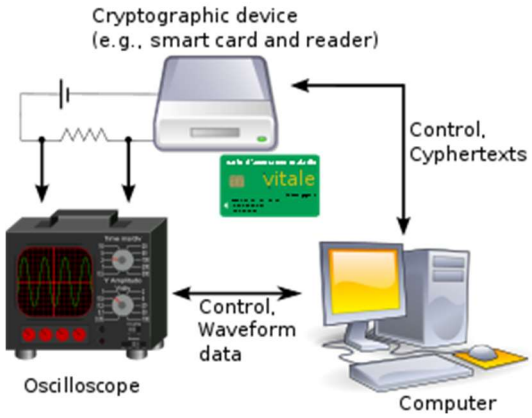
Correlation Power Analysis with a Leakage Model

Context

Side channel attacks break the secret key of a cryptosystem using channels such as sound, heat, time and power consumption which are originally not intended to leak such information.

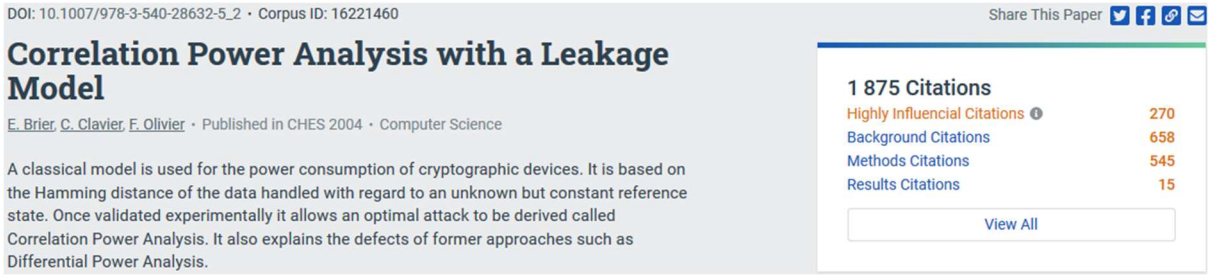
Here, we will focus on power analysis, which is a branch of side channel attacks where power consumption data is used as the side channel to attack the system. First using a device like an oscilloscope power traces are collected when the cryptographic device is doing the cryptographic operation. Then those traces are statistically analyzed using methods such as CPA or DPA to derive the secret key of the system. Being possible to break Advanced Encryption Standard (AES) in few minutes, power analysis attacks have become a serious security issue for cryptographic devices such as smart card. Nowadays, Payments, Mobile, and Defense industries use CPU with content protection against these attacks.



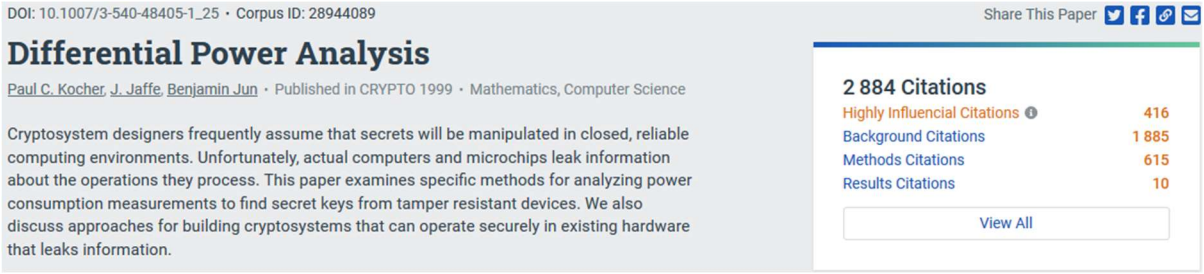
The document that I will present you was published in CHES 2004 (Cryptographic Hardware and Embedded Systems). It was written by Eric Brier, Christophe Clavier and Francis Olivier. These three men were working for Gemplus, a French company specialized in creating smart cards. It merged with Axalto to form the Gemalto group, which is from 2019 a subsidiary of Thales.

Paper's contribution

According to semantic scholar, this document is really influent in the world of side-channel attacks using power analysis. In fact, we can see that it counts 270 highly influencial citations:



In comparison, the famous article from Kocher on DPA counts about 400 highly influencial citations :



We can therefore ensure that this document is a great reference on the subject.

Technical points:

Prerequisites:

SPA = Simple Power Analysis

DPA = Differential Power Analysis

Such as DPA and SPA, Correlation Power Analysis (CPA) is an attack that allows us to find a secret encryption key that is stored on a victim device. There are 4 steps to a CPA attack:

1. Write down a model for the victim's power. This model will look at one specific point in the encryption algorithm. (ie: after step 2 of the encryption process, the intermediate result is x, so the power consumption is f(x).)
2. Get the victim to encrypt several different plaintexts. Record a trace of the victim's power consumption during each of these encryptions.
3. Attack small parts (subkeys) of the secret key:
    a. Consider every possible option for the subkey. For each guess and each trace, use the known plaintext and the guessed subkey to calculate the power consumption according to our model.
    b. Calculate the Pearson correlation coefficient between the modeled and actual power consumption. Do this for every data point in the traces.
    c. Decide which subkey guess correlates best to the measured traces.
4. Put together the best subkey guesses to obtain the full secret key.

