Power Analysis Tutorial

Manfred Aigner and Elisabeth Oswald

Institute for Applied Information Processing and Communication
University of Technology Graz
Inffeldgasse 16a, A-8010 Graz, Austria
Manfred.Aigner@iaik.at
Elisabeth.Oswald@iaik.at

Abstract. Performing a *Differential Power Analysis* (*DPA*) attack requires knowledge in several fields; statistics and cryptography for the attack itself, programming skills and experience in instrumentation to build up an automatic measurement system and electronical skills to improve the results. This tutorial provides information on all these topics on basis of our experience.

1 Introduction

Since increasingly confidential data are being exchanged on electronic way an ever greater importance is attached to the protection of the data. Where cryptosystems are being used in real applications not only mathematical attacks have to be taken into account. Hard- and software implementations themselves present a vast field of attacks. *Side-Channel-Attacks* exploit information that leaks from a cryptographic device. Especially one of these new attacks has attracted much attention since it has been announced. This method is called *Differential Power Analysis (DPA)* and was presented in 1998 by *Cryptography Research*. *DPA* uses the information that naturally leaks from a cryptographic hardware device, namely the power consumption. A less powerful variant, the *Simple Power Analysis (SPA)* was also announced by *Cryptography Research*.

What does a *DPA* attack require? First, an attacker must be able to precisely measure the power consumption. Second, the attacker needs to know what algorithm is computed, and third an attacker needs the plain- or ciphertexts. The strategy of the attacker is to make a lot of measurements, and then divide them with the aid of some oracle into two or more different sets. Then, statistical methods are used to verify the oracle. If and only if the oracle was right, one can see noticeable peaks in the statistics. This vague description of a *DPA* attack should be clearified in this article. In section 2, a power model is developed and related to the statistical methods used in *DPA* . Thereafter, a *DPA* attack is explained on the grounds of the DES. In the third section, a concrete implementation of the *DPA* is discussed. The section begins with a C++-model which will turn out to be useful to verify some countermeasures against *DPA* attacks. Also an attack on a 8052-microprocessor implementation is described. In the fourth section the application of *DPA* on asymmetric cryptosystems is discussed.

2 Power Analysis Foundations

Almost every digital circuit built today is based on Complementary Metal Oxid Semiconductor (*CMOS*) technology. Therefore it is necessary to understand the power consumption characteristics of this technology. If a *CMOS* gate changes its state, this change can be measured at the V_{dd} (V_{ss}) pin. The more circuits change their state, the more power is dissipated. In a synchronous design, gates are clocked which means that all gates change their state at the same time. Power dissipated by the circuit can be monitored by using a small resistor R_m in series between V_{dd}, (or V_{ss}) and the true source (or ground). The two most essential parts of the power consumption during a change of a state are the dynamic charge resp. discharge (appr. 85%) and the dynamic short circuit current (appr. 15%). This is sketched on the example of an inverter (see Figure 1). The output of each gate has a capacitive load, consisting of the parasitic capacity of the connected wires and gates of the following stages. An input transition results in an output transition, which discharges or charges this parasitic capacity, causing a current flow to V_{dd} (or V_{ss}). This current is the dynamic charge resp. discharge current. For further information one should take a look at [21]. By measuring current flow on V_{dd} we can detect whether the output changed from 0 to 1 or not.

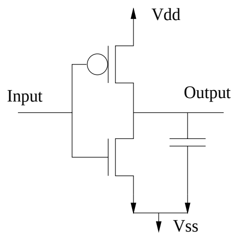

Fig. 1. Inverter

In differential *CMOS* logic, every output appears also in its inverted form, which means a transition always causes charge and discharge on the output and inverted output. By measuring current on V_{dd} or V_{ss} one can't distinguish high and low transitions, but it is possible to detect whether a transition occured or not.

Logic with precharge characteristic always charges the output capacity during a precharge cycle and decharges it during the evaluation cycle, in case that the output value differs from the precharge value. By observing current flow one can detect changes of the output node. Precharge logic has much higher power consumption than differential or standard *CMOS* logic, because dynamic charge current appears also in situations where the output value doesn't toggle.

