Hardware Countermeasures against DPA – A Statistical Analysis of Their Effectiveness

Stefan Mangard*

Institute for Applied Information Processing and Communications
Graz University of Technology
Inffeldgasse 16a, A-8010 Graz, Austria
Stefan.Mangard@iaik.at
http://www.iaik.at/research/sca-lab

Abstract. Many hardware countermeasures against differential power analysis (DPA) attacks have been developed during the last years. Designers of cryptographic devices using such countermeasures to protect their devices have the challenging task to select and implement a suitable combination of countermeasures. Every device has different requirements, and so there is no universal solution to protect devices against DPA attacks.
In this article, a statistical approach is pursued to determine the effect of hardware countermeasures on the number of samples needed in DPA attacks. This approach results in a calculation method that enables designers to assess the resistance of their devices against DPA attacks throughout the design process. This way, different combinations of countermeasures can be easily compared and costly design iterations can be avoided.

Keywords: Smart cards, Side-Channel Attacks, Differential Power analysis (DPA), Hardware countermeasures

1 Introduction

During the last years, a lot of effort has been dedicated towards the research of side-channel attacks [1,9,10] and the development of corresponding countermeasures. In particular, there have been many endeavors to develop effective countermeasures against differential power analysis (DPA) [10,15] attacks.

DPA attacks are based on the fact that the power consumption of a cryptographic device depends on the internally used secret key. Since this property can be exploited with relatively cheap equipment, DPA attacks pose a serious practical threat to cryptographic devices, like smart cards.

The countermeasures that have been developed up till now against these attacks can be categorized into two groups. The first group are the so-called

* This work has been supported by the Austrian Science Fund (FWF Project No. P16110-N04).

T. Okamoto (Ed.): CT-RSA 2004, LNCS 2964, pp. 222–235, 2004.

algorithmic countermeasures [4,5,7,14,22]. The basic idea of these countermeasures is to randomize the intermediate results that are processed during the execution of a cryptographic algorithm. Classical first-order DPA attacks are rendered practically impossible, if this randomization is implemented correctly. However, there are two significant drawbacks of this approach.

The first one is that the randomization is quite expensive to implement for non-linear operations as they are used in symmetric ciphers (see for example [5], [6] and [7]). The second one is that many algorithmic countermeasures do not provide sufficient protection against higher-order DPA attacks [13] or sophisticated SPA attacks [11,18]. The consequence of these facts is that algorithmic countermeasures are typically combined with hardware countermeasures [2,8,12, 16,19,20,21].

The hardware approach to counteract DPA attacks differs significantly from the algorithmic one. The intermediate results that occur during the execution of a cryptographic algorithm are not affected by this type of countermeasure. Instead, the goal of this approach is to bury the attackable part of the power consumption in different kinds of noise.

The more noise there is in the power traces recorded by the attacker, the more measurements are needed for a successful DPA attack. Although the basic idea is relatively simple, hardware countermeasures have proven to be quite effective in practice. This is why cryptographic devices are typically either protected by a combination of hardware and algorithmic countermeasures or solely by hardware countermeasures.

The decision which combination of countermeasures is implemented in a device, is made by the designers. It is their task to choose a combination of countermeasures that provides the resistance against DPA attacks that is necessary for the planned application of the device. The resistance against DPA attacks is typically specified by a number of samples: If DPA attacks with this number of samples fail, the device is resistant enough. Otherwise the requirements are not fulfilled.

Choosing a suitable combination of countermeasures is a very challenging task in practice. This is due to the fact that this decision needs to be made at a very early stage of the design process. Design iterations are costly and so the fabrication of a physical prototype to test whether a combination of countermeasures is sufficient or not, should be avoided.

In order to minimize the number of design iterations, methods are necessary to assess the effect of countermeasures on the number of samples. However, particularly for hardware countermeasures there are no publications that discuss such methods. Publications of hardware countermeasures usually just contain case studies showing that the proposed countermeasure really increases the number of samples. Yet, such case studies are only of limited use for a designer of a device who uses a different technology, a different architecture, and potentially even uses multiple countermeasures simultaneously.

In this article, a statistical approach is pursued to determine the effect of hardware countermeasures on the number of samples. This approach leads to a

calculation method that allows the determination of lower bounds for the number of samples needed in DPA attacks. The presented calculation method is based on only very few parameters that can be assessed already at an early stage of the design process.

