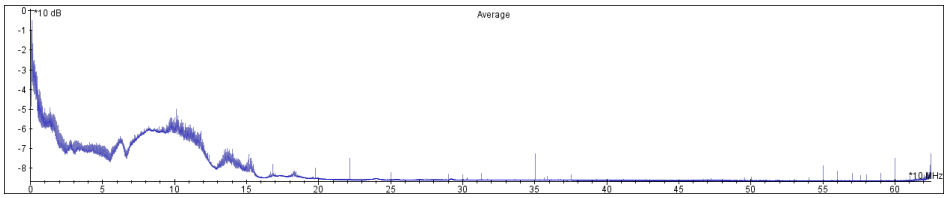

Figure 3.10: Riscure Pinata Board Leakage Spectrum.

Although the main smartcard processor uses an external clock, the DES implementation is running on a separate hardware crypto processor with its own internal clock of 31 MHz. This represents a more unfavourable situation for the ChipWhisperer compared to Training Card 6 as it is significantly more difficult for the OpenADC to synchronise to this internal clock and thus asynchronous sampling has to be used. Training Card 8 communicates with the T=1 protocol of ISO 7816-3.

3. **Riscure's Pinata Board** is an embedded processor from ST Microelectronics (STM 32F407IG) that is based on the ARM Cortex-M4F with a clock frequency of 168MHz. It has a software DES implementation with no SCA countermeasures. The board is shown in Figure 3.9. The clock is internal to the embedded processor and moreover, has a frequency which is higher than the OpenADC specifications. Nevertheless, due to the capacitances present on the board, the leakage happens at a lower frequency than the OpenADC bandwidth of 120MHz. The spectrum of the Pinata board leakage is shown in Figure 3.10. The Pinata board is controlled directly by the PC via Universal Serial Bus (USB).

Chapter 4

Evaluation Metrics for SCA

In 2009, Standaert et al formalised the notion of an evaluation metric [SMY09] and proposed two different classes of metrics; one class of metric to compare SCA attackers and another to compare SCA countermeasures. The goal of such a distinction would be to make statements such as "an implementation X is better than an implementation Y", without making conditional statements about the adversary's capabilities. Such a goal is also desirable when comparing measurement setups as we wish to decouple the comparison of measurement setups from the cryptographic device under test.

Since then, several metrics in both classes have been proposed. Unfortunately, metrics for SCA attacks are usually designed for SCA algorithms and not the underlying measurement hardware. Similarly, metrics for SCA countermeasures focus on the SCA resilience of the cryptographic device under test and pay even less heed to the measurement hardware. As such, when SCA measurement setups need to be evaluated as in [DO09], [OC13b] or [OC13a], it is usual to default to the 'success rate' metric which is the most well known metric. This might not be an optimal choice.

In this Chapter, our goal is to evaluate all the SCA evaluation metrics that have been proposed in academia and conduct a formal study to assess which metric is the most suitable for SCA measurement setups. The evaluation metrics are listed under three usage classes in Table 4.1. The first two usage classes are 'SCA Attacks' and 'SCA Countermeasures' that were proposed by Standaert et al in [SMY09]. The third class of metric measures the 'SCA resistance of cryptographic algorithms'. The impact of this new class of metrics was first recognised in [PPE+14]. In that paper, Picek et al designed new SCA resistant S-Boxes for the AES and PRESENT ciphers with good 'Confusion Coefficient' [FLD12] properties. The authors showed that these improved S-boxes translated to better SCA resistance on the cryptographic device.

It is clear that the metrics in the third class are not suitable for comparing measurement setups but it is not immediately evident which metric proposed for SCA attacks or SCA countermeasures is the most suitable. This Chapter is organised as follows: First we describe the metrics from the first two classes in Section 4.1. Next, in Section 4.2, we implement these metrics and apply them experimentally on different setups to assess their suitability. Finally in Section 4.3.1, we discuss why we have determined signal to noise ratio to be the most suitable metric.

SCA Attacks	**SCA Countermeasures**	**SCA Resistance of Crypto Algorithm**
Success Rate Relative Distinguishing Margin Signal to Noise Ratio Support Confidence	Mutual Information Signal to Noise Ratio	Confusion Coefficient Transparency Order ...

Table 4.1: Comparison Metrics proposed in Academia. Signal to noise ratio had been proposed as a metric to compare SCA distinguishers and SCA hardware countermeasures and so appears under both the first and second class.

4.1 Description of Metrics

4.1.1 Success Rate

The success rate metric has several variations which are the o-th order success and partial guessing entropy as defined in [SMY09]. These metrics are based on the guess vector $\mathbf{g} = [g_1, g_2, ..., g_{2^n}]$ from an SCA attack described in Section 2.2.1.

An o-th order success is the situation in which the correct subkey is ranked among the first o key guesses of $\mathbf{g}$. An o-th order success rate is defined as the probability of o-th order success. When the order is not defined, it is assumed to be 1. Success rate is mostly used in relation to a single subkey but is sometimes used in relation to the full key that comprises all subkeys. The DPAContest [dpa] terms the metric as 'partial success rate' and 'global success rate' when used for a single subkey or the full key respectively. All subsequent reference to success rate in this thesis will be for a single subkey.

'Partial guessing entropy' measures the average number of key guesses to test after the side channel attack. It is calculated simply as the rank of the correct subkey. For example, if the correct subkey is g_i, then its partial guessing entropy is i.

Both success rate and partial guessing entropy are usually plotted against

the number of measurements of the leakage. As the number of measurements increase, one typically expects better results i.e. both the success rate and guessing entropy approach 1. To empirically calculate both o-th order success rate or partial guessing entropy, it is typical to perform the attack several times (usually 100 or 1000).

The problem with this approach is that it is very resource consuming if the target is 'strong' [1]. This is because the minimum number of leakages required to calculate the success rate is then the number of leakages for a success rate of 1, multiplied by a factor of 100 to 1000. Recently, some papers such as [Riv09] and [FLD12] attempt to calculate the success rate more efficiently using theoretical methods which are discussed in Chapter 5.

4.1.2 Relative Distinguishing Margin (RDM)

The relative distinguishing margin metric was proposed by Whitnall and Oswald in [WO11b] and [WO11a] as the distance between the distinguisher value of the correct subkey (i.e. the actual subkey that was used in the cryptographic device) and the value of the highest ranked alternative, normalised by the standard deviation of the distinguisher values of all hypothetical sub-keys. The distinguisher value of a subkey s is denoted as $D(s)$ and depends on the SCA attack being used. If the Pearson's correlation coefficient is used, then the distinguisher value will be the Pearson's correlation coefficient calculated using that subkey i.e. $D(s) = \rho_s$. If MIA or the template attack is used, then $D(s) = \Pr[L|s]$.

The formula for relative distinguishing margin is given as

$$\text{RDM} = \frac{D(s^*) - \max\{D(s) : s \neq s^*\}}{\sigma(\{D(s) : s \in \mathcal{S}\})}, \tag{4.1}$$

where s^* denotes the actual subkey that was used in the cryptographic device.

The relative distinguishing margin is closely related to a family of other metrics also proposed by Whitnall and Oswald such as the 'average distinguishing score', 'absolute distinguishing margin' or 'standard score'. These metrics either replace the value of the highest ranked subkey alternative with the mean of the distinguisher value, or use a different normalisation factor.

The relative distinguishing margin is useful because it informs us of how well a distinguisher can differentiate the correct key from incorrect key guesses. A higher value from the RDM metric implies a better attack as the attacker gains more confidence that his subkey guess is correct. It can be used to compare different SCA attacks because of the normalisation with standard variance.

[1] Strong in the sense that it requires a large number of leakages to successfully retrieve the key

For better statistical properties, we also perform the attack several times and average the result.

4.1.3 Mutual Information

Mutual information between the S-Box input X and leakages L was proposed by Standaert et al in [SMY09] as an information theoretic metric for evaluating SCA countermeasures. It is defined as

$$\mathrm{I}(X;L) = \mathrm{H}[X] + \sum_{x \in \mathcal{X}} \Pr[x] \sum_{l \in L} \Pr[l|x] \cdot \log_2 \Pr[x|l], \tag{4.2}$$

which follows from the standard definition of mutual information in information theory which is $\mathrm{I}(X;L) = \mathrm{H}[X] - \mathrm{H}[X|L]$. $\mathrm{H}[X|L]$ can be re-written as

$$\begin{aligned} \mathrm{H}(X;L) &= -\sum_{l} \Pr[l] \sum_{x} \Pr[x|l] \cdot \log_2 \Pr[x|l] \\ &= -\sum_{x} \Pr[x] \sum_{l} \Pr[l|x] \cdot \log_2 \Pr[x|l]. \end{aligned} \tag{4.3}$$

The advantage of using mutual information as a metric is that it allows capturing any kind of dependency in the physical leakages [VCS09]. In the following discussions, we consider leakage traces with only a single sample point, i.e. $r = 1$. In Equations (4.2) and (4.3), $\mathcal{X}$ and $\mathcal{L}$ are the set of all possible input values and leakages respectively. $\mathcal{X} = \{0,1\}^n$ and $\mathcal{L} = \frac{1}{\text{Oscilloscope Voltage Range}} \cdot \{0,1\}^e$ where e denotes the bit resolution of the oscilloscope.