2.1 Power Model

As a result of the previous explanation we can deduce an abstract model of the instantaneous power consumption of a *CMOS* circuit. The power consumption of a circuit at a particular time t is the sum of the power dissipated by all gates at this time. Of course, when measuring this power dissipation we cannot prevent the influence of noise. Various noise components have to be considered such as external noise, intrinsinc noise, quantization noise and algorithmic noise (see [15]). By the careful use of measurement equipment one can reduce external noise. Algorithmic noise will be reduced by *DPA* itself. The other two noise components should be small compared to the power consumption. We can state this more formally as :

Simple Power Model. *Let t denote the time, and $\mathcal{N}(t)$ be a normal distributed random variable which represents the noise components. Let $f(g,t)$ denote the power consumption of gate g at the time t. Then a simplified power model for the power consumption is the function*

$$P(t) = \sum_{g} f(g,t) + \mathcal{N}(t)$$

The next step is to relate this model to statistics. If we consider the function $f(g,t)$ as random variable from an unknown probability distribution, what can we say about $P(t)$? If all $f(g,t)$ are randomly and independently drawn from this probalility distribution then the *Central Limit Theorem* says that $P(t)$ is normally distributed. In a *DPA* attack the attacker divides the power measurements in two or more different sets and tries to compute the difference between these sets in order to verify the oracle. As we have related the power consumption to statistics we can also say that the attacker wants to compute the difference between the two probability distributions. The methods therefore are discussed in the next section.

2.2 Hypothesis Testing

If one works with probability distributions it is necessary to have characterizations of these distributions. Well known characteristics are the expectation, variance or more in general the *moments* of the distribution. Of course the true expectation (or true variance) is unknown, so one has to estimate it. The construction of good estimators is one of the main goals in statistics. One can easily proof that if the X_i are independently, identically distributed random variables, the statistical mean $\bar{X}$ and variance S^2 are good estimators for the true expectation $E(X_i) = \mu$ and the true variance $Var(X_i) = \sigma^2$. Common strategies for constructing estimators are:

- using the empirical moments as estimators for the theoretical moments
- using the *Maximum-Likelihood* method.

In the case stated above one can prove that both strategies lead to the same estimators, and that these are the best estimators (in the sense that the expected quadratic deviation to the theoretical moment is the smallest) one can find. We conclude that we can distinguish probability distributions if we can distinguish their moments. This is done by using the following test.

T-Test. *We consider two independent samplevectors* $(X_1, \ldots, X_n)$ *with* $X_i \sim N(\mu_x, \sigma)$ *and* $(Y_1, \ldots, Y_m)$ *with* $Y_j \sim N(\mu_y, \sigma)$. *We test the hypothesis that*

$$H_0 : \mu_x - \mu_y = 0 \text{ versus } H_1 : \mu_x - \mu_y \neq 0$$

with the test statistic

$$T = \frac{\bar{X} - \bar{Y}}{S_p} \sqrt{\frac{nm}{n+m}} \sim t_{n+m-2}$$

with $S_p^2 = \frac{1}{n+m-2}((n-1)S_x^2 + (m-1)S_y^2)$, $S_x^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})^2$ *and* $S_y^2 = \frac{1}{m-1}\sum_{j=1}^{m}(Y_i - \bar{Y})^2$.

The principle of the test is, to compute the difference of the distribution means. One can use a modified variant of the test if the variances are not the same. The modifications affect the probability distribution of the test function T, but our experience with *DPA* shows that it is sufficient to compute $T = \bar{X} - \bar{Y}$, so we do not consider this in detail. We will refer to this method as *mean method* in subsequent sections.

Another possibility is to survey a hypothesis about the probability distribution itself so one can test

$$H_0 : F(x) = F_0(x) \text{ versus } H_1 : \exists x \in \mathbb{R} : F(x) \neq F_0(x).$$

Thereby we estimate the unknown distribution function $F(x)$ by the empirical distribution function $F_n(x)$. Two tests are used in general, the $\chi^2 - Test$ and the $Kolmogorov - Smirnov - Test$. We refer for an explanation of these methods to one of the various statistic books.

The third possibility is to ask how many influence does the oracle on the measurements have? If the oracle was correct, then there should be some correlation between it and the measurements. Influence, or dependence is given by the *covariance* σ_{xy} or *correlation* $\rho(x, y)$ with

$$\sigma_{xy} = E(XY) - E(X)E(Y),\ \rho(x, y) = \frac{\sigma_{xy}}{\sigma_x \sigma_y.}$$

Remark 1. Let X, Y denote independent random variables, then $\rho(X, Y) = 0$.