It is therefore ideally suited to help designers to choose the right combination of countermeasures already at the beginning of the design process. Of course, the presented calculation method can also be used at any time during the design process to determine whether a design fulfills certain resistance requirements or not. The more precisely the parameters of the calculation can be determined, the more precise becomes the statement on the number of samples.

This article is organized as follows: Section 2 provides a short summary of the fundamentals of DPA attacks and defines some of the notation that is used in this article. Section 3 analyzes the principles that are used by hardware countermeasures to increase the resistance against DPA attacks. The calculation of lower bounds for the number of samples is presented in section 4. In section 5, the corresponding formulas are empirically verified. Conclusions can be found in section 6.

2 Differential Power Analysis

The power consumption of a digital circuit depends on the data that the circuit processes. Thus, the power trace of a device executing a cryptographic algorithm, depends on intermediate results of this algorithm.

DPA attacks exploit the fact that in all cryptographic algorithms there occur intermediate results which are a function of the ciphertext and only few key bits. We call these key bits a subkey. In a DPA attack, one subkey after the other is attacked until the entire secret key is known or the missing rest of the key can be efficiently determined by a brute-force search. An attacker knowing the cryptographic algorithm that is executed in a device, can reveal a subkey as follows:

First, the power consumption of the device is recorded, while it encrypts S different plaintexts using the same key. In this article, we use the common assumption that these plaintexts are uniformly distributed. We refer to the power traces that are recorded during the encryptions as $P_{1...S,1...T}$, where T is the number of points that are recorded per encryption.

In the next step, the attacker chooses an intermediate result of the executed algorithm that is a function of the ciphertext and a short subkey. Based on the ciphertext and all possible values for the subkey, hypothetical values for the intermediate result are calculated. This leads to a matrix $I_{1...K,1...S}$ of hypothetical intermediate results, where K is the number of possible values for the subkey.

The subkey k_c that is actually used in the attacked device is one of the K possible values for the subkey. Hence, the values $I_{k_c,1...S}$ have actually been processed by the attacked device while it has been doing the S recorded encryptions. Consequently, the values $P_{1...S,t_c}$ depend on $I_{k_c,1...S}$, where t_c is the moment of time at which the attacked intermediate results have been processed.

The attacker determines a hypothetical power consumption value $H_{k,s}$ for every $I_{k,s}$. The absolute values of $H_{1...K,1...S}$ are of no importance for the attack—only the relative distance between the values is relevant.

Nevertheless, the calculation of $H_{1...K,1...S}$ requires some basic knowledge about how the processing of different data affects the power consumption of a device. Many devices use pre-charged buses. Such buses cause a power consumption that is proportional to the Hamming weight of the data block that is being transferred over the bus.

After having determined $H_{1...K,1...S}$, the attacker reveals the correct subkey k_c by correlating the hypothetical power consumptions with the one of the device. In this article, the Pearson correlation coefficient is used to measure this correlation.

In [10], Kocher et. al. measure this correlation by calculating the distance between means. In the context of DPA attacks, there is no significant difference between the two measures for the correlation. However, we favor the Pearson correlation coefficient because there exists a well-established theory on measuring correlations this way—the Pearson correlation coefficient is the common measure to determine the linear relationship between two variables. Equation 1 shows a definition of the correlation ρ between two variables X and Y, where $E(X)$, $E(Y)$ and $E(XY)$ are expected values, $Cov(X,Y)$ is the covariance and $Var(X)$ as well as $Var(Y)$ are the variances of the variables.

$$\rho(X,Y) = \frac{E(XY) - E(X)E(Y)}{\sqrt{Var(X)Var(Y)}} \frac{Cov(X,Y)}{\sqrt{Var(X)Var(Y)}} \tag{1}$$

The definition of the Pearson correlation coefficient r is shown in equation 2. r estimates the correlation ρ between two variables based on S samples. $\bar{x}$ and $\bar{y}$ in equation 2 denote the means of the variables based on S samples.

$$r(< x_1, \ldots, x_S >, < y_1, \ldots, y_S >) = \frac{\sum_{s=1}^{S} (x_s - \bar{x})(y_s - \bar{y})}{\sqrt{\sum_{s=1}^{S} (x_s - \bar{x})^2} \sqrt{\sum_{s=1}^{S} (y_s - \bar{y})^2}} \tag{2}$$

In a DPA attack, the Pearson correlation coefficient between the values $H_{k=fixed,1...S}$ and $P_{1...S,t=fixed}$ is calculated for every fixed k and t. This leads to the matrix $\mathcal{R} = r_{1...K,1...T}$ of correlation coefficients. Since the values $P_{1...S,\forall t \neq t_c}$ and $H_{\forall k \neq k_c,1...S}$ are largely uncorrelated, the correlations $\rho_{\forall k \neq k_c, \forall t \neq t_c}$ are significantly lower than ρ_{k_c,t_c}.

