A DPA Attack against Asymmetric Encryption: RSA Attacks and Countermeasures

Lesky D.S. Anatias

April 18, 2008

Abstract

This paper discusses side-channel attacks based on Power Analysis. This approach utilizes physical side effects of using cryptographic functions in the real world. A side channel is a source of information that is inherent to a physical implementation of cryptographic functions. Research done in the last half of the 1990s has shown that the information transmitted by side channels, such as execution time, computational faults and power consumption, can be disadvantageous to the security of cryptosystem like RSA or AES. This paper surveys the techniques of Differential Power Analysis presented in [1] and [2] and shows how side channel information can be used to break implementations of RSA public key cryptosystem. Possible defenses against this type of side channel attacks are also discussed.

Keywords: Public key cryptography, asymmetric encryption, exponentiation, RSA, side-channel analysis, DPA

1 Introduction

In the last fifteen years, many kinds of cryptanalytic attack have begun to appear in the literature which targets a specific implementation details. Both timing attacks [3] and differential fault analysis [4] make assumptions about the implementation, and use additional information gathered from attacking certain implementations. In addition, failure analysis assumes a one-bit feedback from the implementation in order to break the underlying cryptographic primitive. More recently, differential power analysis or DPA [5] has been developed and applied to a number of cryptographic implementations.

This paper will discuss a side-channel attack based on differential power analysis. A side-channel attack occurs when an attacker is able to use some additional information leaked from the implementation of a cryptographic algorithm to analyze the function. Clearly, given enough side-channel information, it is trivial to break a cipher. An attacker who can, for instance, learn every input into every S-box in every one of DES's rounds can trivially calculate the key.

In addition, this paper also focuses on techniques applied to RSA asymmetric algorithms or public key cryptography. Public key cryptography has been widely used since its introduction by Diffie-Hellman and today’s most famous applications are RSA embedded.

The first power analysis attacks on the RSA algorithm were published by Thomas S. Messerges et al. [6]. Attack scenarios such as SEMD (“Single Exponent, Multiple Data”), MESD (“Multiple Exponent, Single Data”) and ZEMD (“Zero Exponent, Multiple Data”) were introduced. The ZEMD attack uses DPA techniques to compromise the bits of the private RSA exponent successively. This ZEMD attack is applied on the intermediate results during modular exponentiation. DPA attacks against RSA are considered as 'chosen ciphertext' attacks if applied at the RSA decryption and 'chosen plaintext' attacks if the DPA attacks are applied against the RSA signature.

Nevertheless this paper is also aimed to discuss dangerous security effect of this cryptanalytic type. In real-world systems, attackers obviously can cheat. Exploiting weaknesses in cryptographic implementations either by monitoring some "side-channel" of information out of the mechanism implementing the cryptographic primitive (such as timing or power consumption), or by altering some internal data inside that mechanism may feel like cheating, but that just makes their effects more devastating.

This paper starts with some background information on DPA, modular exponentiation, and RSA on Chinese Remainder Theorem (CRT) implementation in section 2. In the following section DPA Attack on Modular Exponentiation is described and discussed. In Section 4 DPA Attack on CRT Implementation is discussed. The countermeasures against these attacks are found in section 5. Finally, this paper is concluded in section 6.

2 Background

2.1 Differential Power Analysis

Differential power analysis or DPA is probably the most threatening attack to result from Kocher's research. To carry out a DPA attack, an adversary must have a number of power traces collected from a token as it repeatedly executes a cryptographic operation. The attack proceeds by deducing bits of the secret key, used in each operation, from the observed power consumption. An adversary must also have knowledge of either the inputs or outputs processed by the device during each operation. Usually, an encryption token will use the same key over multiple operations and any generated ciphertext can be freely obtained by an adversary.

The basic technique of DPA is as follows. Suppose an adversary is able to partition power traces from several cryptographic operations into two groups according to the intermediate value of some bit, b, which is calculated during each operation. This bit is manipulated during each operation and its value may affect the observed power consumption. If this is the case then the two groups of traces should show respectively different power biases at locations when b is manipulated. Averaging the traces in each group helps reduce any noise that may be obscuring these usually small biases. Plotting the difference of the two average traces reveals any locations in the traces where these biases occur.