The conversion does not apply in general, but for normal variables. Again one can estimate the theoretical covariance and correlation by the empirical moments. The hypothesis here would be

$$H_0 : \rho = 0 \text{ versus } H_1 : \rho \neq 0.$$

We will refer to this method as *correlation method* in subsequent sections. From this explanation it is clear, that the simplest method is to compute the difference between the means of the sample sets. This is the *DPA* function we will use for the attack on the DES .

2.3 Differential Power Analysis

In the previous section the foundations for *DPA* have been explained. Now we start with the power analysis itself. In the first part of this section we will give a short review of the DES. Then the construction of an oracle for a known-ciphertext-attack is explained in this context.

The *Data Encryption Standard* ([20]) was invented in 1970 by IBM. It has a *Feistel-Structure* and consists of 16 rounds:

$$\begin{aligned} L_i &= R_{i-1} \\ R_i &= L_{i-1} \oplus f(R_{i-1}, K_i), \\ \text{where } f(R_{i-1}, K_i) &= P(S(E(R_{i-1} \oplus K_i)), \text{ (see Figure 2)}. \end{aligned}$$

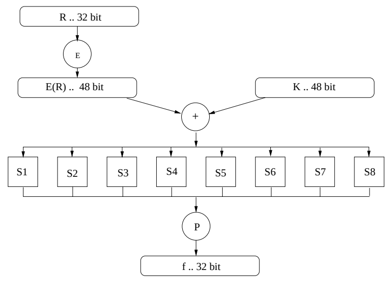

Fig. 2. DES round function

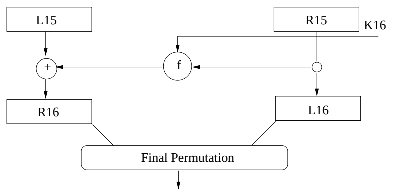

Fig. 3. DES last round

K_i denotes the subkey of the $i-th$ round. The last DES round differs form the others, because L and R are not exchanged (see Figure 3). The oracle or as we now call it, *selection funtion* D makes use of this fact. As we can see in the Figure 3, $R_{15} = L_{16}$. The subkey splits up in eight blocks, one for every sbox (see Figure 2). Therefore we specifiy one *target sbox* for which we list all possible $(= 2^6)$ input values. We will refer to such an input value as subkeyblock. As assumed above we know the ciphertexts, and so we can calculate the value of some of the bits in L_{15} for every possible subkeyblock. We select one of these bits as our *target bit*. The value of the target bit is our *selection function* D. If $D = 1$ the corrsponding power measurement will be put in sample set $S1$, if $D = 0$ it is classified to $S0$. This procedure is repeated for a lot of measurements, so at the end we have, for every ciphertext and all subkeyblocks, a classification of the corresponding measurement. Let n denote the amount of ciphertexts, resp. measurements. Then we can write all our classifications in a $2^6 \times n$ matrix. So every line represents a possible key for the target sbox, and every column represents the classification of one ciphertext resp. measurement.

For the *DPA* attack we go through all lines and build the two sample sets $S0$ and $S1$. Then we compute the mean (pointwise) of the samples in the sets, $M0$

and $M1$, and compute the difference. For the correct subkeyblock there must be a peak in the trace of the difference.

3 Practical Implementation

This section considers implementation issues. Let's start with some *SPA* experiments to familiarize ourselfs with the measurement equipment. *SPA* tries to identify single instructions in the power trace without statistical methods. It can be used to detect the portions in the powertrace where the target bit for *DPA* is manipulated and it can be used to develop a good measurement setup. We used a 8051 compatible ATMEL 89S8252 microcontroller for our experiments. In the first stage it is useful to perform simple operations on the microprocessor and play around with the various possibilities for the measurement setup. We execute a number of $mov\ addr, \#0$ and $mov\ addr, \#255$ instructions and measure the power consumption. The aim of this experiment is to optimize the measurement setup of the microcontroller board. It is easy to see that noise on the power supply reduces the precision of measurements seriously. The first step to improve our setup is reducing noise on the power supply. The next step is to choose the right value for resistor R_m between global supply and the supply pin of the controller across which we measured the current profile. Bigger R_m would mean higher voltage swing across the resistor, which would be easier to measure. One has to keep in mind that this voltage drop across R_m reduces the actual supply voltage of the controller which reduces the power consumption. Therefore it is clear that big values for R_m do not directly lead to the desired effect of higher voltage swing. Power consumption itself also depends heavily on the amount of the supply voltage, so to obtain better results one should run the device at the highest supply voltage possible.