Calculation Method

Before calculating the mutual information, we first need to estimate $\Pr[l|x]$ and $\Pr[x|l]$ using either a non-parametric or parametric method described in Section 2.2.5. For simplicity, we follow the approach taken in [RSVC+11] which is to use the profiling phase from the template attack to estimate $\Pr[l|x]$. This is a parametric method and relies on the assumption that noise follows a univariate gaussian distribution (because we use only a single sample point).

We build 2^n templates that consist of a data dependent signal (calculated as the mean of all leakages with the given S-box input) and the noise variance. We can then estimate $\Pr[l|x]$ as the normal distribution, $\mathcal{N}(\mu_{l|x}, \sigma^2_{l|x})$. $\Pr[x|l]$ can now be obtained from $\Pr[l|x]$ using Bayes' formula as

$$\begin{aligned}\Pr[x|l] &= \frac{\Pr[l|x] \cdot \Pr[x]}{\Pr[l]} \\ &= \frac{\Pr[l|x] \cdot \Pr[x]}{\sum_{x' \in \mathcal{X}} \Pr[l|x'] \cdot \Pr[x']} \\ &= \frac{\Pr[l|x]}{\sum_{x' \in \mathcal{X}} \Pr[l|x']}.\end{aligned} \tag{4.4}$$

In this derivation, we are able to cancel out $\Pr[x]$ from the nominator and denominator because using a random distribution of plaintexts under a fixed subkey, $\Pr[x]$ becomes a constant i.e. $\frac{1}{2^n}$. Because the probability distributions are continuous, we can also calculate mutual information using the continuous representation as

$$\mathrm{I}(X;L) = \mathrm{H}[X] + \sum_{x \in \mathcal{X}} \Pr[x] \int_{-\infty}^{\infty} \Pr[l|x] \cdot \log_2 \Pr[x|l] \tag{4.5}$$

4.1.4 Signal to Noise Ratio

Signal to noise ratio has been proposed as a metric for both SCA attacks and countermeasures. It was first used by Messerges et al in 1999 [MDS99] to determine the amount of information an SCA attack could extract from leakages. Later in 2004 [Man04], Mangard used SNR to compare the efficacy of hardware countermeasures.

Most methods to calculate SNR assume a linear leakage model with gaussian noise as

$$L(p,s) = \epsilon \cdot V(p,s) + L_N + L_0. \tag{4.6}$$

The leakage $L(p,s)$ is the oscilloscope measurement in Volts and is a function of the data being processed, i.e. the S-Box input state p and subkey s. It is comprised of an exploitable component ($\epsilon \cdot V(p,s)$), a noise component L_N which follows a gaussian distribution of $\mathcal{N}(0,\sigma^2)$ and a constant component (L_0) which is an offset caused by other unrelated parts of the cryptographic device. The exploitable component is the output of the leakage model $V(p,s)$ multiplied by the constant ϵ. The constant component L_0 is independent of the exploitable component and the noise.

There are two common but different methods used to define SNR. An established method based on communications theory is $\mathrm{SNR} = \frac{\mathrm{Var(Signal)}}{\mathrm{Var(noise)}}$ which was used in [Man04] and [MOP08]. In this equation, 'Signal' represents the exploitable component ($\epsilon \cdot V(p,s)$) and Var(Noise) is σ^2. More recently, several

papers such as [GMS+11] and [FLD12] use an alternative definition of

$$\text{SNR} = \frac{\epsilon}{\sigma}. \tag{4.7}$$

In the remaining chapters of this thesis, we use this definition.

Calculation Method - Signal

A simple and intuitive method to calculate ϵ is to use $\mu_{i+1} - \mu_i$ where $\mu_i = \text{E}_j[l_j], \forall V(p_j, s) = i$. In this thesis, we use $E_i[\mu_{i+1} - \mu_i]$ to reduce estimation errors.

Guilley et al showed in [GMS+11] that the signal can alternatively be calculated with the covariance of $L(p, s)$ and $V(p, s)$ as

$$\begin{aligned} \text{cov}(L(p,s), V(p,s)) &= \text{cov}(\epsilon V(p,s) + L_N + L_0, V(p,s)) \\ &= \text{cov}(\epsilon V(p,s), V(p,s)) \\ &= \epsilon \cdot \sigma^2(V(p,s)) \end{aligned} \tag{4.8}$$

claiming better accuracy with their method. The formulas for covariance and σ are defined in Equation (2.2) and (2.3). The authors also showed that assuming a Hamming Weight leakage model, $\sigma^2(V(p, s))$ can be estimated as $\frac{\text{Number of bits in S-Box output}}{4}$.

The results from both methods are compared in Figure 4.1 with results from the first and second method labelled as 'mean' and 'covariance' respectively. In this experiment, for a given number of measurements (along the x-axis), we calculate the value of ϵ using both methods for 10 non-overlapping groups of leakages. It can be observed that the 'covariance' method has a wider spread of ϵ values when the number of leakages is low. Nevertheless, both methods converge to the same value as the number of leakages increases. Here, we note that ϵ does not change with the number of measurements but rather, it is the accuracy of our estimation that varies. As such, checking for convergence is a good way to ensure that we have an accurate representation of ϵ.

There are also some signal calculation methods proposed in academia that are based on faulty assumptions. [FDLZ14] states that the CPA and Distance of Mean distinguishers share the same ϵ which is the distance of means for the correct subkey. Their rationale is that theoretically, the difference in $L(x)$ caused by a 1-bit transition of a single S-box bit, or the entire S-Box output should be no different.

However, in practice, ϵ calculated using different bits of the same S-Box varies. In Figure 4.2, we see that ϵ values calculated using DOM with bit 0, bit

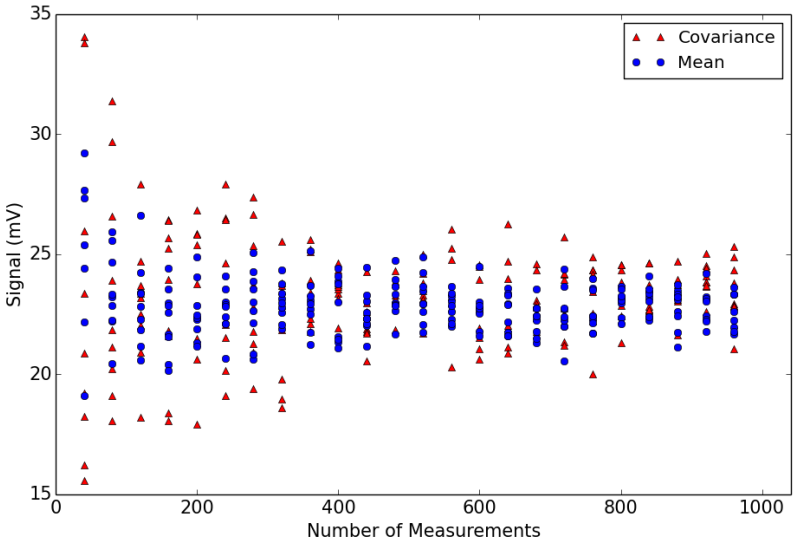

Figure 4.1: Comparison of two different methods to estimate the signal in millivolts.

1, bit 2, bit 3 of DES S-box 1 are quite different from each other and also the ϵ value calculated using CPA. However, if we take the average of the DOM from bit 0 to bit 3, then this value is close to CPA's ϵ value.

Calculation Method - Noise

The noise is calculated as the mean of the standard deviation of the leakages grouped according to the value of $V(x)$ as $\mathrm{E}_i[\sigma_i]$ where $\sigma_i = \sigma[\hat{L}_i]$ such that $\hat{\mathcal{L}}_i = \{l_j | j \in \mathbb{Z}, 0 < j \leq q, V(p_j, s) = i\}$.

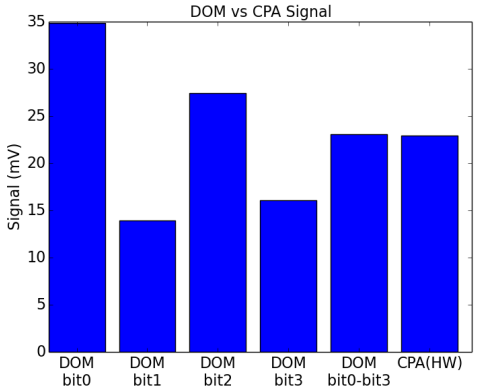

Figure 4.2: Signal from different bits of the S-box.

4.1.5 Support

The support metric was also proposed by Whitnall and Oswald in [WO11b] and [WO11a]. This refers to the success rate of an attack given a reduced subset of the plaintext space. This metric is used to consider a limited profiling phase of SCA profiling attacks. We do not consider it relevant to comparing measurement setups.

