Side-Channel Analysis

Marc Joye Francis Olivier

Gemplus, Card Security Group, France
{marc.joye, francis.olivier}@gemplus.com

1 Introduction

Electronic devices have to comply with consumption constraints especially on autonomous equipments, like mobile phones. Power analysis has been included into most certification processes regarding products dealing with information security such as smart cards.

The electrical consumption of any electronic device can be measured with a resistor inserted between the ground or Vcc pins and the actual ground in order to transform the supplied current into a voltage easily monitored with an oscilloscope.

Within a micro-controller the peripherals consume differently. For instance writing into non-volatile memory requires more energy than reading. Certain chips for smart cards enclose a crypto-processor, i.e., a particular device dedicated to specific cryptographic operations, which generally entails a consumption increase. The consumption trace of a program running inside a micro-controller or a microprocessor is full of information. The signal analysis may disclose lots of things about the used resources or about the process itself. This illustrates the notion of *side channel* as a source of additional information.

Basically a power consumption trace exhibits large scale patterns most often related to the structure of the executed code. The picture below (Fig. 1) shows the power trace of a smart-card chip ciphering a message with the Advanced Encryption Standard (AES). The ten rounds are easily recognised with nine almost regular patterns first followed by a shorter one.

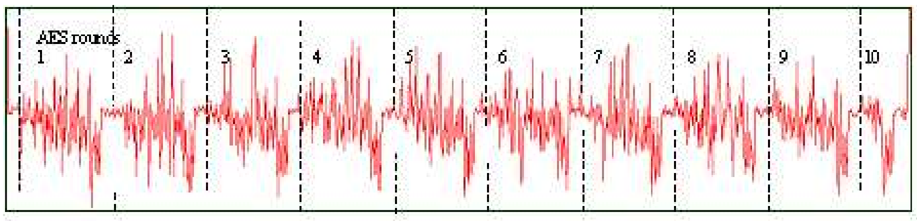

Figure 1: AES power trace

Zooming into a power signal exhibits a local behaviour in close relationship with the silicon technology. At the cycle scale, the consumption curve looks roughly like a capacitive charge and discharge response.

A careful study of several traces of a same code with various input data shows certain locations where power trace patterns have different heights. The concerned cycles indicate some data dependence also called *information leakage*. They may be magnified by a variance analysis over a large number of executions with random data. For instance, by ciphering many random plaintexts with a secret-key algorithm, it is possible to distinguish the areas sensitive to input messages from the constant areas that correspond to the key schedule.

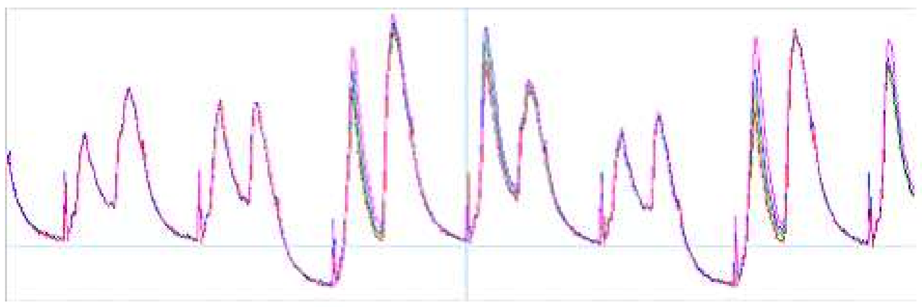

Figure 2: Information leakage

2 Information Leakage Model

The characterisation of data leakage (namely, finding the relationships between the data and the variability of consumption) has been investigated by several researchers. The most common model consists in correlating these variations to the Hamming weight of the handled data, i.e., the number of nonzero bits. Such a model is valid for a large number of devices. However it can be considered as a special case of the transition model which assumes that the energy is consumed according to the number of bits switched for going from one state to the next one. This behaviour is represented by the Hamming distance between the data and some *a priori* unknown constant, i.e., the Hamming weight of the data XOR-ed with this constant.

As shown in the next picture (Fig. 3), for an 8-bit micro-controller, the transition model may seem rough but it suffices to explain many situations, provided that the reference constant state is known. In most microprocessors this state is either an address or an operating code. Each of them has a specific binary representation and therefore a different impact in the power consumption: this is why each cycle pattern is most often different from its neighbours.

Some technologies systematically go through a clear “all-zeros” state that explains the simpler Hamming-weight model.