A second effect of the reduced supply voltage has to be taken into account: Input protection circuits of *CMOS* pads include clamp diodes which turn on, when voltage on the input pad is higher than the circuit's V_{dd}. Introducing R_m leads to reduced V_{dd}, which makes these diodes conductive when the input value is high and voltage drop on R_m is big. This means the internal circuit is supplied by input current from the input pads, which is not measured via R_m. Smaller values of R_m reduce this effect. Another way to get rid of this effect is to make the global supply voltage bigger than the high level, but this would reduce the high noise margin of the circuit.

Although the *CMOS* circuit still works with a big R_m the current profile is more influenced by a bigger resistor. We measured nearly the same voltage swing on R_m values for $R_m = 1\Omega$ and $R_m = 20\Omega$.

Since power consumption of *CMOS* logic arises mostly around clock transitions, the current profile has high frequency components, which lead to voltage overshoot on R_m. To reduce this overshoot, a small fast capacitor should be connected parallel to R_m. The best way to find out the optimal value of this capacitor is to try different devices and decide after some measurements if the desired effect has happened. The input capacitance and resitance of the oscillator probe used for measurements has also a big impact on this effect, so use an active differential probe with high input resistance and very low input capacitance.

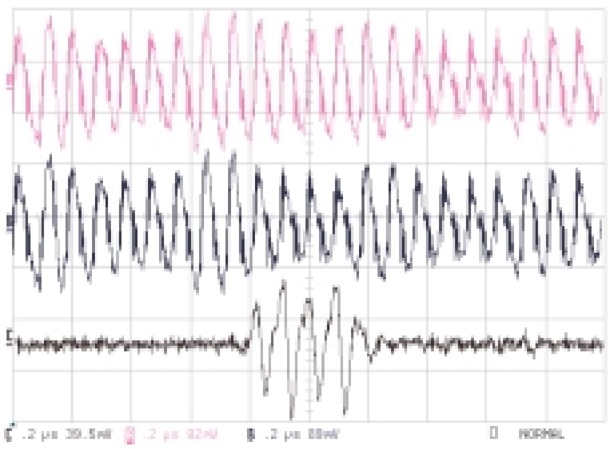

Fig. 4. MOV 0 versus MOV FF

After reducing noise of the power supply, finding the best resistor R_m and capacitores C_m it is possible to detect the 2-bit differences of two consecutive *mov*-instructions of our microcontroller board on the oszilloscope.

See Figure 4 for the current profile of two *mov* commands. Graph A and B show each the profile of a *mov* command while C is the zoomed difference of these two samples. It's even easier to detect differencies of distinct commands. With a good setup we can even find out if the branching condition of a JNZ or JZ command was fulfilled or not.

3.1 C++-Model

The next step is to implement a *DPA* attack. We do this on a DES C++-model first. For the functional verification of the implementation of an algorithm, often C++ is used before the algorithm is written in a hardware description language *(HDL)*. In the *top-down* design flow one splits up the (hardware) module in submodules. The functionality of each of the submodules and the connections between them can be defined in the C++-model too. Finally one can produce cycle-tuned test vectors to verify the HDL-model.

3.1.1 Modelling the Power Consumption in the C++-Model. If one has a bit-level model of a hardware modul, it is fairly easy to model the dynamic charge (discharge) of the capacitances too. It is sufficient to assert that the change of state of a bit implicates current flow in the type of logic used. After all bits of a cycle are processed, a variable holds the value of the instantaneous power consumption. In the following example this procedure is sketched for standard *CMOS* logic and differential *CMOS* logic.

```
extern int i_stdcmos,i_diffcmos;
void FLIP_FLOP::sim(ONE_BIT rst, ONE_BIT D, ONE_BIT phi)
{
  if (phi== LOW) {
       if (rst==HIGH) {
         Q_buf = LOW;
       } else {
         Q_buf = D;
       }
  } /* End if (phi == LOW) */

  if (phi == HIGH) {
       if (Q==LOW && Q_buf==HIGH) { // i on Vdd for standard CMOS logic
         i_stdcmos++;// FF goes from LOW -> HIGH .. i on Vdd
       }
       if (Q != Q_buf) {// i on vdd for differential cmos logic
         i_diffcmos++;
       }
       Q = Q_buf;
  } //End if (phi == HIGH)
}
```

Fig. 5. Flip-Flop C++-model

What are the advantages when proceeding this way? When making hardware implementations writing a C++-model first is part of the usual design flow. Therefore one should get as many information out of this model as possible, including *DPA* related information. Second, real hardware implementations are of high costs. So it doesn't make sense to design for every *CMOS* type a real chip and then to prove *DPA* resistance. Third, such a power model gives one a first hint how difficult a real *DPA* attack will be. It is plain to see that one will need at least as many samples in reality as in the C++-model.