4.1.6 Confidence

The confidence metric was recently proposed by Thillard in [TPR13] as

$$c(s) = \frac{\Pr[S = s]}{\sum_{i=0}^{2^n-1} \Pr[S = i]} \tag{4.9}$$

$c(s*)$ diverges from 1-st order success rate only when the attack refrains from making a key guess when its confidence of being correct is low. Thillard argued that this was important because refraining from a key guess when the confidence was low would improve overall attack effectiveness; Choosing a wrong candidate subkey would lead to an overall failure to decrypt the ciphertext whereas a subkey that was not found could be brute forced which would still lead to a successful attack.

This metric is useful for comparing SCA attacks or distinguishers but again, we do not consider it relevant to comparing measurement setups.

4.1.7 Other Metrics

Outside academia, other metrics such as the monetary cost, development time and requisite knowledge are also important factors when comparing different setups. We do not discuss these metrics in this thesis.

4.2 Evaluation of Metrics

In this section, we evaluate how the metrics perform against each other experimentally. We use three different comparison sets chosen for specific properties that are described below. The specific details of each setup is detailed in Table 4.2. We label each setup 'better' or 'worse' based on their overall performance with the various metrics.

A Two very different measurement setups against Training Card 6 which is a software DES implementation that leaks the Hamming Weight of the S-Box output.

B Two very different measurement setups against Training Card 8 which is a hardware DES implementation that leaks the Hamming Distance between the input and output of each Feistel Round.

C Two measurement setups which differ only very slightly against Training Card 6.

Table 4.2: Setup Details

	Better Setup	**Worse Setup**	**Target**
A	Lecroy + Power Tracer	OpenADC + Multi-Target Victim Board (Smartcard Socket)	Training Card 6
B	Lecroy + Power Tracer	OpenADC + Multi-Target Victim Board (Smartcard Socket)	Training Card 8
C	Lecroy + Multi-Target Victim Board (Smartcard Socket)	PicoScope + Multi-Target Victim Board (Smartcard Socket)	Training Card 6

4.2.1 Success Rate

The comparison results using 1^{st}-order success rate and partial guessing entropy as metrics (with CPA as the distinguisher) are shown in Figure 4.3 and 4.4.

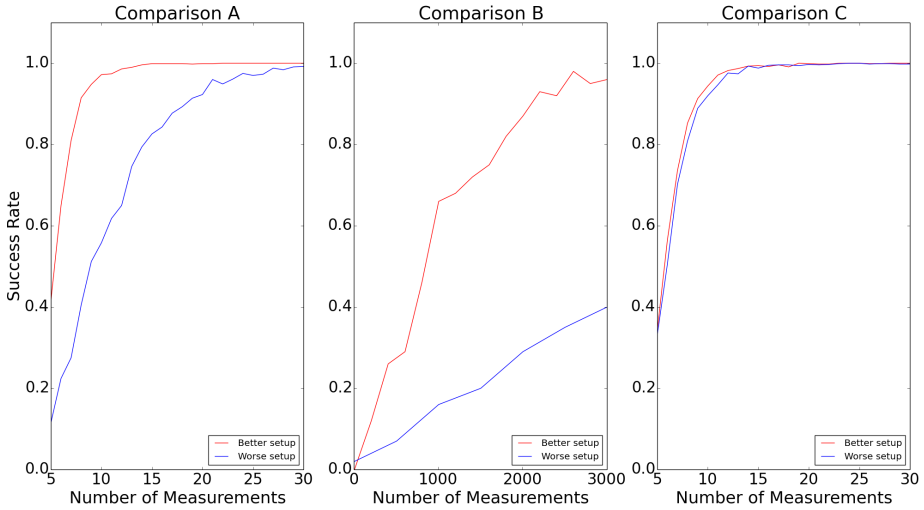

Figure 4.3: 1st order success rate.

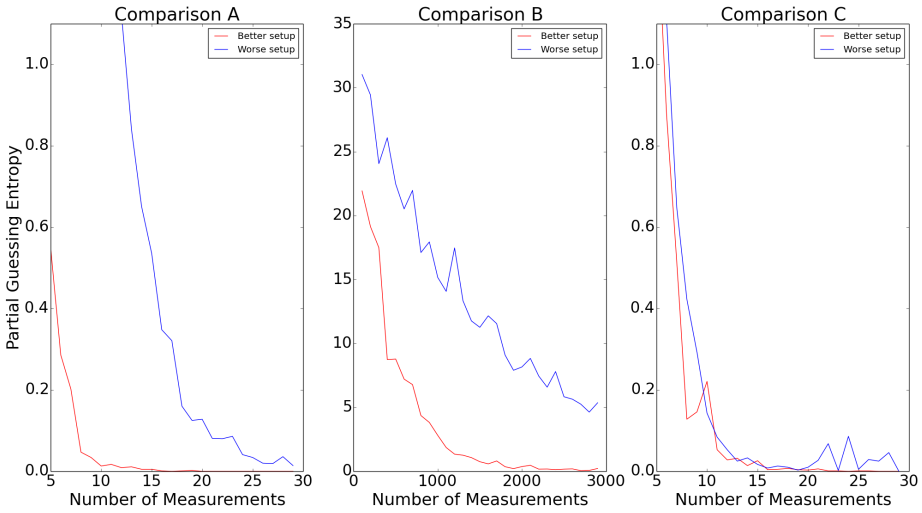

Figure 4.4: Partial Guessing Entropy.

Analysis

When the difference in quality of the measurement setups is big as per Comparison A and Comparison B, both metrics are able to differentiate between the setups. However, when the quality difference is small, both metrics are not as effective in differentiating between the measurement setups.

Moreover, the results for comparison A and C are computed using 1000 attack iterations (requiring 30k leakages) and results for comparison B and computed using 100 attack iterations (requiring 300k leakages) due to resource limitations. Nevertheless, using either 100 or 1000 attack iterations requires a large number of leakages and a long computational time. The actual computational time of the success rate metric and all other metrics is discussed in Section 4.3.1.

4.2.2 Relative Distinguishing Margin

The comparison results for relative distinguishing margin (with the CPA distinguisher) are shown in Figure 4.5. The results are computed using 100 attack iterations.

Analysis

The results are quite similar to success rate (because they are both based on the CPA distinguisher) although visually, relative distinguishing margin seems better able to distinguish measurement setups with a small quality difference.

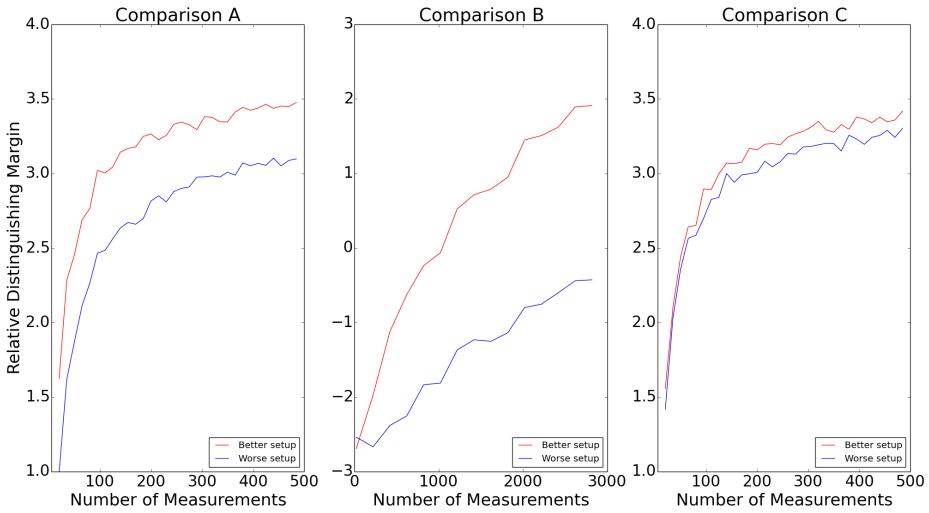

Figure 4.5: Relative Distinguishing Margin.

The number of leakages that are required as well as the computational time is similar to success rate as it is also computed using a large number of CPA attack iterations.

4.2.3 Mutual Information

The comparison results for $\mathrm{I}(X, L)$ are shown in Table 4.3.

	Comparison A	**Comparison B**	**Comparison C**
Better setup	2.48	0.0029	2.43
Worse setup	0.94	0.0032	2.04

Table 4.3: Mutual Information

Analysis

The MI metric is sensitive enough to notice both big and small differences in the quality of the measurement setups as evinced in Comparison A and C. Nevertheless, the results for Comparison B are *inaccurate* as the measurement setup with poorer quality actually has a higher MI value. This might be because the actual leakage model of the cryptography target in Comparison B is the Hamming distance of the round input and round output.

This shows that using MI as a metric with $\mathrm{I}(X; L)$ as recommended in [RSVC+11] and [SMY09] does not always provide the best results. We can apply here the findings from [VCS09] that 'while MIA better resists incorrect

leakage models than correlation attacks, it is not immune against them'. As such, we still need to take the leakage model into account by using $\mathrm{I}(V; L)$.