As an example, Figure 1 below shows four traces prepared using known plaintexts entering a DES encryption function on another smart card. On top is the reference power trace showing the average power consumption during DES operations. Below are three differential traces, where the first was produced using a correct guess for K_s. The lower two traces were produced using incorrect values for K_s. These traces were prepared using 1000 samples (i.e. m = 103). Although the signal is clearly visible in the differential trace, there is a modest amount of noise.

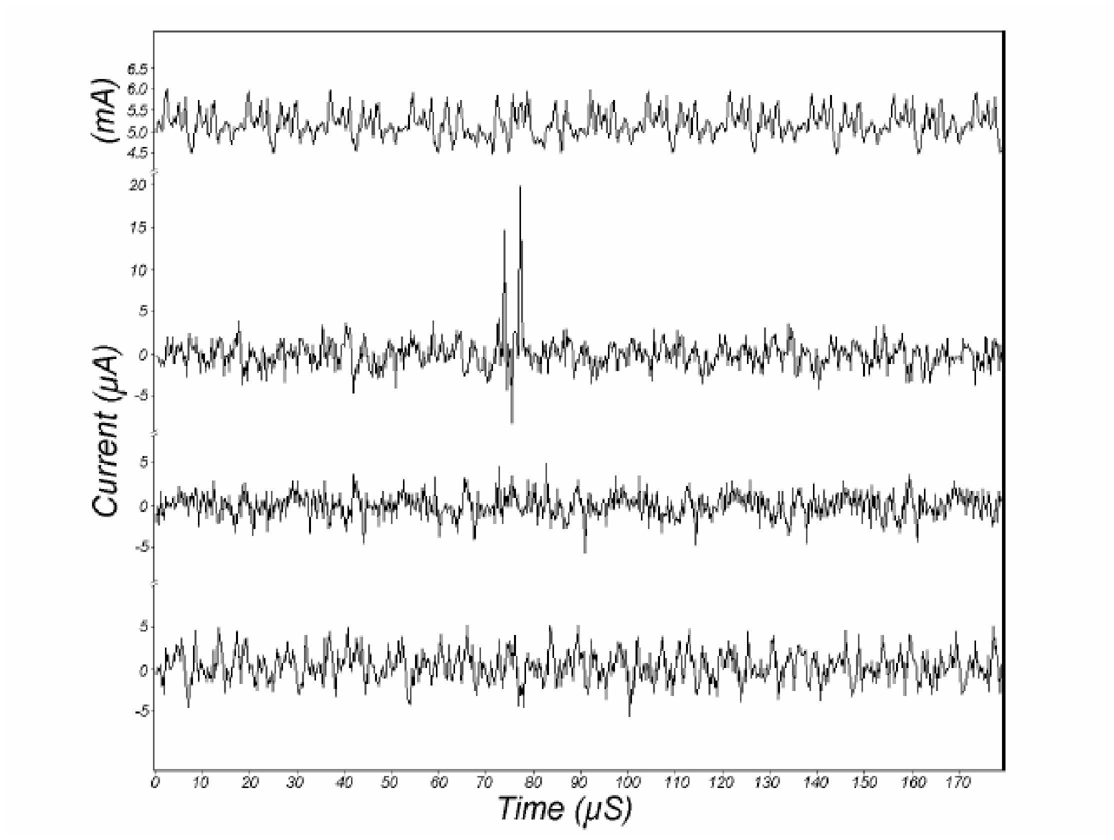


Figure 1: DPA traces, one correct and two incorrect, with power reference.

For a discussion and further explanation of differential power analysis attacks the reader is referred to their paper ([6]).

2.2 Modular Exponentiation

RSA cryptosystem is the most famous public-key cryptosystem in practical use, and it is implemented in many security applications. Especially, security solutions with smart cards have been focused because of its flexibility and security. To describe the previously mentioned RSA cryptosystem, this section and the following one will discuss the underlying algorithms of this cryptosystem. For more information on the basic of RSA algorithm the reader is referred to appendix A.