With this model we did some *DPA* experiments on the various *CMOS* types. The standard *CMOS* logic was used as a reference implementation to prove the correctness of the attack. On the differential *CMOS* logic the *DPA* attack did not work as presented; after a slight modification on the selection function, *DPA* was successful again. Adding precharge property could accomplish *DPA* resistance. Other types of countermeasures (see [9], [3]) suggest to conceal the targetbits with the aid of some function that represent bits without having their actual value. This type of countermeasures can also be tested easily on a C++-model.

3.2 DPA on a 8052 DES Implementation

For an easy *DPA* attack it is important to reduce noise wherever possible. Therefore we assembled a board with an 8-bit Atmel microcontroller. Because of the

8-bit architecture low algorithmic noise may be expected. The DES implementation we use is straightforward, but without grave design problems concerning *SPA* (e.g. no branches depending on key bits etc.). A block in the internal memory called Key is used to store the shifted key. Another block called *Round* stores intermediate values. The sboxes are in the code memory. BL and BR, the results of the rounds, reside also in the internal memory. For a DES round the key is first shiftet in the Key memory block, and then the result of $PC2$ is put in *Round*. The expansion takes its input from BR and holds its output in the accumulator. The result of the following *xor* and the sbox substitution result is stored in *Round*. P permutation output values are stored in the $Temp$ memory block. Finally, the result is written to BR after its previous value was copied to BL.

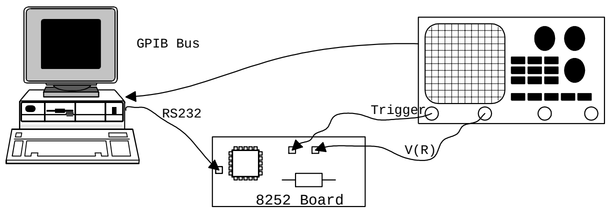

Fig. 6. DPA Setup

The measurement setup is sketched in Figure 6. Data is written from a PC via the $RS232$ to the 8052-board where the encryption is performed. The board also sets the trigger for the digital oszilloscope which is connected via the $GPIB$-bus to the PC. For our experiments we use an ordinary Intel Pentium II with 128 MB RAM and two 3 GB harddisks, where all software and measurement data is stored. Our analysis software is written in C^{++} as well as in Matlab, which we primarily use because of its simplicity when visualizing the measurements and *DPA* results. Because of our *SPA* experiments we decided to trigger first on the *mov* instruction in round 15 where the result of round 14 is stored to L_{15}. We identified the corresponding code segment and set an appropriate trigger. Because of the exact trigger signal we can make very precise measurements for a very short time period. This results in very small measurement samples for faster *DPA* computation. We can test more than one subkeyblock per second. With our best setup we need less then 200 samples to identify the correct subkeyblock.

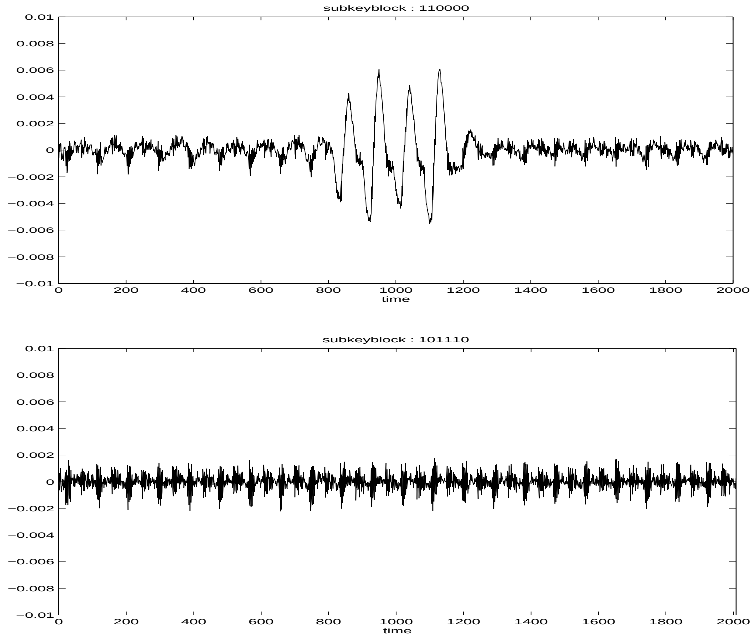

Fig. 7.