When we compute MI using $\mathrm{I}(V; L)$ with V denoting the predicted leakage (i.e. Hamming distance between round inputs and outputs for Comparison B and Hamming weight for Comparison A and C), we get the expected results whereby the better quality setup has a higher MI compared to the lower quality setup. $\mathrm{I}(V, L)$ is shown in Table 4.4.

	Comparison A	**Comparison B**	**Comparison C**
Better setup	1.96	0.0062	1.71
Worse setup	0.74	0.0016	1.46

Table 4.4: Mutual Information

To assess the number of leakages required for an accurate MI computation, we plot the results of the 10 different MI values computed using a different randomly selected set of leakages each time. We see that the mean and also the spread of the 10 values stabilises after 10k leakages which means that we have accurately modelled $\Pr[L|X]$.

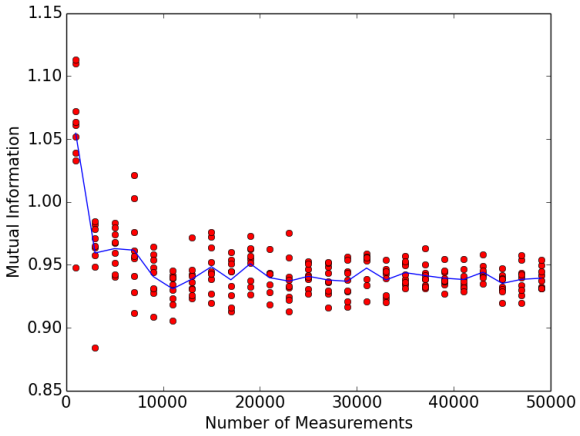

Figure 4.6: Mutual Information plotted against the number of leakages used to create the Gaussian Template.

As a note of caution when using the mutual information metric, the amount of information that can be extracted about X or V from L is limited. Beyond the maximum limit set by $\mathrm{H}(V)$ or $\mathrm{H}(X)$, any improvements in the measurement setup will not increase the mutual information. As an example, we consider a hypothetical $\Pr[L|V]$ (assuming a Hamming weight model) obtained from a less noisy setup in Figure 4.7 and a more noisy setup in Figure 4.8. Both set of leakages have the same $I(V, L) = 2.03$.

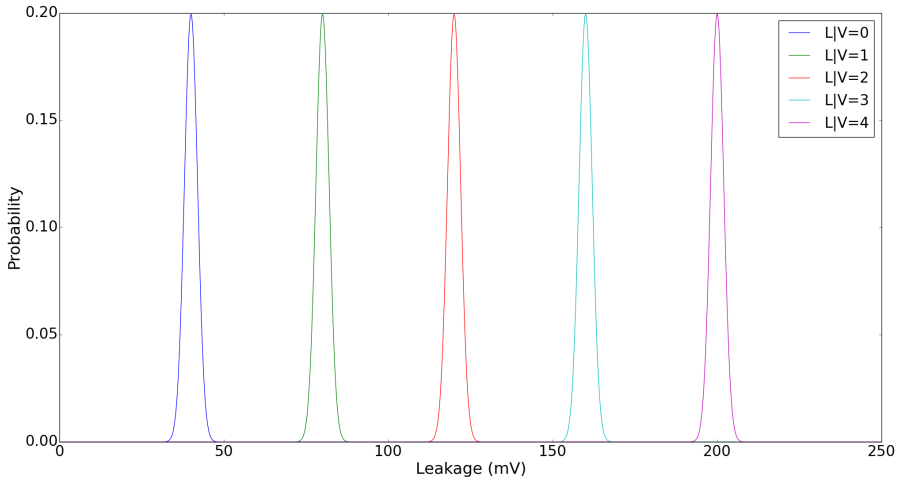

Figure 4.7: Measurement setup with less noise has a mutual information between V and L of 2.03.

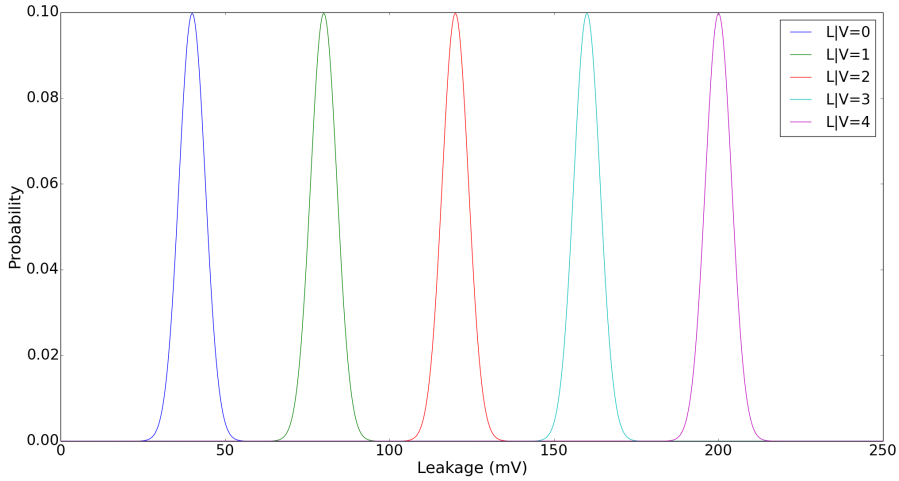

Figure 4.8: Measurement setup with less noise has a mutual information between V and L of 2.03.

4.2.4 Signal to Noise Ratio

The signal to noise ratio values are shown in Table 4.5.

Analysis

The SNR metric is sensitive enough to notice small differences in the quality of the measurement setups.

Figure 4.9 shows the plot of signal, noise and signal to noise ratios against the number of measurements. For each number of measurements, we obtain

	Comparison A	Comparison B	Comparison C
Better setup	4.90	0.090	3.68
Worse setup	1.39	0.045	3.37

Table 4.5: Signal to Noise Ratio

10 different SNR values from a randomly selected set of leakages. We see that the mean and also the spread of the 10 different SNR values stabilises after 1k leakages which means that we have obtained an accurate estimation of the SNR.

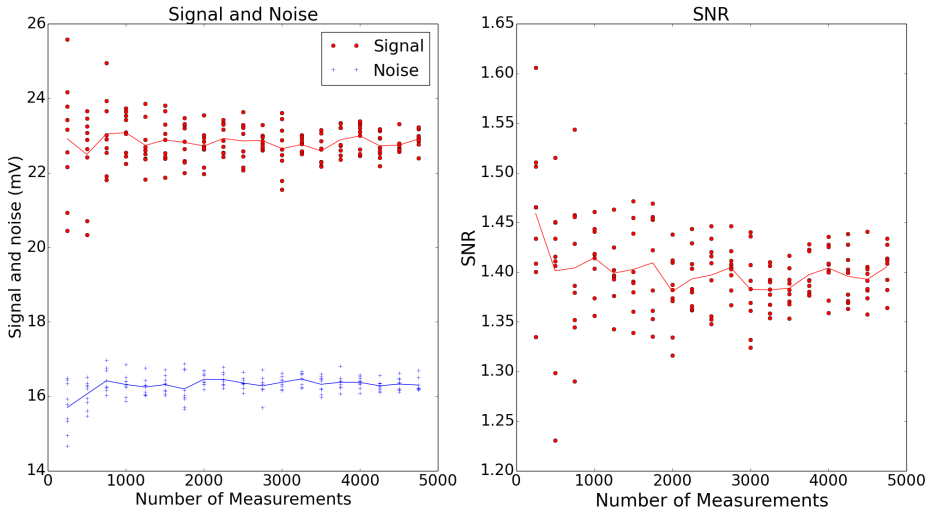

Figure 4.9: Signal to noise ratio.

4.3 Discussion

4.3.1 Computational Time

The computational time for Success rate, relative distinguishing margin and partial guessing entropy for N number of leakages is

$$\text{Computation Time} = 0.66 \cdot N \text{ seconds.} \tag{4.10}$$

This is the metric for a single value of leakages i.e. calculating the success rate for 100 leakages requires 66 seconds. Correspondingly, the time required to calculate the success rate curve from 0 to 100 leakages is $\sum_{i=0}^{100} 0.66 \cdot N$. Equation (4.10) is given for 100 attack iterations. It scales linearly with the number of attack iterations so the time taken for 1000 attack iterations is $6.6(N)$ seconds.

The computational time for calculating mutual information is

$$\text{Computation Time} = 0.0088 \cdot N \text{ seconds.} \tag{4.11}$$

The computational time for calculating SNR is

$$\text{Computation Time} = 0.0001 \cdot N \text{ seconds.} \tag{4.12}$$

All metrics are fully implemented using only the Python programming language without special speed optimisation techniques e.g. multithreading. The code is publicly available at `https://github.com/alvincai/SCA-Algorithms`.

4.3.2 Summary

The results from evaluating the different metrics are summarised in Table 4.6. Signal to noise ratio shows the best properties for comparing setups in terms of the ability to observe small differences in quality between setups while requiring the least computational time and number of leakages. As such, the SNR metric is the best choice when we are only interested in the relative performance of one measurement setup against another. Future evaluations of measurement setups in both academia and the industry will benefit by using the SNR metric instead of the success rate metric.

Signal to noise ratio is used for all subsequent setup comparisons in Section 6.