If S is sufficiently large in an attack, this difference between the correlations can be detected in the matrix $\mathcal{R}$ of Pearson correlation coefficients. In this case, one correlation coefficient of $\mathcal{R}$ is significantly larger than all other ones. The position of this peak in $\mathcal{R}$ reveals the correct subkey k_c.

The number of samples that is needed in a DPA attack to reveal k_c is mainly determined by the value ρ_{k_c,t_c}. This observation has already been made previously by Messerges et. al. in [15].

Since ρ_{k_c,t_c} is the maximum value of $\rho_{1...K,1...T}$, we refer to this correlation as ρ_{max} throughout the remainder of this article. The higher ρ_{max} is, the less samples are needed to see a significant peak at the position (k_c, t_c) of $\mathcal{R}$.

This is why it is the goal of hardware countermeasures to reduce ρ_{max} to a value that is as close to zero as possible.

3 Hardware Countermeasures

In order to increase the number of samples needed in DPA attacks, hardware countermeasures decrease the correlation between the hypothetical power consumptions and the power consumption of the device.

The hypothetical power consumptions are determined by the attacker, and therefore they cannot be controlled by the designers of a device. Yet, designers can alter the power consumption of their devices in such a way that ρ_{max} is reduced. There exist two possibilities to lower this correlation. All hardware countermeasures that have been proposed so far, rely on these two possibilities.

3.1 Reduction of the SNR

The first possibility to reduce the correlation ρ_{max} is to bury the part of the power consumption that is caused by the processing of the attacked intermediate result in a lot of noise.

The burying of this signal in noise is best measured by a signal-to-noise ratio (SNR). For the definition of this SNR, we define Q to be the power consumption caused by the attacked intermediate result and N to be additive noise. Consequently, the power consumption of a device at the time t_c can be written as $P_{s,t_c} = Q_s + N_s$.

Equation 3 shows the definition of the SNR for the signal Q. Since the DC components of N and Q are not relevant for the calculation of the correlation, only the AC components (i.e. the variances) of the signals are considered in this equation.

$$SNR = \frac{Var(Q)}{Var(N)} \tag{3}$$

The lower the SNR is, the lower is also the correlation between the correct hypothetical power consumption and the power consumption of the device.

There are several hardware countermeasures that reduce the SNR. The most prominent examples are special logic styles that minimize the data dependency of the power consumption. Such logic styles are presented by Moore et. al. in [16, 17], by Tiri et. al. in [20,21] and by Saputra et. al. in [19].

However, there are many more ways to reduce the SNR. For example, also flattening the power consumption or random charging of on-chip capacitances reduce the SNR. In fact, any processing that occurs in parallel to the execution of the cryptographic algorithm, leads to this result.

In general, the effect of a hardware countermeasure on the SNR can be determined already at an early stage of the design process. Even before the implementation of a device has started, it is possible to assess the SNR. During the implementation phase, the SNR can be determined by using tools that assess the power consumption of a device. Due to the fact that the overall power consumption of integrated circuits has become increasingly important during the last few years, several tools of this kind are available.

3.2 Random Disarrangement of t_c

The second possibility to reduce the correlation ρ_{max} is to randomly disarrange the moment of time at which the attacked intermediate result is processed. If the time t_c is different in every power trace, the correlation between the correct hypothetical power consumption and the one of the device is significantly reduced.

Random disarrangement techniques lead to the fact that there is a certain probability distribution for t_c. Clearly, the highest correlation in DPA attacks occurs at the moment of time of the power traces, where the maximum of this probability distribution is located. We refer to this moment of time as $\hat{t}_c$. The maximum probability $\hat{p}$ that is located at $\hat{t}_c$ is the decisive value determining how much the correlation is reduced in DPA attacks. The lower $\hat{p}$ is, the more samples are required in DPA attacks.