Figure 7 shows a typical result of this experiment. As target bit we used bit 3 of sbox 4. The correct value for the subkeyblock shows similar characteristics as the *mov* plot from our previous experiments. Sampling the power consumption during the last round, or during the complete DES-computation leads to lower measurement resolution, because memory of the digital oszilloscope is limited. Lower measurement resolution of the power sample leads to a higher amount of neccesary measurements to identify the correct subkeyblock. We looked therefore for other classifications methods of our *DPA* results. It is interesting not only to look for peaks in the time domain but also in the frequency domain.

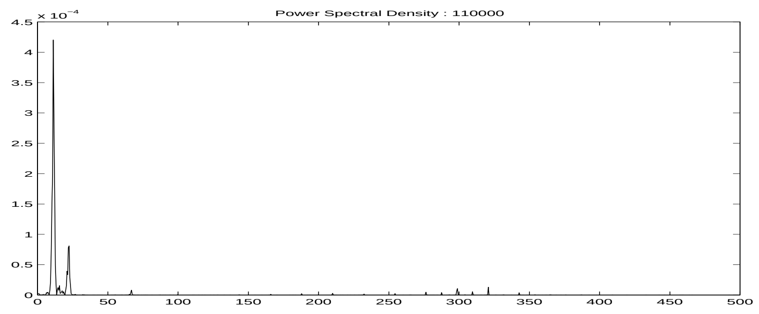

Fig. 8.

Figure 8 shows the power spectral density for the correct subkeyblock. The dominating peak in the low frequency region shows exactly the base frequency of the DPA-burst in the time domain (Fig. 7). This spike (as the burst) only appears in the sample for the correct subkeyblock, that means it is very easy to distinguish the correct subkeyblock from the others in the frequency domain.

4 *DPA* and Public Key Cryptography

Also public key cryptosystems are vulnerable to *DPA* or *SPA* attacks. Specifically modular exponentiation as used in RSA and point mulitplication as used in ECC are investigated in this section. The two different statistical methods described in section 2.2 give two different approaches for a *DPA* attack on these cryptosystems. First we briefly review the encryption functions of the two cryptosystems.

In RSA, one has a public key (e, n) and a private key $d = d_{n-1}d_{n-2}\ldots d_0$. When creating an encrypted message C one has to compute $C = P^e \mod n$. Decryption is done by $P = C^d \mod n$. The modular exponentiation is usually done by the *square-and-multiply* algorithm. The ECC scheme uses as its encryption function the scalar multiplication of a point P on the elliptic curve. The decryption requires the computation of dP where $d = d_{n-1}d_{n-2}\ldots d_0$ is the secret parameter. The scalar multiplication is done by the *double-and-add* method which works similar to the square-and-multiply algorithm. Both methods (they will be refered to as the *binary algorithm* in subsequent sections) are sketched in Algorithm 1 where $*$ denotes the group operation, which is the multiplication for RSA and the addition for ECC .

Multiplication and squaring can be implemented to produce the same power traces, but this is much more difficult for point addition and point duplication. It is clear that in order to be *SPA* resistant, one must prevent data depending branches. A simple modification of Algorithm 1 is Algorithm $1'$ where the "if" statement has been avoided. This solution is of course more cost expensive.

Algorithm 1

```
X = X
for $i = n - 2$ downto 0
  $X = X * X$
  if ($d_i == 1$) then
    X = X*M
return X
```

Algorithm 1'

```
$X[2] = M$
for $i = n - 2$ downto 0
  $X[0] = X[2] * X[2]$
  $X[1] = X[2] * M$
  $X[2] = X[d_i]$
return $X[2]$
```

Elliptic curve subtraction has the same cost as addition because $-P(x, y) = P(x, -y)$. The double-and-add algorithm can be improved with the *addition-subtraction* method. The problem of computing dP with the fewest number of elliptic curve operations is equivalent to find the shortest addition-subtraction chain [19].