Table 4.6: Summary of the properties of the various metrics.

	Ability to notice small quality differences	**Computational Time**	**Number of Leakages**
Success Rate	Poor	Very High	Very High
Relative Distinguishing Margin	Moderate	Very High	Very High
Mutual Information	Good	Moderate	Moderate
Signal to Noise Ratio	Very Good	Low	Low

Chapter 5

Theoretical Success Rate

Success rate is a very interesting metric for both SCA attackers and security designers as the number of measurements required for a successful attack is one of most practical measures of security. Nevertheless, a major drawback mentioned in Chapter 4 was the computational time and number of traces that was required in calculating this metric empirically.

Prompted by this problem, there have been several efforts to develop more efficient methods to calculate success rate. In [Man04] Mangard investigated the effect of SNR on the correlation coefficient and presented a statistical model linking the success rate to the number of measurements. However, this was not accurate as it was based on the wrong assumption that the correlation coefficient corresponding to the wrong key was asymptotically null.

Later, Rivain [Riv09] and Fei ([FLD12] and [FDLZ14]) were able to accurately model the success rate by relaxing this assumption. Both methods have also been extended in [LPR+14] and [DZFL14] to calculate the success rate for higher order DPA against cryptographic devices that employ SCA countermeasures such as masking. The success rate of higher order DPA will not be discussed in this thesis.

5.1 Rivain's Method

In [Riv09], Rivain showed that the CPA correlation coefficients tended towards a multi-variate guassian distribution which an SCA attacker could estimate. By making the assumption that plaintext inputs are randomly distributed and that physical leakages tends towards a univariate gaussian distribution that we can estimate through profiling (identical to the profiling phase of the template attack), Rivain showed that the simplified correlation coefficient expression (which he denoted as $\ddot{\rho}$) follows the same distribution as the original Pearson's corre-

lation coefficient in Equation (2.1). Again for simplicity, we assume that all leakages L have only a single sample point. Rivain's simplified correlation coefficient can be expressed as

$$\ddot{\rho}(V(p,s),L) = \mathrm{E}_i[V(p_i,s)L_i] \tag{5.1}$$

Working with this new representation, we can obtain the mean of $\ddot{\rho}(V,L)$ as

$$\mathrm{E}_p[\ddot{\rho}_s] = \frac{1}{2^n}\sum_{p\in\mathcal{P}} v(p,s)\cdot\mu_{p\oplus s^*}, \tag{5.2}$$

and its covariance as

$$\mathrm{Cov}[\ddot{\rho}_s,\ddot{\rho}_{s'}] = \frac{1}{q\cdot 2^n}\sum_{p\in\mathcal{P}} V(p,s)\cdot V(p,s')\cdot\sigma^2_{p\oplus s^*}. \tag{5.3}$$

Here, μ_x and σ_x are the templates which we obtained during the profiling phase. P represents the S-box input state, where $\mathcal{P} = \{0,1\}^n$ and q the number of leakages.

With the model of the correlation coefficients, Rivain showed that we could directly construct the comparison vector c_s which is another multivariate gaussian distribution of size $|\mathcal{S}|-1$ where $c_s = \ddot{\rho}_{s^*} - \ddot{\rho}_s$, with s^* representing the correct subkey. The Cumulative Distribution Function (CDF) of this distribution when the limits of all variables are positive is the 1-st order success rate. When y limits are negative, then we have success at the $(y+1)^{th}$ order. By varying the limits of the CDF of this distribution, we can accurately calculate the success rate of any order. This can be simply extended to calculate the partial guessing entropy; the most likely success rate order will be the guessing entropy.

5.2 Fei's Method

Fei et al in [FLD12] and [FDLZ14] has detailed an alternate approach to calculate the comparison vector based on a different mathematical model. In their proofs, they use the maximum likelihood method so that the comparison vector $c_s = \Pr[S=s^*] - \Pr[S=s]$. As per Rivain's formula, the 1-st order success rate is estimated with the CDF of this distribution when the limits of all variables are positive. Unfortunately, Fei et al's formula for calculating success rate has not proved popular perhaps due to it being more complicated to implement. Nevertheless, their work is a significant contribution because it has helped to isolate the SCA success rate factors to the cryptographic algorithm's inherent SCA resistance and also the quality of the physical implementation.

5.2.1 Algorithmic Properties

The resistance of a cryptographic algorithm to an SCA attack is measured by what Fei et al term as the 'confusion coefficient'. The confusion coefficient between two keys is the expectation of the squared distance between the leakage model outputs:

$$\kappa(s, s') = \mathrm{E}[(V|s - V|s')^2]. \tag{5.4}$$

A large value of $\kappa(s, s')$ indicates that the V values are different for a large portion of the S-Box inputs which makes it easy to distinguish s and s' through an SCA attack. Conversely, a small confusion coefficient makes the SCA attack harder but may suffer from differential attacks as shown in [HRG14].

5.2.2 Implementation Quality

Implementation quality is represented by the signal to noise ratio defined in Section 4.1.4. The SNR replaces the profiling required by Rivain's method, because it makes the additional assumption that the leakage takes the more simple form represented in Equation (4.6).

Finally, the mean and covariance of the comparison vector can be calculated with Equation (5.5) and (5.6) where $\varkappa(s^*, s^j)$ and $\varkappa^*(s^*, s^j)$ are the three-way confusion coefficients as defined in Equation (5.7) and (5.8). The derivation and proof of these formulas can be found in [FDLZ14] and will not be reproduced in this thesis.

$$\mathrm{E}[c_s] = \frac{1}{2}(\frac{\epsilon}{\sigma})^2 \cdot \kappa(s^*, s) \tag{5.5}$$

$$\mathrm{Cov}[c_{s_i}, c_{s_j}] = (\frac{\epsilon}{\sigma})^2 \cdot \varkappa_{ij} + \frac{1}{4}(\frac{\epsilon}{\sigma})^4 \cdot (\varkappa^*_{ij} - \kappa(s^*, s_i) \cdot \kappa(s^*, s_j)) \tag{5.6}$$

$$\varkappa(s_i, s_j) = \mathrm{E}[(V|s^* - V|s_i) \cdot (V|s^* - V|s_j)] \tag{5.7}$$

$$\varkappa^*(s_i, s_j) = \mathrm{E}[(V|s^* - V|s_i)^2 \cdot (V|s^* - V|s_j)^2] \tag{5.8}$$

5.3 Empirical versus Theoretical Methods

The theoretical 1st order success rates using Fei's algorithm are plotted against the empirical 1st order success rates for different targets/setups of varying SNR

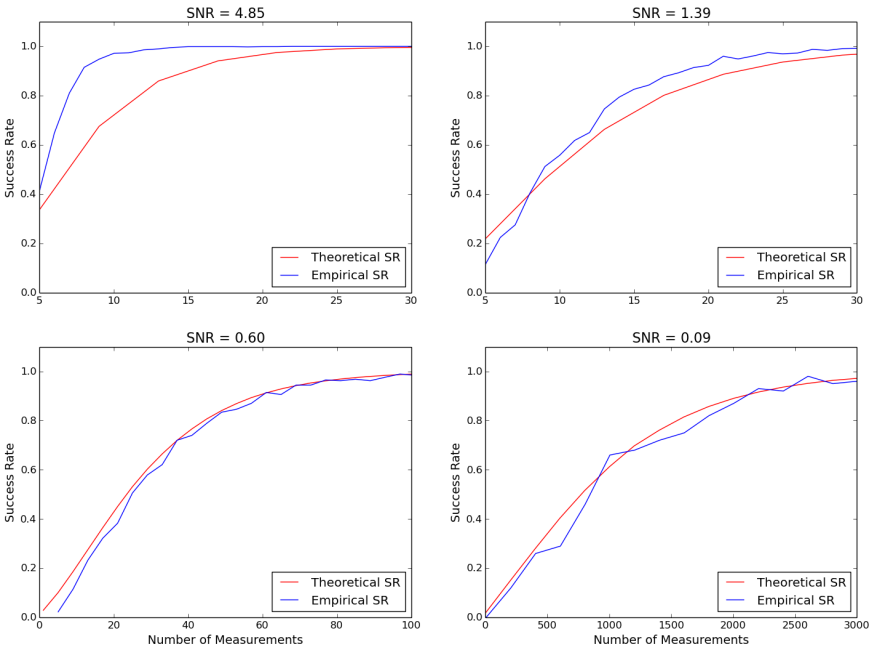

Figure 5.1: Theoretical versus Empirical success rates for different SNR values.

in Figure 5.1. We observe that when the SNR is high, the theoretical success rate underestimates the actual success rate. When the SNR is low, the theoretical success rate tracks the empirical success rates closely.

5.4 Conclusion

In Chapter 4 we have shown that if the evaluator is interested in measuring the relative performance between measurement setups then the SNR metric is the best choice. However, the evaluator may also want to know the practical impact a measurement setup has on the success rate.