The main computational operation of RSA without the Chinese Remainder Theorem or CRT is the modular exponentiation of a message, *m*. The RSA without CRT scheme has more computational loads than other signature schemes, DSS and ECDSA. In order to perform a modular exponentiation $c = a^b \bmod m$ in $\mathbb{Z}m$, the bitwise representation $b = [b_{n-1}b_{n-2} \cdot \cdot \cdot b_1b_0]$ is used. The 'square - multiply' algorithm evaluates this representation either starting from the least significant bit b_0 (algorithm 1) or from the most significant bit b_{n-1} (algorithm 2).

Algorithm 1:
```
t := a
c := 1
for k := 0 to n-1 do {
    if b[k]=1 then c := c*t mod m
    t := t*t mod m
}
return c
```

Algorithm 2:
```
c := 1
for k := n-1 down to 0 do {
    c := c*c mod m
    if b[k]=1 then c := c*a mod m
}
return c
```

2.3 RSA on CRT Implementation

The RSA with CRT algorithm was proposed to speed up the original RSA signature or decryption computation [7]. To reduce calculation time of a RSA exponentiation with the secret key one can solve a simultaneous system of modular congruencies. The existence of such a solution is ensured by the Chinese Remainder Theorem (CRT).

Based on Fermat's little theorem, the precalculation of the reduced secret exponent values $d_p = d \bmod (p - 1)$ and $d_q = d \bmod (q - 1)$ is performed. Using Algorithm 1 or Algorithm 2 one calculates then $v_1 = x^{dp} \bmod p$ and $v_2 = x^{dq} \bmod q$. For the CRT algorithm according to Garner [1] (Algorithm 3) one precalculated multiplicative inverse $P_q = p^{-1} \bmod q$ is needed:

Algorithm 3:
```
u := (v2-v1)*Pq mod q
y := v1+u*p
return y
```

Alternatively, the CRT algorithm according to Gauss [1] (Algorithm 4) uses the two precalculated multiplicative inverses $P_q = p^{-1} \bmod q$ and $Q_p = q^{-1} \bmod p$ and a final reduction modulo N:

Algorithm 4:
```
y := (v1*q*Qp + v2*p*Pq) mod N
return y
```

Note that during exponentiation a modular reduction modulo a secret value instead of a public one takes place. This is used for the attack described below.

3 DPA Attack on Modular Exponentiation

At first, in order to apply DPA to RSA the attacker should have the possibility to randomly vary the input data x of the RSA implementation. Single power consumption measurements $P(x, t)$ of the cryptographic module are typically carried out with a digital oscilloscope and can be stored on a server.

3.1 Hypothesis

If the RSA implementation uses a ’top-down square-multiply’ algorithm as in Algorithm 2, the key hypotheses are set up on the next bits of the exponent to proceed. The intermediate results of the exponentiation algorithm preceding these bits can be determined offline. In case that the second hypothesis is correct, correlations are present for both key hypotheses as the result of the first hypothesis is an intermediate result of the second hypothesis. The correlations for the correct key hypothesis appear last.

Generally, it is even more useful to set up the key hypotheses on the sequence of elementary operation (squarings ‘S’ and multiplications ‘M’). The simplest hypotheses would be

1. ‘the next 2 modular multiplication units are composed of ‘SM’,
2. ‘the next 2 modular multiplication units are composed of ‘SS’, and – in case that the previous correct hypothesis ends up with a ‘S’ – additionally
3. ‘the next 2 modular multiplication units are composed of ‘MS’.

In general, this set-up of key hypotheses is of interest if we deal with a greater number of key hypotheses.

3.2 Selection Function and Correlation

In a power analysis the attacker knows or assumes a model for a dependency of the power consumption on the value of intermediate data. A common model is that the power consumption correlates with the Hamming weight of intermediate data (see [1], [5]).

The selection function has to be calculated on the intermediate result of each key hypothesis that is applied. Intermediate results of the RSA exponentiation are generally of the same bit length as the modulus used. The n-bit bus architecture of the RSA coprocessor used determines the number of bits that are taken into account for the Hamming weight.