There exist many proposals for hardware countermeasures that are based on a random disarrangement of t_c. The classic countermeasure that is based on this principle is the insertion of random delays [3], which can even be implemented in software. Another approach that is also based randomizing t_c is pursued by Irwin et. al. in [8] and by May et. al. in [12]. They propose to use a non-deterministic processor to foil DPA attacks.

The countermeasure proposed by Benini et. al. in [2], also gains most of its strength by randomizing t_c. Of course, also asynchronous logic styles [16,17] are very well suited for the insertion of non-deterministic delays.

In order to determine the effect of a random disarrangement of t_c on DPA attacks early in the design process, it is necessary that $\hat{p}$ can be determined very early.

In case of the insertion of random delays, t_c is binomially distributed and so $\hat{p}$ can be calculated in a straightforward manner. In case of the other countermeasures, the distributions of t_c may be more complex. Yet, even if a direct calculation of $\hat{p}$ is not practical, it is always possible to approximate it empirically based on a software model of the countermeasure.

In this section, we have introduced the possibilities that can be used to lower the correlation ρ_{max} in DPA attacks. There are two properties of (combinations of) hardware countermeasures that largely determine the effect of the countermeasures on the number of samples: the SNR defined in equation 3 and $\hat{p}$.

Both properties can be assessed already at an early stage of the design process. The following section introduces the calculation of lower bounds for the number of samples based on these two parameters.

4 Calculation of Lower Bounds for the Number of Samples

The effect of hardware countermeasures on the number of samples is largely determined by the two parameters discussed in section 3. However, there are also some other parameters that have a certain influence of the number of samples. Throughout this article, these parameters are set to worst-case values from a designer's point of view (i.e. all unknown parameters are set in favor of a potential attacker). Hence, the calculation method introduced in this section, leads to lower bounds for the number of samples. This conservative measure is exactly what designers should use to determine the effectiveness of the hardware countermeasures in their design.

In the following subsection, first formulas are derived to calculate ρ_{max} in the presence of hardware countermeasures. Subsection 4.2 then introduces the calculation of lower bounds for the number of samples based on ρ_{max}.

4.1 ρ_{max} in the Presence of Hardware Countermeasures

The Effect of SNR on ρ_{max}: In a DPA attack on a device without random disarrangement of t_c, ρ_{max} is the correlation between the hypothetical power consumption for the correct subkey and the one of the device at the time t_c.

Equation 4 shows the calculation of ρ_{max} based on SNR. In this equation, the variable H refers to the hypothetical power consumption for the correct subkey. Q and N are used as defined in section 3: Q denotes the power consumption of the device caused by the attacked intermediate result and N denotes uncorrelated additive noise.

$$\begin{aligned}\rho(H, Q+N) &= \frac{E(H(Q+N)) - E(H)E(Q+N)}{\sqrt{Var(H)(Var(Q)+Var(N))}} \\ &= \frac{E(HQ+HN) - E(H)(E(Q)+E(N))}{\sqrt{Var(H)Var(Q)}\sqrt{1+\frac{Var(N)}{Var(Q)}}} = \frac{\rho(H,Q)}{\sqrt{1+\frac{1}{SNR}}} \end{aligned} \quad (4)$$

The Effect of $\hat{p}$ on ρ_{max}: If t_c is randomly disarranged, the correlation ρ_{max} occurs between the correct hypothetical power consumption and the one of the device at the time $\hat{t}_c$.

In equation 5, the variable $\hat{P}$ refers to the power consumption of the device at this time $\hat{t}_c$. The probability that a power consumption at this time is caused by the processing of an attacked intermediate result is $\hat{p}$. With a probability of $(1-\hat{p})$, the power consumption at the time t_c is caused by the processing of some other data.