In this chapter, we have shown that success rate can be efficiently and accurately estimated through a profiling phase (with Rivian's method) or using only the SNR (with Fei's method). Fei's method requires the leakage model of the cryptographic device to be in the form of Equation (4.6). This is not an issue when evaluating measurement setups as the evaluator can freely select the cryptographic device under test. Moreover, such a linear leakage model with gaussian noise is commonly encountered in many cryptographic devices [FDLZ14].

As such, with the SNR value, the evaluator can still accurately estimate the success rate which is yet another compelling reason to use the SNR metric.

Chapter 6

Experimental Setups and Results

The SNR of the different measurement setups for Training Card 6 (software DES), Training Card 8 (hardware DES) and Pinata board (software DES) are displayed in Table 6.1, 6.2 and 6.3 respectively.

For both smartcards, the most comprehensive experiments are conducted only for Training Card 6 whereby we not only compare the full Inspector and ChipWhisperer setups, but also perform mix and match of both Inspector and ChipWhisperer equipment. For Training Card 8, we only compare the best possible results we can obtain from ChipWhisperer setup versus the Inspector setup.

The Pinata board does not have a measurement point at which we can easily obtain a voltage that is representative of the power consumed during the cryptographic operation. However, we can insert a power measurement circuit in series between the ground voltage of the embedded processor and the ground voltage of the power supply unit. The power measurement circuit we use here is the Inspector current probe and also a 50 Ohm resistor which we use to represent the low budget setup. A 50 Ohm resistor was chosen to match the impedance of OpenADC so as to maximise power transfer.

6.1 Discussion of Results

6.1.1 Power Measurement Circuits

From the table of results, by comparing Experiment Number (EN) 101 with 108, 102 with 107, and 105 with 110 we see that Power Tracer has a better SNR than the smartcard socket of ChipWhisperer's multi-target victim board. Of

Table 6.1: Results for Training Card 6 (Software DES on smartcard)

EN	**Oscilloscope**	**Power Measurement Circuit**	**SNR**	**Signal (mV)**	**Noise (mV)**
101	Lecroy (Sampling at 1.25 GHz)	Power Tracer	4.90	43.82	8.95
102	PicoScope (Sampling at 1GHz)	Power Tracer	4.85	43.62	9.02
103	PicoScope (Sampling at 100MHz)	Power Tracer	4.15	45.50	11.01
104	CW-Capture OpenADC (Synchronous sampling at 4x smartcard clock frequency)	Multi-Target Victim Board (Smartcard Socket)	1.39	22.85	16.44
105	CW-Capure OpenADC (Aynchronous sampling at 100MHz)	Multi-Target Victim Board (Smartcard Socket)	1.93	23.21	12.05
106	CW-Lite OpenADC (Aynchronous sampling at 100MHz)	Power Tracer	0.37	3.45	9.44
107	PicoScope (Sampling at 1GHz)	Multi-Target Victim Board (Smartcard Socket)	3.37	11.55	3.43
108	Lecroy (Sampling at 1.25GHz)	Multi-Target Victim Board (Smartcard Socket)	3.68	10.39	2.83
109	CW-Capture OpenADC (Synchronous sampling at 4x smartcard clock frequency)	Power Tracer	0.29	45.04	157.72
110	CW-Capure OpenADC (Asynchronous sampling at 100MHz)	Power Tracer	2.19	14.02	6.43

course, this is to be expected as the smartcard socket really just provides 8 electrical connections to the smartcard IO pins without additional noise reduction measures.

For the Pinata board, when using the Lecroy, the current probe has a better SNR than the resistor as seen in Experiment Number 301 versus 302. Again, this is expected due to the properties of the current proble explained in Section 3.5.3. However, when using the OpenADC, the resistor produces better results compared to the current probe. We have no conclusive reasons but guess that the input impedance of the OpenADC has some inductive component which conflicts with the current probe.

Table 6.2: Results for Training Card 8 (Hardware DES on Smartcard)

EN	Oscilloscope	Power Measurement Circuit	SNR	Signal (mV)	Noise (mV)
201	Lecroy (Sampling at 1.25 GHz)	Power Tracer	0.090	0.42	4.71
202	CW-Capture OpenADC (Asynchronous sampling at 100MHz)	Multi-Target Victim Board (Smartcard Socket)	0.045	0.33	16.44

Table 6.3: Results for Pinata Board (Software DES on embedded processor)

EN	Oscilloscope	Power Measurement Circuit	SNR	Signal (mV)	Noise (mV)
301	Lecroy (Sampling at 1.25 GHz)	Current Probe	0.94	2.95	3.14
302	Lecroy (Sampling at 1.25 GHz)	50 Ohm Resistor	0.6	0.54	0.90
303	OpenADC (Asynchronous sampling at 100MHz)	50 Ohm Resistor	0.70	12.3	17.33
304	OpenADC (Asynchronous sampling at 100MHz)	Current Probe	0.51	8.48	16.51

6.1.2 Oscilloscope

Lecroy vs PicoScope

By comparing Experiment Number 101 with 102 and 103, and Experiment Number 107 with 108, we see that the Lecroy (sampling at 1.25 GHz) provides a better quality measurement than the Picoscope (sampling at 100 MHz or 1 GHz). As such, for all subsequent targets (Training Card 8 and the Pinata board), we only use the Lecroy with a sampling speed of 1.25 GHz.

Synchronous Sampling

Unexpectedly, the OpenADC sampling synchronously at 4x the smartcard clock frequency (Experiment Number 104) has a lower SNR than the OpenADC sampling asynchronously at 100 MHz (Experiment Number 105). To qualify this result, no phase adjustment was performed for synchronous sampling. As such, it could be because the interesting leakage happens at a specific phase offset from the rising clock edge which was not captured with a 4x synchronous clock without phase adjustment. Better results *might* be obtained by varying the phase of the synchronous sampling clock but we excluded this from our experiment as such micro adjustment would be quite time consuming with the current

tools. Perhaps this could be automated in future versions of the ChipWhisperer software whereby measurements with different phases could be obtained, and the SNR computed to determine the optimal phase adjustment.

Another unexpected result was that the OpenADC could not lock onto the Power Tracer clock which explains the significantly lower SNR when using the OpenADC to sample synchronously with the Power Tracer in experiment 108. Successful synchronous sampling depends on the ability of the FPGA Digital Clock Management (DCM) module to lock onto the target clock. To do so, the clock input needs to meet all the specifications of the DCM module such as the frequency range and jitter. The PowerTracer clock meets these requirements so again, we have no definitive explanation for this result.

ChipWhisperer Lite versus ChipWhisperer Capture

The acquisition of the ChipWhisperer-Lite (Experiment Number 106) has a much poorer performance compared to the ChipWhisperer Capture (Experiment Number 110) with the same power measurement circuit (Power Tracer). This could be due to the poorer production quality of the ChipWhisperer-Lite as it is a pre-production prototype. It could also be caused by design reasons as conducted/radiated noise might arise from co-locating the FPGA, microprocessor and OpenADC on the same PCB. Furthermore, the clock speed of both the FPGA and microprocessor is faster at 96MHz in the ChipWhisperer-Lite compared to 30MHz in the ChipWhisperer Capture.

Acquisition Time

The time required for the acquisition is also an important factor in practical security evaluations. For Experiment Numbers 10x and 20x, it is the smartcard communications that is the bottleneck andnot the oscilloscopes. Acquisitions with the Power Tracer take about 6 hours for 100k traces which is four times faster than with the smartcard module of the ChipWhisperer Capture. For Experiment Numbers 30x, measurements with both the Lecroy and OpenADC take an equivalent amount of time (2 hours for 160k acquisitions) but whether this speed is limited by the oscilloscope acquisition or the Pinata board cannot be determined.

6.2 Target Independence

The Signal to noise ratios in Table 6.1, 6.2 and 6.3 are useful as a relative measure of how one measurement setup compares against another. Nevertheless,

all the SNR results for the different measurement setups are also target dependent as they include the noise values of the target. Ideally, if we are able to characterise the noise of the target, we can isolate the noise introduced by the measurement setup through the relationship:

$$\begin{aligned} \text{Target Noise} &\sim \mathcal{N}(0, \sigma_T^2) \\ \text{Measurement Setup Noise} &\sim \mathcal{N}(0, \sigma_{MS}^2) \\ \text{Total Noise} &\sim \mathcal{N}(0, \sigma_{MS}^2 + \sigma_T^2) \end{aligned} \tag{6.1}$$

Unfortunately, the value of the target's noise is difficult to characterise and is further compounded by the fact that the different measurement setups have different values for 'signal' due to the different power measurement circuits, and that the OpenADC performs voltage amplification. To simplistically address this problem, we scale the signal and noise figures for Training Card 6 and Training Card 8 so that all setups have the same signal as the Lecroy and Power Tracer setup. We then solve several linear equations to determine the target and measurement setup noise. This procedure depends on the following assumptions which may not hold true in practice:

Assumption-1 Noise scales linearly with the Signal

Assumption-2 Noise introduced by a measurement setup is constant even across different targets

Nevertheless we perform this exercise to illustrate that with the noise of the different measurement setups, we can calculate the theoretical success rate for any signal and noise values of the hypothetical target. The normalised signal and noise figures for Training Card 6 and Training Card 8 are shown in Table 6.4. Using these figures, we are able to estimate that the noise of Training Card 6 and Training Card 8 are 2 mV and 7 mV respectively, and the noise of the Inspector setup (Lecroy and Power Tracer) is 9mV, and the noise of the ChipWhisperer setup (OpenADC and smartcard socket) is 22.5mV.