3 Statistical Analyses

With information leakage models in mind, it is possible to design statistical methods in order to analyse the data leakage. They require a large amount of power traces assigned to many executions of the same code with varying data, generally at random, and make use of statistical estimators such as averages,

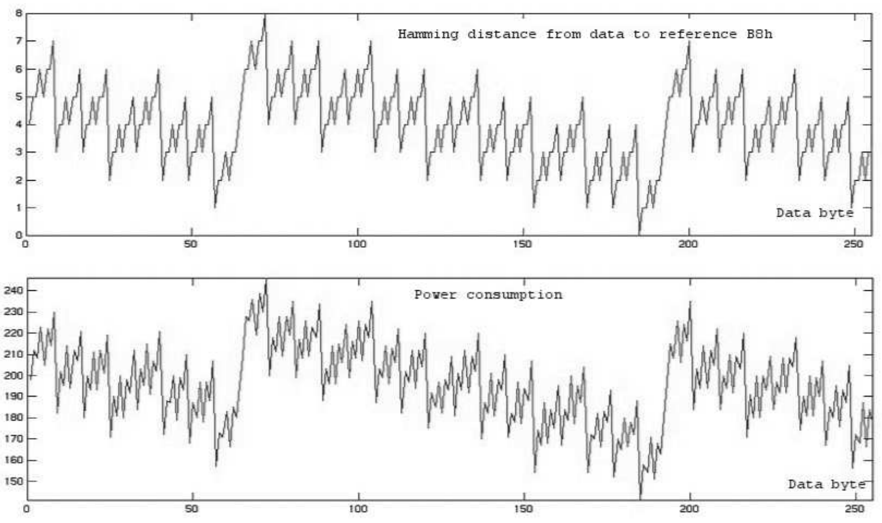

Figure 3: Transition model

variances and correlations. The most famous method is due to Paul Kocher *et al.* and is called *Differential Power Analysis* (DPA).

Basically the purpose of DPA is to magnify the effect of a single bit inside a machine word. Suppose that a random word in a Ω-bit processor is known and uniformly distributed. Suppose further that the associated power consumption obeys the Hamming-weight model. On average the Hamming weight of this word is $\Omega/2$. Given N words, two populations can be distinguished according to an arbitrary selection bit: the first population, $\mathcal{S}_0$, is the set of t words whose selection bit is 0 and the second population, $\mathcal{S}_1$, is the set of $N-t$ words whose selection bit is 1. On average, the words of set $\mathcal{S}_0$ will have a Hamming weight of $(\Omega-1)/2$ whereas the words of set $\mathcal{S}_1$ will have a Hamming weight of $(\Omega+1)/2$. The same bias can be seen through the corresponding power consumption traces since it is supposed to be correlated with the Hamming weight of the data. Let C_0 and C_1 respectively denote the averaged power consumption traces of on the blue curvesets $\mathcal{S}_0$ and $\mathcal{S}_1$. The *DPA trace* is defined as the difference $C_0 - C_1$.

The resulting DPA curve has the property of erecting bias peaks at moments when the selection bit is handled. It looks like noise everywhere else: indeed, the constant components of the signal are cancelled by the subtraction whereas dynamic ones are faded by averaging, because they are not coherent with the selection bit.

This approach is very generic and applies to many situations. It works similarly with the transition model. Of course the weight of a single selection bit is relatively more important in processors with short words like 8-bit chips. If the machine word is larger, the same DPA bias can be obtained by increasing the number of trials.

A first application of DPA is called *bit tracing*. It is a useful reverse engineering tool for monitoring a predictable bit during the course of a process. In principle a DPA peak rises up each time it is processed. This brings a lot of information about an algorithm implementation. To achieve the same goal Paul

Fahn and Peter Pearson proposed another statistical approach called *Inferential Power Analysis* (IPA). The bits are inferred from the deviation between a single trace and an average trace possibly resulting from the same execution: for instance the average trace of a DES round can be computed over its sixteen instances taken from a single execution. IPA does not require the knowledge of the random data to make a prediction on a bit value. But as counterpart it is less easy to implement and the interpretation is less obvious.

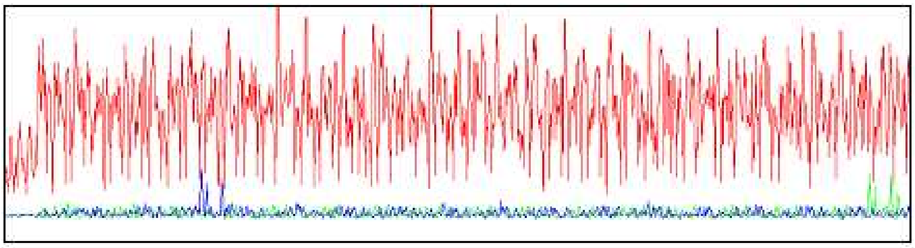

Figure 4: Bit tracing (Upper curve: power consumption of a single execution; 2 lower curves: DPA curves respectively tracing the first and last data bit of a targetted process.)