4.1 Attacking the secret parameter with the *mean method*

It is assumed that the attacker has full control over the cryptographic device that performs the binary algorithm. This includes that the attacker can make as many measurements as needed.

4.1.1 Single-Exponent, Multiple-Data Attack. The SEMD attack [16] compares the power signal of an encryption operation using a known parameter to a power signal using an unknown parameter. The attacker can learn where the two signals differ and thus learn the unknown (secret) parameter. Due to noise components, direct comparisons of power signals are unreliable, thus *DPA* techniques are applied. One computes n random values with the secret and the known parameter. The average signals are calculated and subtracted as in the *mean method* . The portions of the *DPA* signal that depend on the (random) data will be wiped out by the averaging and subtraction. The portion of the *DPA* signal that is dependend on the paramter will average out to two different values depending on the performed operation. So the portions in the *DPA* signal that are ≈ 0 are data dependend or the operations in the binary algorithm agree. The other portions indicate that the operations in the binary algorithm differ.

4.1.2 Multiple-Exponent, Single-Data Attack. It is assumed that the attacker can encrypt a constant (maybe unknown) value P with parameters chosen by the attacker. One measures mean power consumption S_m of the encryption of the value P. Then the bits of the secret parameter are attacked successively. To attack the ith bit, the attacker first guesses that the ith bit is a 1 and then a 0 and starts the encryption for both guesses. We assume that the attacker has already learned the first $i - 1$ bits. If the attacker guesses bit i correctly the power trace will agree to bit i with the power trace S_m. If the guess was wrong, one should see a difference in the corresponding power trace [16].

4.1.3 Zero-Exponent, Multiple-Data Attack.

4.1.3 Zero-Exponent, Multiple-Data Attack. The ZEMD attack [16] assumes that one can encrypt many random messages using the secret paramter. The attacker additionally needs to be able to compute intermediate results of the binary algorithm. The attacker learns the secret parameter bitwise. One guesses the ith bit of the secret parameter and creates a *DPA* power signal. This is done by encrypting some random input P and measuring its power consumtpion. The power signals from the guesses can be partitioned by their Hamming weight and the average signals are subtracted. If the guess was correct, one can see the *DPA* bias in the correct partitioning.

4.2 Attacking the secret parameter with the *correlation method*

When using algorithm $1'$ the *mean method* will be not succesful because there is no difference in the sequence of instructions. But if one knows the representation of the computed points one can again mount a succesful attack [5]. At step i the processed point X depends only on the first bits $d_{l-1} \dots d_i$ of the secret parameter d. When X is processed, power consumptions is correlated to the bits of X. No correlation will be observed if the point is not computed. The second most significant bit can be learned by calculating the correlation between the power consumption and any specific bit of $4X$. If $d_{l-2} = 0$, $4X$ is computed during the binary algorithm. Otherwise if $d_{l-2} = 1$, $4X$ is never computed and thus there will be no correlation observed. The other bits can be recursivly recovered in the same way.

4.3 Countermeasures

The mechanism described in [12] can be used to prevent this types of power analysis techniques. Message blinding would prevent the MESD and ZEMD attack, but not the SEMD attack. Therefore one would also have to blind the secret parameter. When implementing ECC schemes there is one more countermeasure. Projective coordinates [14] can be used to randomize the point X. Before each new computation of the scalar multiplication dX, the projective coordinates of X are randomized, this makes an attack infeasible since it is not possible to predict any bit of x, see [5]. As pointed out in [15], another way could be to randomize the binary algorithm. One could, for example, randomize the addition-subtraction chains in [19].

5 Resuming

The various components of a *DPA* attack are discussed in this tutorial. On the grounds of the power consumption characteristic of *CMOS* logic we develop a simple power model. This model is then linked to statistics in order to relate the powerful methods of hypothesis testing to *DPA*. An example of a *DPA* oracle (or selection function) is given on the basis of the DES algorithm. Afterwards we describe a *DPA* implementation with the simplest statistical method and the developed selection function. First we explain how to improve the measurement setup and how to verify the automatic analysis by the use of a C++-model. Second we illustrate our measurement system and give hints on how to interpret and characterize the *DPA* results. The last section covers *DPA* attacks on public key cryptosystems. We explain briefly how the two statistical methods developed in the first section can be applied to RSA and ECC cryptosystems.

Following these hints it should be possible to make a *DPA* attack on such a software DES implementation with less then 200 measurements. Our reference list includes not only articles used for this tutorial, but also most actually available literature on this topic.