We simulate the success rates for four hypothetical targets with signal to noise ratios given in Figure 6.1 for both the Inspector and ChipWhisperer setups. In the top two plots, we see that as the target's signal decreases from 0.35mV to 1mV while it's noise remains constant at 7mV, the relative shapes of the success rate curve hardly change and it is only the x-scale which increases. On the other hand, if the target's signal is kept constant at 1mV and the noise increased from 7mV to 14mV and 21mV, we observe that the distance between the measurement's setups success rates reduces.

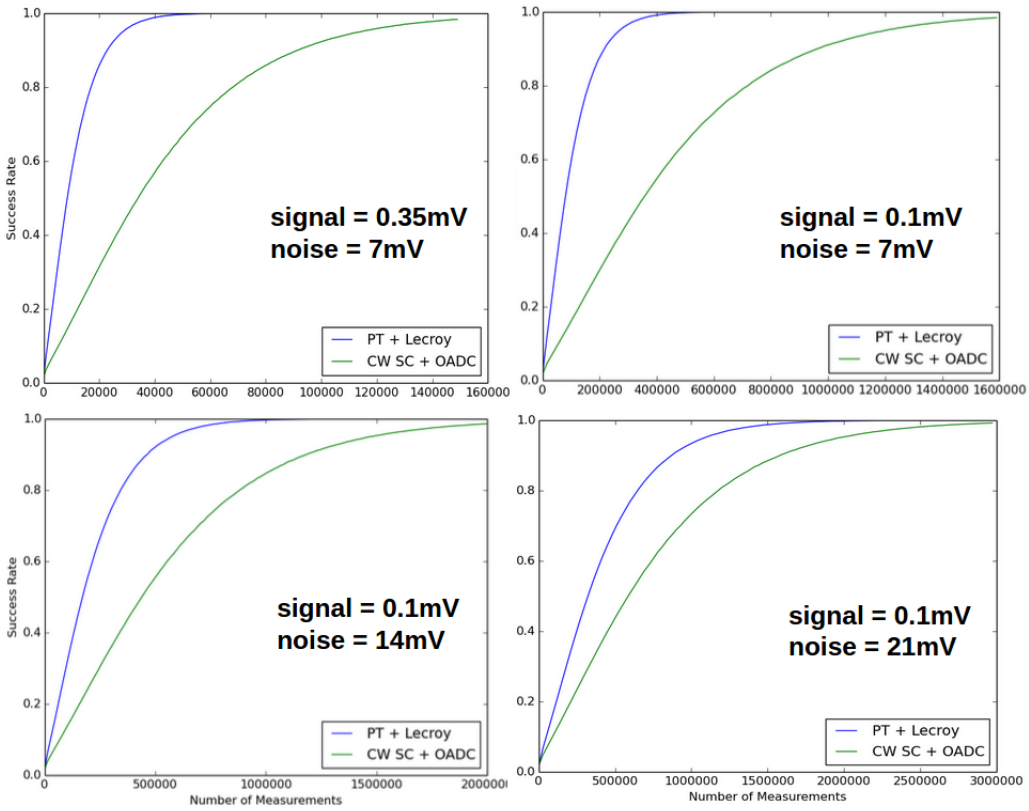

Figure 6.1: Simulated success rates for various hypothetical targets.

Table 6.4: Normalised signal and noise figures. The SNR from the original experiment numbers remain unchanged.

EN	**Oscilloscope**	**Power Measurement Circuit**	**Target**	**Signal (mV)**	**Noise (mV)**
101	Lecroy (Sampling at 1.25 GHz)	PowerTracer	Training Card 6	43.82	8.95
105	CW-Capture OpenADC (Asynchronous sampling at 100MHz)	Multi-Target Victim Board (Smartcard Socket)	Training Card 6	43.82	22.75
303	Lecroy (Sampling at 1.25 GHz)	Power Tracer	Training Card 8	0.42	4.71
304	CW-Capture OpenADC (Asynchronous sampling at 100MHz)	Multi-Target Victim Board (Smartcard Socket)	Training Card 8	0.42	20.92

This phenomenon is due to the quadratic relationship expressed in Equation (6.1). When the target has a low noise, then the noise from the measurement setup dominates the total noise and vice versa. This leads to the observation that when comparing measurement setups, a target with low noise is more suitable as small differences between measurement setups become more marked. This can also be empirically verified by observing Table 6.1 and 6.2.

Chapter 7

Conclusions and further work

We have compared many comparison metrics that have been proposed in academia and concluded that the signal to noise ratio metric is the most suitable for the comparison of SCA measurement setups as it is best able to detect small quality differences and requires the least number of leakages and computational time. When using the SNR metric, the cryptographic device used as the target should fulfil the linear leakage characteristic expressed in Equation (4.6) and preferably have as little noise as possible so that the difference between the measurement setups can be best discerned. This comparison method will be useful for SCA attackers and SCA tool designers to quantitatively assesses and optimise their measurement setups. Furthermore, if the target's noise characteristics are known, we can isolate the noise due to the measurement setup. This is useful because we can then calculate the effect of the measurement setup noise on the theoretical success rate for any signal and noise values of a hypothetical target.

We have also compared the ChipWhisperer measurement setup against a high end commercial setup and as expected, the ChipWhisperer introduces more noise than the commercial setup. The ChipWhisperer might also be unsuitable for mounting a successful attack on many modern state of the art cryptographic devices due to its physical limitations on the number of samples per acquisition, low sampling speed and low voltage range. It is also unsuitable for use when the reproducibility of results is an important criteria (e.g. for security laboratories) as the ChipWhisperer sometimes produces results that are difficult to explain and counter-intuitive. Engineering time also has to be devoted before one can effectively use the ChipWhisperer. Nevertheless, when the ChipWhisperer (or

other low budget SCA measurement setups) meets a target that is within its specifications, it actually has a good enough performance. In such a situation, the biggest factor for a successful attack is the attackers ability to perceive the SCA signal (through the noise) using appropriate SCA algorithms and not the measurement setup itself. This opens the opportunity to an attacker with a low budget to still crack a state-of-the-art device when using a correct attack approach.

In conclusion, accessible and low budget tools such as the ChipWhisperer are potentially capable of mounting very practical SCA attacks. The ChipWhisperer represents the dawn of this low budget and open source movement and as interest in SCA grows, the hardware will undoubtedly improve. At the moment, mounting an attack with the ChipWhisperer still requires considerable technical knowledge of both SCA and the target device and is thus out of the reach of the typical 'script kiddy'. However, as the ChipWhisperer community [1] grows, new skilled entrants will be attracted. The SCA software will become more polished and unethical individuals might even post detailed SCA tutorials detailing the procedure to break specific commercial products. Inevitably, the SCA threat landscape will be revolutionised as SCA becomes as mainstream as computer hacking. Security designers should take precautionary countermeasures before this happens.

[1] Or any other similar community built around low budget and standardised SCA hardware

7.1 Further work

The noise figure of measurement equipment is typically given over a range of frequencies. Future work could study the impact of measurement setup noise at different leakage frequencies, or the impact of other factors such as device power. It was also stated in Section 6.2 that to isolate the noise from the measurement setup, it was first necessary to know the noise from the cryptographic device. A set of reference cryptographic devices with known leakage frequencies and noise figures could be designed. These could be used to universally rate different measurement setups.

Appendix A

Porting code to a different ZTEX FPGA module

The detailed process to port the microprocessor and FPGA code from the ZTEX FPGA module 1.11c to the ZTEX FPGA module 1.15a is documented below. The ZTEX FPGA module 1.15a has also been made obsolete and should now be replaced by the versions 2.13 and 2.16. The same process applies for porting the code to these newer versions.

- Download the project from `https://www.assembla.com/code/chipwhisperer/git/nodes/master`
- Generate the Xilinx (FPGA) project using: <Parent_Directory>\hardware\capture\chipwhisperer-rev2\hdl\makeprojects_win.bat. The relevant FPGA project file to modify from here on is <Parent_Directory>\hardware\capture\chipwhisperer-rev2\hdl\ztex_rev2_1.11c_ise \ztex_rev2_1.11c_ise.xise. Change the FPGA hardware version to the appropriate version e.g. XC6SLX45 if using a Spartan 6 LX45. [1]
- The microcontroller code to modify is located in <Parent_Directory>\hardware\capture\chipwhisperer-rev2\ ezusb-firmware\ztex-sdk\examples\usb-fpga-1.11\1.11c\openadc\OpenADC.c. Change the value of the special function registers to represent the correct IO pin connections to the FPGA.