After Paul Kocher, Thomas Messerges *et al.* have proposed to extend DPA by considering multiple selection bits in order to increase the signal to noise ratio (SNR). If the whole machine word is taken into account a global approach consists in considering the transition model as suggested by Jean-Sébastien Coron *et al.*

4 From Power Analysis to Power Attacks

Obviously if the power consumption is sensitive to the executed code or handled data, critical information may leak through power analysis. This section explains how to turn a side-channel analysis into an attack.

4.1 SPA-type attacks

A first type of power attacks is based on *Simple Power Analysis* (SPA). For example, when applied to an *unprotected* implementation of an RSA cryptosystem, such an attack may recover the whole private key (i.e., signing or decryption key) from a single power trace.

Suppose that a private RSA exponentiation, $y = x^d \bmod n$, is carried out with the square-and-multiply algorithm. This algorithm processes the exponent bits from left to right. At each step there is a squaring, and when the processed bit is 1 there is an additional multiplication. A straightforward (i.e., unprotected) implementation of the square-and-multiply algorithm is given below.

The corresponding power curve exhibits a sequence of consumption patterns among which some have a low level and some have a high level. These calculation units are assigned to a crypto-processor handling n-bit arithmetic. Knowing that a low level corresponds to a squaring and that a high level corresponds to a multiplication, it is fairly easy to read the exponent value from the power trace:

```
$k \leftarrow \text{bitsize}(d)$
$y \leftarrow x$
**for** $i = k - 2$ **downto** $0$ **do**
    $y \leftarrow y^2 \pmod{n}$
    **if** (bit $i$ of $d$ is $1$) **then** $y \leftarrow y \cdot x \pmod{n}$
**endfor**
**return** $y$
```

Figure 5: Square-and-multiply exponentiation algorithm

- a low-level pattern followed by another low-level pattern indicates that the exponent bit is 0, and
- a low-level pattern followed by a high-level pattern indicates that the exponent bit is 1.

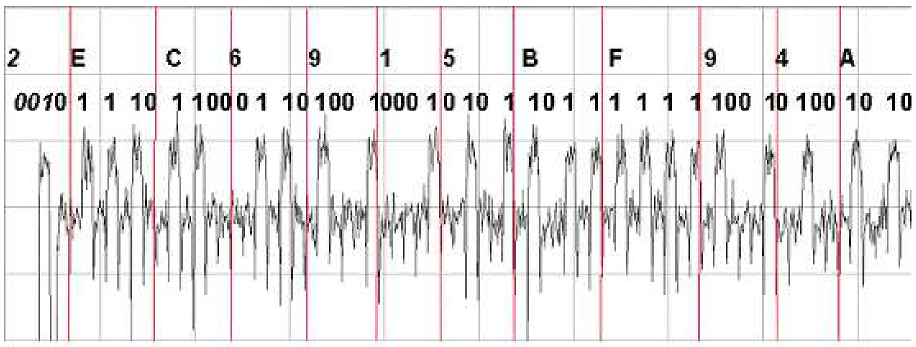

Figure 6: SPA trace of the basic square-and-multiply algorithm

This previous picture also illustrates why the Hamming weight of exponent d can be disclosed by a timing measurement.

4.2 DPA-type attacks

Historically, DPA-type attacks —that is, power attacks based on *Differential Power Analysis* (DPA)— were presented as a means to retrieve the bits of a DES key.

At the first round of DES, the output nibble of the i^{th} S-box ($1 \le i \le 8$) can be written as $\mathrm{S}_i(M \oplus K)$ where

- M is made of 6 bits constructed from the input message after IP- and E-permutations: it has to be chosen at random but is perfectly known and predictable, and
- K is a 6-bit sub-key derived from the key scheduling.

Rising up a DPA bias would require the knowledge of the output nibble. As K is unknown to the adversary this is not possible. But sub-key K can be easily exhausted as it can take only $2^6 = 64$ possible values. Therefore the procedure consists in reiterating the following process for $0 \le \widehat{K} \le 63$:

1. form sets $\mathcal{S}_0 = \{M \mid g(\text{S-box}_i(M \oplus \widehat{K})) = 0\}$ and $\mathcal{S}_1 = \{M \mid g(\text{S-box}_i(M \oplus \widehat{K})) = 1\}$ where selection function g returns the value of a given bit in the output nibble; and
2. compute the corresponding DPA curve.