References

1. E. Biham, A. Shamir, *Power Analysis of the Key Scheduling of the AES Candidates* Second AES Candidate Conference, Rome, March 1999, pp 115-121.
2. S. Chari, Ch. Jutla, J. Rao, P. Rohatgi.*A Cautionary Note Regarding Evaluation of AES Candidates on Smart-Cards.* Second AES Candidate Conference, Rome, March 22-23,1999, pp 133-147.
3. S. Chari, Ch. Jutla, J. Rao, P. Rohatgi.*Towards Sound Approaches to Counteract Power-Analysis Attacks*, Proceedings of Advances in Cryptology-CRYPTO'99, Springer-Verlag, 1999, pp.398-412
4. C. Clavier, J.-S. Coron, N. Dabbous, *Differential Power Analysis in the presence of Hardware Countermeasures*, to appear in Proceedings of Workshop on Cryptographic Hardware and Embedded Systems 2000
5. J.-S. Coron, *Resistance against differential power analysis for elliptic curve cryptosystems*, Workshop on Cryptographic Hardware and Embedded Systems, Lecture Notes in Computer Science, vol. 1717, Springer,1999, pp.292-302
6. J.-S. Coron, L. Goubin, *On Boolean and Arithmetic Masking against Differential Power Analysis*, to appear in Proceedings of Workshop on Cryptographic Hardware and Embedded Systems 2000
7. J.-S. Coron, P. Kocher, D. Naccache, *Statistics and Secret Leackage*, to appear in Proceedings of Financial Cryptography, Springer-Verlag, February 2000
8. P. Fahn, P. Pearson. *IPA: A New Class of Power Attacks*, Proceedings of Workshop on Cryptographic Hardware and Embedded Systems (CHES), Springer-Verlag's Lecture Notes in Computer Science (LNCS), 1999
9. L. Goubin, J. Patarin.*DES and Differential Power Analysis.* Workshop on Cryptographic Hardware and Embedded Systems, Lecture Notes in Computer Science, vol. 1717, Springer 1999, pp 158-172.
10. M. A. Hasan, *Power Analysis Attacks and Algorithmic Approaches to Their Countermeasures for Koblitz Cryptosystems*, to appear in Proceedings of Workshop on Cryptographic Hardware and Embedded Systems 2000
11. P. Kocher, *Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS and Related Attacks*, Proceedings of Advances in Cryptology-CRYPTO'96, Springer-Verlag, 1996, pp. 104-130.
12. P. Kocher, J. Jaffe and B. Jun, *Differential Power Analysis*, Proceedings of Advances in Cryptology-CRYPTO'99, Springer-Verlag, 1999, pp. 388-397
13. R. Mayer-Sommer, *Smartly Analyzing the Simplicity and the Power of Simple Power Analysis on Smartcards*, to appear in Proceedings of Workshop on Cryptographic Hardware and Embedded Systems 2000

14. A. J. Menezes, *Elliptic Curve Public Key Cryptosystems*, Kluwer Academic Publishers, 1993
15. T.S. Messerges, E. A. Dabbish and R. H. Sloan, *Investigations of Power Analysis Attacks on Smartcards*, Proceedings of USENIX Workshop on Smartcard Technology, May 1999, pp. 151-61.
16. T.S. Messerges, E. A. Dabbish and R. H. Sloan, *Power Analysis Attacks of Modular Exponentiation in Smartcards*, Workshop on Cryptographic Hardware and Embedded Systems, Lecture Notes in Computer Science, vol. 1717, Springer,1999.
17. T. S. Messerges, *Using Second-Order Power Analysis to Attack DPA Resistant Software*, to appear in Proceedings of Workshop on Cryptographic Hardware and Embedded Systems 2000
18. A. Shamir,*Protecting Smart Cards from Passive Power Analysis with Detached Power Supplies*, to appear in Proceedings of Workshop on Cryptographic Hardware and Embedded Systems 2000
19. F. Morain, J. Olivos. *Speeding up the computation on an elliptic curve using addition-subtraction chains*, Inform. Theory Appl. 24 (1990), 531-543.
20. National Bureau of Standards, *Data Encryption Standard*, U.S. Department of Commerce, FIPS pub. 46, Jannuary 1977.
21. N. Weste and K. Eshraghian, *Principles of CMOS VLSI Design*, Addison-Wesley Publishing Company, 1993.