[1] Make sure that the .ucf file from <Parent_Directory>\hardware\capture\chipwhisperer-rev2\hdl\ztex_rev2_1.11c_ise\ztex_1_11.ucf is used instead of <Parent_Directory>\openadc\hdl\example_targets\ztex_11.1c_ise\ztex_1_11.ucf

- Remap the internal IO pin connections between the FPGA and microcontroller by referencing `http://www.ztex.de/downloads/usb-fpga-1.11.pdf` and `http://www.ztex.de/downloads/usb-fpga-1.15.pdf`. These IO pins route the microcontroller clock, and USB FIFO to the FPGA. In this process, the FPGA file (ztex_1_11.ucf) file and microcontroller file (OpenADC.c) need to be changed.

- Remap the external IO pin connections from the FPGA to the ChipWhisperer Carrier board by referencing `http://www.ztex.de/downloads/usb-fpga-1.xls`. These pins provide the IO connections to the OpenADC and the cryptographic targets. Only the FPGA file (ztex_1_11.ucf) file needs to be changed.

- Generate the new microcontroller firmware and FPGA bitstream. These are now ready to be downloaded to the ChipWhisperer hardware.

- The modified FPGA (ztex_1_11.ucf) file and microcontroller file (OpenADC.c) fileare provided for reference at `https://github.com/alvincai/chipwhisperer-explorations/`

The discussions with Colin (the ChipWhisperer developer) about this process is also documented in the ChipWhisperer mailing list at `http://mail.newae.com/pipermail/chipwhisperer-talk_newae.com/2014-November/thread.html#33`.

Bibliography

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In *Cryptographic Hardware and Embedded Systems-CHES 2004*, pages 16–29. Springer, 2004.

[BV11] Riscure BV. Inspector datasheet - Power Tracer. `https://www.riscure.com/documents/datasheet_powertracer4.pdf?1376577653`, 2011.

[BV15a] Riscure BV. Inspector datasheet - Current Probe. `https://www.riscure.com/documents/datasheet_currentprobe1c1.pdf?1425390987`, 2015.

[BV15b] Riscure BV. Inspector SCA. `https://www.riscure.com/security-tools/inspector-sca/`, 2015.

[CRR03] Suresh Chari, Josyula R Rao, and Pankaj Rohatgi. Template attacks. In *Cryptographic Hardware and Embedded Systems-CHES 2002*, pages 13–28. Springer, 2003.

[DO09] Abid Uveys Danis and Berna Ors. Differential power analysis attack considering decoupling capacitance effect. In *Circuit Theory and Design, 2009. ECCTD 2009. European Conference on*, pages 359–362. IEEE, 2009.

[dpa] DPA contest V4 frequently asked questions. `http://www.dpacontest.org/v2/rules.php`. Accessed: 2015-05-11.

[DZFL14] A Adam Ding, Liwei Zhang, Yunsi Fei, and Pei Luo. A statistical model for higher order DPA on masked devices. In *Cryptographic Hardware and Embedded Systems–CHES 2014*, pages 147–169. Springer, 2014.

[FDLZ14] Yunsi Fei, A Adam Ding, Jian Lao, and Liwei Zhang. A statistics-based fundamental model for side-channel attack analysis. *IACR Cryptology ePrint Archive*, 2014:152, 2014.

[FLD12] Yunsi Fei, Qiasi Luo, and A Adam Ding. A statistical model for DPA with novel algorithmic confusion analysis. In *Cryptographic Hardware and Embedded Systems–CHES 2012*, pages 233–250. Springer, 2012.

[GBTP08] Benedikt Gierlichs, Lejla Batina, Pim Tuyls, and Bart Preneel. Mutual information analysis. In *Cryptographic Hardware and Embedded Systems–CHES 2008*, pages 426–442. Springer, 2008.

[GMS+11] Sylvain Guilley, Houssem Maghrebi, Youssef Souissi, Laurent Sauvage, J Danger, and Secure-IC SAS. Quantifying the quality of side-channel acquisition. *COSADE, February*, 2011.

[HRG14] Annelie Heuser, Olivier Rioul, and Sylvain Guilley. A theoretical study of Kolmogorov-Smirnov distinguishers. In *Constructive Side-Channel Analysis and Secure Design*, pages 9–28. Springer, 2014.

[kic] Kickstarter for ChipWhisperer-Lite. `https://www.kickstarter.com/projects/coflynn/chipwhisperer-lite-a-new-era-of-hardware-security/`. Accessed: 2015-05-11.

[KJJ99] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Advances in CryptologyCRYPTO99*, pages 388–397. Springer, 1999.

[Koc96] Paul C Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *Advances in CryptologyCRYPTO96*, pages 104–113. Springer, 1996.

[LPR+14] Victor Lomné, Emmanuel Prouff, Matthieu Rivain, Thomas Roche, and Adrian Thillard. How to estimate the success rate of higher-order side-channel attacks. In *Cryptographic Hardware and Embedded Systems–CHES 2014*, pages 35–54. Springer, 2014.

[Man04] Stefan Mangard. Hardware countermeasures against DPA–a statistical analysis of their effectiveness. In *Topics in Cryptology–CT-RSA 2004*, pages 222–235. Springer, 2004.

[MDS99] Thomas S Messerges, Ezzy A Dabbish, and Robert H Sloan. Investigations of power analysis attacks on smartcards. In *USENIX workshop on Smartcard Technology*, volume 17, page 17, 1999.

[MOP08] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power analysis attacks: Revealing the secrets of smart cards*, volume 31. Springer Science & Business Media, 2008.

[MR04] Silvio Micali and Leonid Reyzin. Physically observable cryptography. In *Theory of Cryptography*, pages 278–296. Springer, 2004.

[new] ChipWhisperer based products - all you need for side channel power analysis, glitch attacks, etc. `http://store.newae.com/embedded-security/`. Accessed: 2015-05-11.

[NSA14] NSA. TEMPEST documents, online collection. *http://cryptome.org/nsa-tempest.htm*, 2014.

[OC13a] Colin O'Flynn and Zhizhang Chen. A case study of side-channel analysis using decoupling capacitor power measurement with the OpenADC. In *Foundations and Practice of Security*, pages 341–356. Springer, 2013.

[OC13b] Colin O'Flynn and Zhizhang Chen. Synchronous sampling and clock recovery of internal oscillators for side channel analysis and fault injection. *Journal of Cryptographic Engineering*, 5(1):53–69, 2013.

[OC14] Colin O'Flynn and Zhizhang David Chen. ChipWhisperer: An open-source platform for hardware embedded security research. In *Constructive Side-Channel Analysis and Secure Design*, pages 243–260. Springer, 2014.

[PPE+14] Stjepan Picek, Kostas Papagiannopoulos, Barış Ege, Lejla Batina, and Domagoj Jakobovic. Confused by confusion: Systematic evaluation of DPA resistance of various s-boxes. In *Progress in Cryptology–INDOCRYPT 2014*, pages 374–390. Springer, 2014.

[Res15] Cryptography Research. DPA workstation overview. `http://www.cryptography.com/technology/dpa-workstation.html`, 2015.

[Riv09] Matthieu Rivain. On the exact success rate of side channel analysis in the gaussian model. In *Selected Areas in Cryptography*, pages 165–183. Springer, 2009.

[RO05] Christian Rechberger and Elisabeth Oswald. Practical template attacks. In *Information Security Applications*, pages 440–456. Springer, 2005.

[RSVC+11] Mathieu Renauld, François-Xavier Standaert, Nicolas Veyrat-Charvillon, Dina Kamel, and Denis Flandre. A formal study of power variability issues and side-channel attacks for nanoscale devices. In *Advances in Cryptology–EUROCRYPT 2011*, pages 109–128. Springer, 2011.

[SMY09] François-Xavier Standaert, Tal G Malkin, and Moti Yung. A unified framework for the analysis of side-channel key recovery attacks. In *Advances in Cryptology-EUROCRYPT 2009*, pages 443–461. Springer, 2009.

[TPR13] Adrian Thillard, Emmanuel Prouff, and Thomas Roche. Success through confidence: Evaluating the effectiveness of a side-channel attack. In *Cryptographic Hardware and Embedded Systems-CHES 2013*, pages 21–36. Springer, 2013.

[VCS09] Nicolas Veyrat-Charvillon and François-Xavier Standaert. Mutual information analysis: how, when and why? In *Cryptographic Hardware and Embedded Systems-CHES 2009*, pages 429–443. Springer, 2009.

[vWWB11] Jasper GJ van Woudenberg, Marc F Witteman, and Bram Bakker. Improving differential power analysis by elastic alignment. In *Topics in Cryptology–CT-RSA 2011*, pages 104–119. Springer, 2011.

[Wit02] Marc Witteman. Advances in smartcard security. *Information Security Bulletin*, 7(2002):11–22, 2002.

[WO11a] Carolyn Whitnall and Elisabeth Oswald. A comprehensive evaluation of mutual information analysis using a fair evaluation framework. In *Advances in Cryptology–CRYPTO 2011*, pages 316–334. Springer, 2011.

[WO11b] Carolyn Whitnall and Elisabeth Oswald. A fair evaluation framework for comparing side-channel distinguishers. *Journal of Cryptographic Engineering*, 1(2):145–160, 2011.