In principle the bias peak should be maximised when the guess $\widehat{K}$ is equal to the real sub-key K. Then inverting the key schedule permutation leads to the value of 6 key bits. In other words the DPA operator is used to validate sub-key hypotheses. The same procedure applies to the 7 other S-boxes of the DES. Therefore the whole procedure yields $8 \times 6 = 48$ key bits. The 8 remaining key bits can be recovered either by exhaustion or by conducting a similar attack on the second round.

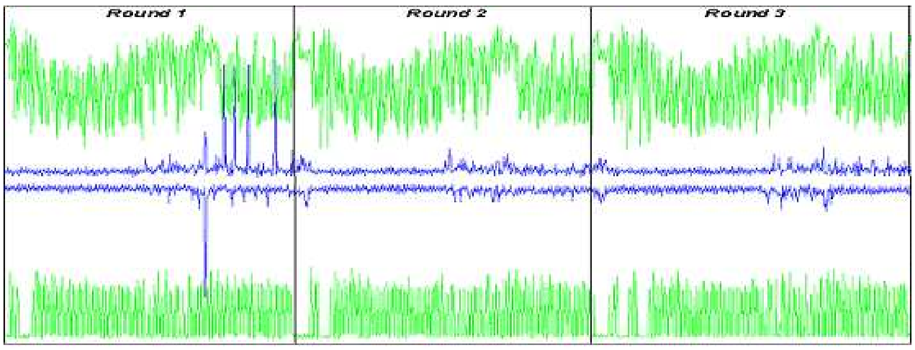

Figure 7: DPA trace of the 3 first rounds of DES (2 upper (resp. lower) curves: power consumption curve of maxima (resp. minima) of a single execution and DPA curve of maxima (resp. minima).)

The main feature of a DPA-type attack resides in its genericity. Indeed it can be adapted to many cryptographic routines as soon as varying and known data are combined with secret data through logical or arithmetic operations.

A similar attack can be mounted against the first round of AES; the difference being that there are 16 byte-wise bijective substitutions and therefore 256 guesses for each. Finally, we note that DPA-type attacks are not limited to symmetric algorithms, they also apply to certain (implementations of) asymmetric algorithms, albeit in a less direct manner.

4.3 Other attacks

Amongst the other statistical attacks, IPA is more difficult and less efficient. Its purpose is to retrieve key bits without knowing the processed data. It proceeds by comparing the power trace of a DES round with an average power trace computed for instance over the 16 rounds. In principle key bits could be inferred this way because the differential curve should magnify the bits deviation where they are manipulated.

Dictionary (or template) attacks can be considered as a generalisation of IPA to very comfortable but realistic situations. They have been widely studied in the field of smart cards where information on secret key or personal identification numbers (PIN) could potentially be extracted. They consist in building a complete dictionary of all possible secret values together with the corresponding side-channel behaviour (e.g., power trace) when processed by the device (e.g., for authentication purpose). Then a secret value embedded in a twin device taken from the field can be retrieved by comparing its trace and the entries of the dictionary.

In practice, things do not happen that easily for statistical reasons and application restrictions. Only part of the secret is disclosed and the information leakage remains difficult to fully exploit.

Finally, in addition to power consumption, other side channels can be considered; possible sources of information leakage include running time or electromagnetic radiation.

5 Countermeasures

The aforementioned attacks have all been published during the second half of the 90's. In front of this new threat the manufacturers of cryptographic tokens have designed a large set of dedicated countermeasures especially to thwart statistical attacks like DPA. All the related research activity has now resulted in tamper resistant devices widely available on the market. It has given rise to the new concept of "secure implementation" which states that information leakage is not only due to the specification of an application (cryptographic processing or whatever), but also to the way it is implemented.

If information leaks through a physical side-channel there are two defensive strategies. The first consists in decorrelating the secret data from the side-channel. The second consists in decorrelating the side-channel from the secret data. The borderline between both is sometimes fuzzy but roughly speaking, the former is rather software oriented and intends to mask the data since they have to leak anyway, whereas the latter is more hardware oriented and intends to shut the side-channel physically in order to make the device tamper-resistant.

Chip manufacturers have introduced into their hardware designs many security features against power attacks. They are stricto sensu countermeasures since they aim at impeding the power measurement and make the recorded signal unworkable.