In equation 5, we refer to the power consumption caused by an attacked intermediate result as P. With O we refer to the one caused by the processing of other data. In practice, O is largely independent from the correct hypothetical power consumption H. This is why we set $Cov(H, O)$ to zero in equation 5.

$$\rho(H,\hat{P}) = \frac{\hat{p} * Cov(H,P) + (1-\hat{p})\,Cov(H,O)}{\sqrt{Var(H)Var(\hat{P})}}$$
$$= \frac{\hat{p} * Cov(H,P)}{\sqrt{Var(H)Var(\hat{P})}} = \rho(H,P) * \hat{p} * \sqrt{\frac{Var(P)}{Var(\hat{P})}} \quad (5)$$

Calculation of ρ_{max}: The equations 4 and 5 can be combined into one formula (see equation 6) that allows to determine the effect of a given combination of hardware countermeasures on ρ_{max}.

$$\rho_{max} = \frac{\rho(H,Q)}{\sqrt{1+\frac{1}{SNR}}} * \hat{p} * \sqrt{\frac{Var(P)}{Var(\hat{P})}} \quad (6)$$

Besides the parameters SNR and $\hat{p}$, also the correlation $\rho(H,Q)$ and the term $F = \sqrt{\frac{Var(P)}{Var(\hat{P})}}$ influence ρ_{max}. While the correlation $\rho(H,Q)$ solely depends on how well the attacker knows the power consumption characteristics of a device, the factor F is a device-specific property.

However, unlike SNR and $\hat{p}$, F is rather difficult to assess at very early stages of the design process. In order to reasonably assess F, designers need some knowledge about how the power consumption of the device looks like before and after the attacked intermediate result is processed. The range that needs to be known is the bigger, the wider the probability distribution of t_c is.

In practice, F should be set to the worst-case value 1 at the very early stages of the design process. As soon as first assessments on the power consumption of the device are available, F can be updated accordingly in the calculation of the number of samples.

Since designers should always determine the number of samples in a conservative manner, $\rho(H,Q)$ should be set to 1 throughout the design process.

Based on equation 6, ρ_{max} can be determined at any point of the design process. The better the parameters of this equation can be assessed, the better becomes the statement on the number of needed samples.

4.2 Mapping ρ_{max} to a Number of Samples

The number of samples needed in a DPA attack is the commonly used measure for the resistance of a device against these attacks. In order to reveal the correct subkey k_c, the number of samples needs to be increased in an attack until a significant peak is visible in the matrix $\mathcal{R}$.

The Pearson correlation coefficients in this matrix $\mathcal{R}$ estimate the correlations $\rho_{1\ldots K,1\ldots T}$ based on S samples. The sampling distribution of a Pearson correlation coefficient r is best described by transforming r to a variable z that is

normally distributed. This transformation (known as Fisher's Z-Transformation) is shown in the equations 7 to 9.

$$z = \frac{1}{2} \ln \frac{1+r}{1-r} \tag{7}$$

$$\mu = \frac{1}{2} \ln \frac{1+\rho}{1-\rho} \tag{8}$$

$$\sigma^2 = \frac{1}{S-3} \tag{9}$$

Based on these formulas, the sampling distribution of each correlation coefficient r of the matrix $\mathcal{R}$ can be determined easily based on $\rho_{1...K,1...T}$ and S. The equations 7 to 9 are an approximation for $S > 30$. Yet, since the number of samples is typically much higher, this approximation is sufficient.

Calculating the exact number of samples that are needed for a DPA attack is quite difficult in practice. This has several reasons. First of all, the designers of a cryptographic device don't know to which sampling rate an attacker will set the oscilloscope in an attack, and the designers also don't know how long the recorded power traces will be. Clearly, these parameters strongly influence the size and the values of the matrix $\rho_{1...K,1...T}$.

Even if we would assume the designers knew $\rho_{1...K,1...T}$, the designers would still not know the correlation between the values of the matrix $\rho_{1...K,1...T}$. Yet, these values are correlated significantly in practice.

In order to calculate a lower bound for the number of samples only based on ρ_{max}, we use the following observation: The number of samples that is needed to see a peak in practice, is mainly determined by the distance between the sampling distributions with $\rho = 0$ and $\rho = \rho_{max}$. All values of $\mathcal{R}$ are drawn from one of these two sampling distributions. Clearly, the more overlap there is between these distributions, the less likely it is to see a significant peak in $\mathcal{R}$. An attacker can decrease this overlap by increasing the number of samples (see equation 9).

In order to measure the distance between the distributions, we calculate the probability that a value drawn from the distribution with $\rho = \rho_{max}$ is bigger than one that is drawn from the distribution with $\rho = 0$. This probability α can be calculated as shown in equation 10. This equation can be transformed to equation 11, which allows a direct calculation of the number of samples based on ρ_{max}.

$$\alpha = \Phi \left(\frac{\frac{1}{2} \ln \frac{1+\rho_{max}}{1-\rho_{max}} - \frac{1}{2} \ln \frac{1+0}{1-0}}{\sqrt{\frac{2}{S-3}}} \right) \tag{10}$$

$$S = 3 + 8 \left(\frac{Z_\alpha}{\ln \left(\frac{1+\rho_{max}}{1-\rho_{max}} \right)} \right)^2 \tag{11}$$

The quantile Z_α determines the distance between the distributions with $\rho = 0$ and $\rho = \rho_{max}$. The higher the probability α is, the bigger is the distance between the distributions and, consequently, the more likely it is to see a peak.