DPA selection functions $d(x)$ should use the bit-width of the bus architecture to setup functions on the Hamming weight $W(x)$ of intermediate data. A simple selection function $d(x)$ assesses all intermediate data values that have a greater Hamming weight than the n-bit expectation value $E(n) = n/2$ with 1, all values with smaller Hamming weights than $E(n)$ with -1 and to ignore all values that meet the expectation value $E(n)$.

$$d(x) = \begin{cases} -1, \text{ if } W(x) < E(n) \\ 0, \text{ if } W(x) = E(n) \\ +1, \text{ if } W(x) > E(n) \end{cases}$$

The selection function $d(x)$ can be refined in the linear model below

$$d(x) = W(x) - E(n).$$

The easiest selection function is the Hamming weight of intermediate data, or for example the Hamming weight of just a byte of transported data. In this case, DPA identifies the correct key hypothesis by assessing the absolute maximum of the correlation coefficients for each key hypothesis. The correlation is carried out between the result of the selection function $d(x, j)$ on the base of the key hypothesis j and the input data x and the power consumption $P(x, t)$ of the single measurements as a function of x and the time t. The variable t could be narrowed to a small time interval if simple power characteristics of the implementation are observable. The number i runs through all single measurements. The correlation coefficient $c(t, j)$ has to be assessed for each key hypothesis j as presented below.

$$c(t,j) = \frac{\sum_i (d(x_i,j) - \overline{d(x_i,j)})(P(x_i,t) - \overline{P(x_i,t)})}{\sqrt{\sum_i (d(x_i,j) - \overline{d(x_i,j)})^2}\sqrt{\sum_i (P(x_i,t) - \overline{P(x_i,t)})^2}}$$

It will be near zero if there aren't any correlations between the selection function $d(x, j)$ and $P(x, t)$ and will approaches 1 in case of a strong correlation $c(t, j)$ at some specific points in time.

For a discussion and further explanation of differential power analysis attacks against non-CRT implementation the reader is referred to their paper ([1] and [8]).

4 DPA Attack on CRT Implementation

4.1 Hypothesis

In contrast to the RSA implementation of 'top-down square-multiply' algorithm that has to correlate on the intermediate results of the modular exponentiation algorithm, the attack on the CRT implementation attacks the modular reduction modulo one of the primes performed prior to the CRT exponentiation. It exploits power consumption signals that are caused by the processing and data bus transfers of the residue.

As mentioned in [1] the DPA attack on the CRT implementation uses measurement series with input values of RSA that are equidistant. It assumes that input values can be chosen by the attacker. At the first measurement series a starting value x_0 is chosen and the following input values are generated by decrementing the previous input value by 1. It also assumes that each series contains m elements. Series are numbered with k. Within the second series the input values have a distance of 256: $x_0, x_0 - 1 \cdot 256, x_0 - 2 \cdot 256, x_0 - 3 \cdot 256, ..., x_0 - m \cdot 256$. Other series follow with step size $256k$ until the exponent k reaches the size of the prime to be attacked. Thus, for each series k it defines the i-th value as follow

$$x_{\mathrm{i}} = x_0 - i \cdot (256)^k$$

The DPA attack on the modular reduction sets up hypotheses on the remainder r after the reduction modulo the prime q. The aim of the first measurement series with the distance 1 of the input values is to compromise the least significant byte of the remainder r_0 that fulfills

$$r_0 \equiv x_0 \bmod q$$

Then it also defines

$$r_i = x_i \bmod q$$

The further measurement series aim to compromise the *k*-th byte of the remainder r_0, correspondingly.

As presented in [1], to demonstrate the usage of the measurement series with an equal distance of 1, it assures that all input values x_i of the first measurement series have a remainder with the prime (in the following we assume that the prime q is going to be attacked) that does not equal zero. Then it excludes the unlikely case of crossing a multiple of q by calculating the GCD with the public modulus N and all input values x_i of the first measurement series: $gcd(x_i,N)$.