- Some countermeasures consist in blurring the signal using smoothing techniques, additive noise or de-synchronisation effects. These countermeasures are poorly efficient against SPA working on broad scale traces. They are rather designed against statistical attacks. They may require some complementary circuits to generate parasitic components into the consumed current. De-synchronisation aims at misaligning a set of power traces by the means of unstable clocking or the insertion of dummy cycles at random, making the statistical combination of several curves ineffective.

- Other countermeasures rather intend to decrease or cancel the signal at the source. Reduction is a natural consequence of the shrinking trend in silicon industry that diminishes the power consumption of each elementary gate. More interesting (and expensive) is the emerging technology called "pre-charged dual rail logic" where each bit is represented by a double circuitry. At a given time a logical 0 is represented physically by a 01, and a logical 1 by 10. The transition to the next time unit goes through a physical 00 or 11 state so that the same amount of switching occurs whatever the subsequent state is. Consequently if both rails are perfectly balanced, the overall consumption of a microprocessor does not depend on the data anymore.

Software countermeasures enclose a large variety of techniques going from the application level to the most specific algorithmic tricks. One can classify them into three categories: application constraints, timing counter-measures and data masking.

- Application constraints represent an obvious but often forgotten means to thwart statistical analyses. For instance DPA requires known data with a high variability. An application wherein an input challenge (or an output cryptogram) would be strictly formatted, partially visible and constrained to vary within hard limits (like a counter) would resist to DPA fairly well.

- Timing countermeasures mean the usage of empirical programming tricks in order to tune the time progress of a process. A critical instruction may have its execution instant randomised by software: if it never occurs at the same time, statistical analysis becomes more difficult. Conversely other situations require the code to be executed in a constant time, in order to protect it from SPA or timing analysis. For instance a conditional branch may be compensated with a piece of fake code with similar duration and electrical appearance.

- Data masking (also known as whitening or randomisation), covers a large set of numerical techniques designed by cryptographers and declined in various manners according to the algorithm they apply to. Their purpose is to prevent the data from being handled in clear and to disable any prediction regarding their behaviour when seen through the side channel. For example, the modular exponentiation $y = x^d \bmod n$ (as used in the RSA cryptosystem) can be evaluated as:

  $$y = \{(x + r_1 n)^{d + r_2 \varphi(n)} \bmod r_3 n\} \bmod n$$

  for randoms r_i and where ϕ denotes Euler totient function.

To illustrate how fuzzy the borderline between hardware and software countermeasures can be, we have mentioned that for instance de-synchronisation can be implemented by hardware or software means. The same remark applies to data masking for which some manufacturers have designed dedicated hardware tokens or mechanisms such as bus encryption or wired fast implementations of symmetric algorithms.

The experience shows that combined countermeasures act in synergy and increase the complexity in a much larger proportion than the sum of both. For instance the simple combination of de-synchronisation tricks and data masking makes DPA (or more sophisticated variants thereof) quite harmless. In the same way, new hardware designs resist to the most state-of-the-art and best equipped experts.

Further Reading

[1] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In M. Wiener, editor, *Advances in Cryptology – CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer-Verlag, 1999.

[2] Jean-Sébastien Coron, Paul Kocher, and David Naccache. Statistics and secret leakage. In Y. Frankel, editor, *Financial Cryptography (FC 2000)*, volume 1962 of *Lecture Notes in Computer Science*, pages 157–173. Springer-Verlag, 2001.

[3] Paul N. Fahn and Peter K. Pearson. IPA: A new class of power attacks. In Ç.K. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems (CHES '99)*, volume 1717 of *Lecture Notes in Computer Science*, pages 173–186. Springer-Verlag, 1999.

[4] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic analysis: Concrete results. In Ç.K. Koç, D. Naccache, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2001*, volume 2162 of *Lecture Notes in Computer Science*, pages 251–261. Springer-Verlag, 2001.

[5] Paul Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In N. Koblitz, editor, *Advances in Cryptology – CRYPTO '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer-Verlag, 1996.

[6] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In M. Wiener, editor, *Advances in Cryptology – CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer-Verlag, 1999.

[7] Thomas S. Messerges. Using second-order power analysis to attack DPA resistant software. In Ç.K. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2000*, volume 1965 of *Lecture Notes in Computer Science*, pages 238–251. Springer-Verlag, 2000.

[8] Thomas S. Messerges, Ezzy A. Dabbish, and Robert H. Sloan. Examining smart-card security under the threat of power analysis attacks. *IEEE Transactions on Computers*, 51(5):541–552, May 2002.