In practice, several values are drawn from each of these distributions. Yet, these values are not drawn independently. Therefore, it is hard to calculate the exact probability for a peak, and we have to rely on empirical results to approximate a lower bound for the number of needed samples.

Based on several practical attacks and simulations, we have determined that $\alpha = 0.9$ is a reasonable value to calculate a lower bound for the number of samples needed in a DPA attack. Setting $\alpha = 0.9999$ in equation 11 on the other hand, leads to a number of samples that reveals the attacked subkey with very high probability. Between $\alpha = 0.9$ and $\alpha = 0.9999$ there is a "gray area". The lower the value of α is, the lower is the probability of observing a significant peak in the correlation trace $r_{k_c,1...T}$. The levels $\alpha = 0.9$ and $\alpha = 0.9999$ have been chosen in a very conservative way.

In order to get more exact bounds for a particular device, the levels α may be refined as soon as simulated or measured power traces of the device are available.

Based on the formulas we have provided in this subsection, designers of cryptographic devices can determine the effect of hardware countermeasures on the number of samples as follows:

First, ρ_{max} is calculated according to equation 6. The parameters needed for this calculation, are conservatively assessed by the designers as good as it is possible at the respective stage of the design process. Based on ρ_{max}, a lower bound for the number of needed samples can then be calculated according to equation 11.

5 Empirical Verification

In order to empirically verify the formulas derived in the last section, we implemented AES-128 on an 8-bit micro controller. The micro controller was clocked with 11MHz and its power consumption was sampled with 250 MS/s during 4000 AES-128 encryptions.

We attacked an 8-bit intermediate result of AES-128 at the time it was transferred over the pre-charged bus of the micro controller. In order to verify equation 6, a different number of bits of this intermediate result were attacked. From an attacker's point of view the bits that are transferred over the bus, but are not part of the attacked intermediate result, are noise. Of course, there is also other noise in the measurement, besides the power consumption of these bits.

However, since we are not familiar with the details of the design of the micro controller, we had to assume that this noise is zero for our first calculation of ρ_{max} based on equation 12. In this equation, b is the number of bits that are attacked on the bus, and n is the variable representing the additional noise in the power traces.

Table 1. Comparison of calculated correlations with the empirically determined correlation coefficients for 4000 samples

Number of Attacked Bits	1	2	3	4	5	6	7	8
Calculated ρ_{max} $(n = 0)$	0.35	0.50	0.61	0.71	0.79	0.87	0.94	1.00
Calculated ρ_{max} $(n = 2)$	0.32	0.45	0.55	0.63	0.71	0.77	0.84	0.89
Measured r_{max} (4000 samples)	0.31	0.44	0.53	0.63	0.70	0.76	0.82	0.90

$$\rho_{max} = \frac{1}{\sqrt{1 + \frac{1}{SNR}}} = \frac{1}{\sqrt{1 + \frac{n+(8-b)}{b}}} \tag{12}$$

The first line of table 1 shows ρ_{max} for $n = 0$ and $b = 1 \ldots 8$. In the second line, the corresponding values are shown for $n = 2$. The correlation coefficients we determined empirically by performing a DPA attack with 4000 samples, can be found in line three.

The values ρ_{max} calculated based on $n = 0$, are higher than the ones determined empirically. This is a logical consequence of the fact that no noise was assumed for the calculation. However, the values in the second and third line match almost exactly—obviously setting $n = 2$ models the noise of the micro controller very well. The slight deviations between the lines two and three are a consequence of the fact that not all wires of the bus of the micro controller have the same power consumption characteristics.