There are 256 hypotheses $H_{j0}(0 \leq j \leq 255)$ on the value of the least significant byte of r_0

$$H_{j0} \text{ is } \{r_0 \bmod 256 = j\}$$

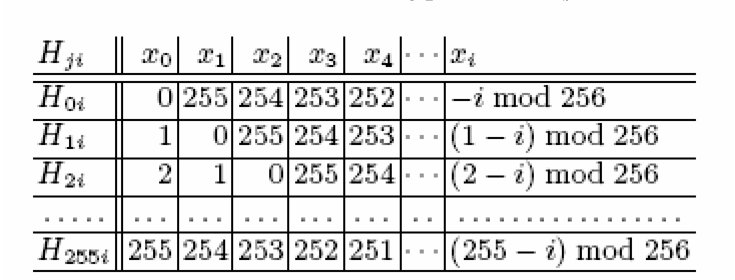
that are going to be analysed with DPA.

All values of r_i are related to the value r_0. As the input values x_i are equally distant the difference between r_0 and r_i directly gives the value of the last byte of the remainder for each hypothesis

$$H_{ji} \text{ is } \{r_i \bmod 256 = (j - i) \bmod 256\}$$

The corresponding value of H_{ji} *i* can be read out from the Table 1.

Table 1: Table of hypotheses H_{ji}

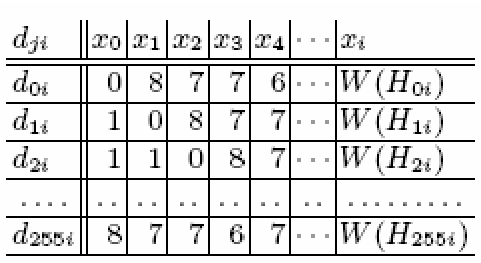


H_{ji}	x_0	x_1	x_2	x_3	x_4	$\cdots$	x_i
H_{0i}	0	255	254	253	252	$\cdots$	$-i \bmod 256$
H_{1i}	1	0	255	254	253	$\cdots$	$(1-i) \bmod 256$
H_{2i}	2	1	0	255	254	$\cdots$	$(2-i) \bmod 256$
.....	...	...	...	...	...	..	....
H_{255i}	255	254	253	252	251	$\cdots$	$(255-i) \bmod 256$

The correlation is carried out with the Hamming weight $W(x)$ for each hypothesis H_{ji}.The selection function $d(x, j)$ is therefore based on 8 bit (see Table 2).

The strongest results are expected for that value of j where the hypothesis corresponds to the reality. This value is called f_0 from now on. The cyclic property of H_{ji} yields secondary correlation peaks. The second strongest correlations are expected at the hypothesis $H_{j\pm128}$. The third strongest correlations should be at the two hypotheses $H_{j\pm64}$. Therefore there are additional indices of the correct hypothesis.

Table 2: Table of the selection functions $d(x_i, j)$ on the base of hypotheses H_{ji} using the 8-bit Hamming weight $W(x)$.



d_{ji}	x_0	x_1	x_2	x_3	x_4	$\cdots$	x_i
d_{0i}	0	8	7	7	6	$\cdots$	$W(H_{0i})$
d_{1i}	1	0	8	7	7	$\cdots$	$W(H_{1i})$
d_{2i}	1	1	0	8	7	$\cdots$	$W(H_{2i})$
....	..	..	..	..	..	..	....
d_{255i}	8	7	7	6	7	$\cdots$	$W(H_{255i})$

As result of the first measurement series it is found that $(x_0 - f_0) \bmod q$ is divisible by 256.

For a further explanation of the above general DPA attack that is referred as "MRED" (Modular Reduction using Equidistant Data) the reader is referred to their paper ([1]).