Based on the micro controller, we also verified the effect of random delays on ρ_{max}. For this purpose, we disarranged the 4000 power traces using a binomial distribution with $p = \frac{1}{2}$ and $n = 50$ clock cycles for the delay—the maximum probability of this distribution is $p = \frac{1}{2^{50}}\binom{50}{25}$. However, when calculating $\hat{p}$, the fact that the micro controller processes the attacked intermediate result twice needed to be considered. The micro controller we used, processed the attacked intermediate result in two subsequent clock cycles. Consequently, $\hat{p}$ was approximated by $2 * \frac{1}{2^{50}}\binom{50}{25}$. We attacked a 4-bit intermediate result using the disarranged traces. Consequently, ρ_{max} could be determined as shown in equation 13.

$$\rho_{max} \approx \frac{1}{\sqrt{1 + \frac{6}{4}}} * 2 * \frac{1}{2^{50}}\binom{50}{25} * \sqrt{\frac{47}{260}} = 0.06 \tag{13}$$

The quotient $\frac{47}{260}$ was determined empirically. Performing 1000 attacks with different random delays based on 4000 power traces of the micro controller, lead to mean of $r_{max} = 0.063$.

In the next step, we verified the calculation of the number of samples. We calculated S based on equation 11 for the attacks without random delays which we described before. The calculated number of samples for $\alpha = 0.9$ and $\alpha = 0.9999$ are shown in table 2 (ρ_{max} was calculated using equation 12 with $n = 2$). The big difference between the number of samples for the same attack with

Table 2. The number of samples needed in a DPA attack on the micro controller for $\alpha = 0.9$ and $\alpha = 0.9999$

Number of Attacked Bits	1	2	3	4	5	6	7	8
S calculated with $\alpha = 0.9$	34	17	12	9	7	6	5	5
S calculated with $\alpha = 0.9999$	261	122	76	53	39	29	22	16

different values α, again shows that the levels $\alpha = 0.9$ and $\alpha = 0.9999$ have been chosen very conservatively.

We have performed DPA attacks with the calculated number of samples. Clearly visible peaks occurred in the attacks conducted with the numbers of samples calculated based on $\alpha = 0.9999$. In attacks with the numbers of samples shown in the first line of table 2, only some sporadic peaks occurred in hundreds of attacks.

Hence, we have been able to verify empirically all formulas presented in this article, based on attacks on an 8-bit micro controller.

6 Conclusions

Designers of cryptographic devices require methods to assess the effect of hardware countermeasures on the number of samples needed in DPA attacks. Such methods are necessary in order to avoid costly design iterations.

In this article, we have identified those properties of hardware countermeasures that affect the number of samples needed in DPA attacks. Based on these properties, we have derived formulas that allow the calculation of lower bounds for the number of samples needed in DPA attacks.

The presented formulas enable designers to assess the resistance of their devices against DPA attacks from the earliest stages of the design process onwards until the fabrication. This way designers can verify that the combination of countermeasures they have chosen to implement in their devices, indeed provides the required protection against DPA attacks.

References

1. D. Agrawal, B. Archambeault, J.R. Rao, and P. Rohatgi. The EM Side-channel(s). In *Cryptographic Hardware and Embedded Systems – CHES 2002*, Lecture Notes in Computer Science (LNCS). Springer-Verlag, 2002.
2. L. Benini, A. Macii, E. Macii, E. Omerbegovic, M. Poncino, and F. Pro. Energy-Aware Design Techniques for Differential Power Analysis Protection. In *40th Design Automation Conference – DAC 2003*. ACM, 2003.
3. C. Clavier, J.-S. Coron, and N. Dabbous. Differential Power Analysis in the presence of Hardware Countermeasures. In *Workshop on Cryptographic Hardware and Embedded Systems – CHES 2000*, volume 1965 of *Lecture Notes in Computer Science (LNCS)*, pages 252–263. Springer-Verlag, 2000.