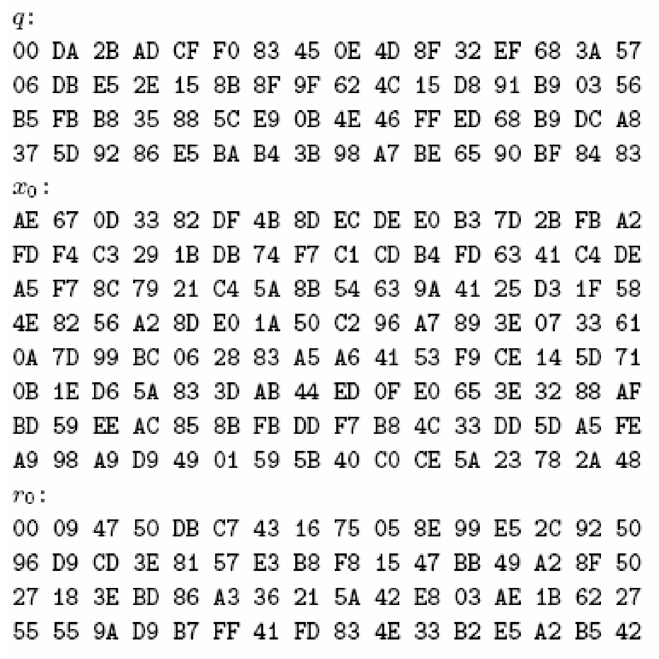
4.2 Result

In [1] the results that are expected using this DPA attack on the CRT implementation are demonstrated using simulated measurement data. The generation of these data is based on the power leakage model that the power consumption $P(x, t)$ at a certain point in time t can be split into a power contribution that varies with the Hamming weight of the data x processed, into a power consumption that represents a constant portion and a power consumption that is caused by noise.

In this paper, the underlying bus-architecture is chosen to be 8 bit. The generation of simulated measurement data gives an output file for each exponentiation that contains the Hamming weight of all intermediate data processed. These output files replace the single measurement data files. The number of bits used for the calculation of the Hamming weight is given by the bus-architecture.

The starting value x_0 was chosen randomly as 128 byte value. The value of prime q was 63 byte long.

The test values used are the following.



```
q:
00 DA 2B AD CF F0 83 45 0E 4D 8F 32 EF 68 3A 57
06 DB E5 2E 15 8B 8F 9F 62 4C 15 D8 91 B9 03 56
B5 FB B8 35 88 5C E9 0B 4E 46 FF ED 68 B9 DC A8
37 5D 92 86 E5 BA B4 3B 98 A7 BE 65 90 BF 84 83
x0:
AE 67 0D 33 82 DF 4B 8D EC DE E0 B3 7D 2B FB A2
FD F4 C3 29 1B DB 74 F7 C1 CD B4 FD 63 41 C4 DE
A5 F7 8C 79 21 C4 5A 8B 54 63 9A 41 25 D3 1F 58
4E 82 56 A2 8D E0 1A 50 C2 96 A7 89 3E 07 33 61
0A 7D 99 BC 06 28 83 A5 A6 41 53 F9 CE 14 5D 71
0B 1E D6 5A 83 3D AB 44 ED 0F E0 65 3E 32 88 AF
BD 59 EE AC 85 8B FB DD F7 B8 4C 33 DD 5D A5 FE
A9 98 A9 D9 49 01 59 5B 40 C0 CE 5A 23 78 2A 48
r0:
00 09 47 50 DB C7 43 16 75 05 8E 99 E5 2C 92 50
96 D9 CD 3E 81 57 E3 B8 F8 15 47 BB 49 A2 8F 50
27 18 3E BD 86 A3 36 21 5A 42 E8 03 AE 1B 62 27
55 55 9A D9 B7 FF 41 FD 83 4E 33 B2 E5 A2 B5 42
```

The correct value f_0 of the least significant byte of x_0 mod q is in this example $42h$ represent 66 in decimal representation.

The DPA calculation reveals a list of the best 17 candidates (out of 256 candidates) for the correct value of f_0 on the base of 256 single measurements. Besides to the correct value 66 (decimal notation) secondary positive correlation signals with decreasing amplitudes appear at the relative displacements of ± 128, ± 64, ± 32, ± 16, ± 8, ± 4, ± 2 and ± 1 of the correct hypothesis 66. If the number of single measurements is not a multiple of 256 the correlation coefficients of the secondary positive correlation coefficients differ slightly.

In the Figure 2 both positive and negative correlation coefficients are taken into account. Negative correlation coefficients occur mainly at small correlation amplitude. Strong correlation signals are caused by positive correlation coefficients.

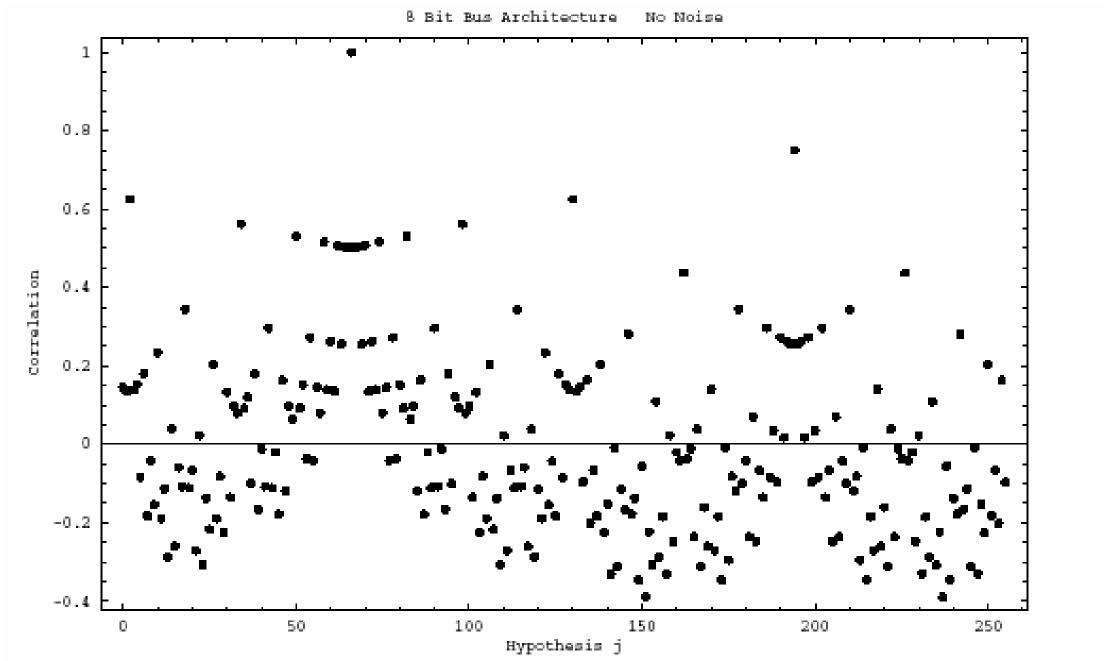


Figure 2: Graphical representation of the correlation coefficients on the base of 256 single measurements.

For a discussion and further result of differential power analysis attacks against CRT implementation of RSA on 32-bit architecture the reader is referred to their paper ([1]).

5 Countermeasures

Countermeasures against physical attacks range among a large variety of solutions. However, in the present, no single technique allows to provide perfect security, even considering a particular attack only. Protecting implementations against physical attacks consequently intends to make the attacks harder. In this context, the implementation cost of a countermeasure is of primary importance and must be evaluated with respect to the additional security obtained. The exhaustive list of all possible solutions to protect cryptographic implementations from physical opponents would deserve a long survey in itself.

In this section, we will only discuss a few exemplary countermeasures in order to illustrate that security can be added at different levels. We refer the reader to [1] and [2] for a more comprehensive review of existing countermeasures.

In general, the mechanism described in [5] and [8] can be used to prevent these 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. However, when dealing with MRED attack, we have to consider three basic assumptions, namely

1. a sufficient number of single measurements can be collected,
2. the input data x can be varied arbitrarily to construct equidistant input data, and
3. $(x_0 - i \cdot (256)^k) \bmod q$ holds $(r_0 - i \cdot (256)^k)$ at least for a subgroup of single measurements.

5.1 Counter Usage

The first assumption deals with the number of single measurements that are needed for this DPA attack. A DPA attack against 1024 bit RSA key demands for about $30.000 < n < 300.000$ single measurements. The upper boundary of single measurements may conflict with physical constraints of smart cards. A general countermeasure to prevent these kind of statistical attacks is an usage counter for the number of RSA exponentiations. To secure the RSA decryption an additional failure counter can be implemented if a check of padding formats fails.

5.2 Padding Format

The second assumption affects RSA signing ("chosen plaintext"), but not the RSA decryption ("chosen ciphertext"). The second assumption fails if the attacker has to deal with padding formats in case of digital signature applications. Typical padding formats used limit the range of variable data to the least significant 20 bytes of data that is the outcome of a hashing function. At the presence of padding formats MRED will reveal at maximum the least significant 20 bytes of the remainder of both primes p and q.