4. J.-S. Coron. Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems. In *Workshop on Cryptographic Hardware and Embedded Systems – CHES 1999*, volume 1717 of *Lecture Notes in Computer Science (LNCS)*, pages 292–302. Springer-Verlag, 1999.
5. J. Dj. Golic and C. Tymen. Multiplicative Masking and Power Analysis of AES. In *Cryptographic Hardware and Embedded Systems – CHES 2002*, Lecture Notes in Computer Science (LNCS). Springer-Verlag, 2002.
6. L. Goubin. A Sound Method for Switching between Boolean and Arithmetic Masking. In *Workshop on Cryptographic Hardware and Embedded Systems – CHES 2001*, volume 2162 of *Lecture Notes in Computer Science (LNCS)*, pages 3–15. Springer-Verlag, 2001.
7. L. Goubin and J. Patarin. DES and Differential Power Analysis – The Duplication Method. In *Cryptographic Hardware and Embedded Systems - CHES 1999*, volume 1717 of *Lecture Notes in Computer Science*, pages 158–172. Springer-Verlag, 1999.
8. J. Irwin, D. Page, and N.P. Smart. Instruction Stream Mutation for Non-Deterministic Processors. In *IEEE International Conference on Application-Specific Systems, Architectures, and Processors – ASAP 2002*, pages 286–295. IEEE, 2002.
9. P.C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS and Related Attacks. In *Advances in Cryptology – CRYPTO 1996*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer-Verlag, 1996.
10. P.C. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In *Advances in Cryptology – CRYPTO 1999*, volume 1666 of *Lecture Notes in Computer Science (LNCS)*, pages 388–397. Springer-Verlag, 1999.
11. S. Mangard. A Simple Power-Analysis (SPA) Attack on Implementations of the AES Key Expansion. In *Information Security and Cryptology – ICISC 2002*, volume 2587 of *Lecture Notes in Computer Science (LNCS)*, pages 343–358. Springer-Verlag, 2002.
12. D. May, H.L. Muller, and N.P. Smart. Non-deterministic Processors. In *Information Security and Privacy – ACISP 2001*, volume 2119 of *Lecture Notes in Computer Science (LNCS)*, pages 115–129. Springer-Verlag, 2001.
13. T.S. Messerges. Using Second-Order Power Analysis to Attack DPA Resistant Software. In *Cryptographic Hardware and Embedded Systems – CHES 2000*, volume 1965 of *Lecture Notes in Computer Science (LNCS)*, pages 238–251. Springer-Verlag, 2000.
14. T.S. Messerges, E. A. Dabbish, and R. H. Sloan. Power Analysis Attacks of Modular Exponentiation in Smartcards. In *Cryptographic Hardware and Embedded Systems – CHES 1999*, volume 1717 of *Lecture Notes in Computer Science*, pages 144–157. Springer-Verlag, 2000.
15. T.S. Messerges, E.A. Dabbish, and R.H. Sloan. Examining Smart-Card Security under the Threat of Power Analysis Attacks. *IEEE Transactions on Computers*, 51(5), 2002.
16. S. Moore, R. Anderson, P. Cunningham, R. Mullins, and G. Taylor. Improving Smart Card Security using Self-timed Circuits. In *Eighth IEEE International Symposium on Asynchronous Circuits and Systems – Async 2002*. IEEE Computer Society Press, 2002.
17. S. Moore, Ross Anderson, Robert Mullins, and George Taylor. Balanced Self-Checking Asynchronous Logic for Smart Card Applications. In *Microprocessors and Microsystems Journal*, to appear.

18. E. Oswald. Enhancing Simple Power-Analysis Attacks on Elliptic Curve Cryptosystems. In *Cryptographic Hardware and Embedded Systems – CHES 2002*, volume 2523 of *Lecture Notes in Computer Science (LNCS)*, pages 82–97. Springer-Verlag, 2002.
19. H. Saputra, N. Vijaykrishnan, M. Kandemir, M.J. Irwin, R. Brooks, S. Kim, and W. Zhang. Masking the Energy Behavior of DES Encryption. In *Design, Automation and Test in Europe Conference and Exhibition – DATE 2003*, pages 84–89. IEEE, 2003.
20. K. Tiri, M. Akmal, and I. Verbauwhede. A Dynamic and Differential CMOS Logic with Signal Independent Power Consumption to Withstand Differential Power Analysis on Smart Cards. In *29th European Solid-State Circuits Conference – ESSCIRC 2002*, 2002.
21. K. Tiri and I. Verbauwhede. Securing Encryption Algorithms against DPA at the Logic Level: Next Generation Smart Card Technology. In *Cryptographic Hardware and Embedded Systems – CHES 2003*, volume 2779 of *Lecture Notes in Computer Science (LNCS)*, pages 125–136. Springer-Verlag, 2003.
22. E. Trichina, D. De Seta, and L. Germani. Simplified Adaptive Multiplicative Masking for AES and its Secure Implementation. In *Cryptographic Hardware and Embedded Systems – CHES 2002*, Lecture Notes in Computer Science (LNCS). Springer-Verlag, 2002.