5.3 Message Blinding

The third assumption can be destroyed by message blinding. Multiplicative message blinding scheme as e. g. proposed by [3] use pairs (v_i, v_k) that are used for the blinding of the input data and unblinding of the result. This multiplicative blinding is applicable to prevent the likeliness of the third assumption. Another approach uses random blinding technique on message as proposes by [2].

6 Conclusion

This paper discusses a number of DPA attack on the remainder that can be applied at a CRT implementation of RSA to compromise one of the secret RSA primes. The basic assumption for MRED is that $(x_0 - i \cdot (256)^k) \bmod q$ holds $(r_0 - i \cdot (256)^k)$ at least for a subgroup of single measurements. The results of this MRED attack are shown on the base of simulated measurement data. One of the countermeasures against MRED should include the use of multiplicative blinding schemes to protect the reduction modulo a secret prime.

The discussion clearly underlines that side-channel attacks especially DPA constitute a very significant threat for actual designers. However, it must be noted that the actual implementation of physical attacks (and more specifically, their efficiency) may be significantly platform-dependent. Moreover, the presented techniques usually require a certain level of practical knowledge, somewhat hidden in our abstract descriptions.

In conclusion, good security may nevertheless be obtained by a merely combination of these countermeasures. However, it is more an engineering process than a scientific derivation of provably secure schemes. Theoretical security against physical attacks is a large scope for further research.

References

[1] Bert den Boer, K. Lemke, and G. Wicke, “A DPA attack against the modularreduction within a CRT implementation of RSA,” *CHES’02*, LNCS 2523, Springer-Verlag, 2002, pp. 228 – 243.

[2] C. Kim, J. Ha, S. Moon, S. Yen, and S. Kim, “A CRT-Based RSA Countermeasure Against Physical Cryptanalysis”, HPCC 2005, LNCS 3726, Springer-Verlag Berlin Heidelberg 2005, pp. 549 – 554.

[3] P. Kocher, “Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems”, *Advances in Cryptology CRYPTO '96 Proceedings*, Springer-Verlag, 1996, pp. 104 – 113.

[4] E. Biham and A. Shamir, “Differential Fault Analysis of Secret Key Cryptosystems”, *Advances in Cryptology CRYPTO '97 Proceedings*, Springer-Verlag, 1997, pp. 513 – 525.

[5] P. Kocher, J. Jaffe and B. Jun, “Differential Power Analysis”, *Proceedings of Advances in Cryptology – CRYPTO ’99, Lecture Notes in Computer Science*, Vol. 1666, Springer, Berlin 1999, pp. 388–397.

[6] T. S. Messerges, E. A. Dabbish and R. H. Sloan, “Power Analysis Attacks of Modular Exponentiation in Smartcards”, *Proceedings of Workshop on Cryptographic Hardware and Embedded Systems*, Springer, Lecture Notes in Computer Science, Vol. 1717, Springer, Berlin 1999, pp. 144–157.

[7] A.J. Menezes, P.C.van Oorchot, and S.A. Vanstone, “Handbook of applied cryptography”, CRC Press, 1997.

[8] M. Aigner and E. Oswald, “Power Analysis Tutorial”, Institute for Applied Information Processing and Communication University of Technology Graz, Austria, 2000.

A RSA

RSA, named after the initials of its authors, Rivest, Shamir and Adleman is probably the most famous asymmetric encryption primitive. It works as follows:

1. Alice chooses two large prime numbers p and q and computes their product $n = pq$ and $\varphi(n) = (p - 1)(q - 1)$.
2. She also chooses a value e that has no common factor with $\varphi(n)$ and computes $d = e{-}1 \bmod \varphi(n)$.
3. Alice publishes (n, e) as her public key, and keeps d as her private key.
4. To send her a message m (with $0 \leq m < n$), Bob computes $c = me \bmod n$.
5. Alice decrypts c by computing $cd \bmod n$. By Euler’s theorem, it can easily be shown that the result is equal to m.

We refer the reader to [7] for more information on RSA.