In cryptography, an **SBox** (substitution-box) is a basic component of symmetric key algorithms which performs substitution. In block ciphers, they are typically used to obscure the relationship between the key and the ciphertext.

The Hamming Distance Consumption Model

Classically, most power analyses found in literature are based upon the Hamming weight model.

The Hamming weight of a string is the number of symbols that are different from the zero-symbol of the alphabet used. It is thus equivalent to the Hamming distance from the all-zero string of the same length. For the most typical case, a string of bits, this is the number of "1" in the string, or the digit sum of the binary representation of a given number. It is thus equivalent to the Hamming distance from the all-zero string of the same length

This model is very used because it's generally assumed that the data leakage through the power side-channel depends on the number of bits switching from one state to the other. It is also assumed that switching a bit from 0 to 1 or from 1 to 0 requires the same amount of energy. It seems relevant since the current consumed is related to the energy required to flip the bits from one state to the next.

Let's take R as reference, it's a constant machine word, not necessarily zero. The number of flipping bits to go from R to D is described by $H(D \oplus R)$. It is also called the Hamming distance between D and R.

The chip consumption depends on data part but not only (all the remaining things are noted "b"). However, the bus line are usually considered as the most consuming elements within a micro-controller.

The basic model for the data dependency can be written: $W = aH(D \oplus R) + b$

Where

- a is a scalar gain between the Hamming distance
- b is the power consumption of the chip minus the data part
- W is the power consumed

The Linear Correlation Factor

ρ_{WH} is the correlation factor between the Hamming distance and the measured power to assess the linear model fitting rate.

Property: $-1 \leq \rho_{WH} \leq +1$

for a perfect model the correlation factor tends to ± 1 if the variance of noise tends to 0, the sign depending on the sign of the linear gain a

Secret Inference Based on Correlation Power Analysis

In a m-bit microprocessor:

Let R be the true reference and $H = H(D \oplus R)$ the right prediction on the Hamming distance. Let R' represent a candidate value and H' the related model $H' = H(D \oplus R')$. Assume a value of R' that has k bits that differ from those of R, then: $H(R \oplus R') = k$. Since b is independent from other variables, the correlation test leads to:

$$\rho WH' = \rho WH \frac{m - 2k}{m}$$

This formula shows how the correlation factor is capable of rejecting wrong candidates for R. For instance, if a single bit is wrong amongst an 8-bit word, the correlation is reduced by 1/4. If all the bits are wrong (R' = ¬R), then an anti-correlation should be observed with ρWH' = −ρWH.

Estimation

In a real case with a set of N power curves Wi and N associated random data words Mi, for a given reference state R the known data words produce a set of N predicted Hamming distances $H_{i,R} = H(M_i \oplus R)$. An estimate ^ρWH of the correlation factor ρWH is given by the following formula:

$$\hat{\rho}_{WH}(R) = \frac{N\sum W_i H_{i,R} - \sum W_i \sum H_{i,R}}{\sqrt{N\sum W_i^2 - (\sum W_i)^2}\sqrt{N\sum H_{i,R}^2 - (\sum H_{i,R})^2}}$$

The summations are performed on the N samples (i = 1,N) at each time step within the power traces $W_i(t)$.

It is theoretically difficult to compute the variance of the estimator ^ρWH with respect to the number of available samples N. In practice a few hundred experiments suffice to provide a workable estimate of the correlation factor, and it's useless to go beyond.

Experimental Results

Here we will confront the leakage model to a real experience.

Hardware : Basic XOR algorithm implemented in a 8-bit chip known for leaking information.

Method :

- Load a byte D_1 into the accumulator
- XOR D_1 with a constant D_2
- Store the result from the accumulator to a destination memory cell

Results of 256 executions with D_1 varying from 0 to 255 :

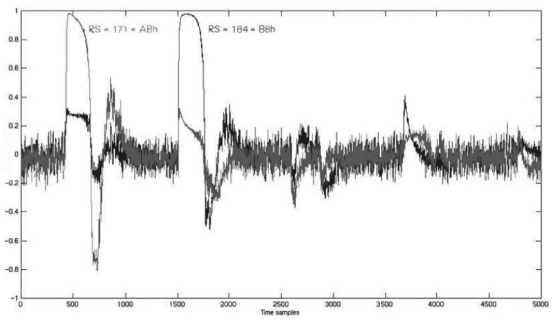


Figure 2: Consecutive correlation peaks for two different reference states

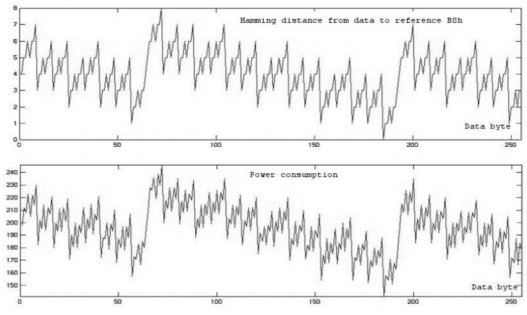


Figure 3: Model array take at the time of the second correlation peak, for varying data (0-255)

Two significant correlation peaks were obtained with two different reference states (the 1st one being the address of D_1 and the 2nd one the opcode of the XOR instruction).

These curves bring the experimental evidence of leakage principles. This behavior can also be observed on a wide variety of chips, even those implementing 16 or 32-bit architecture.

Correlation rates from 60% to more than 90% can be obtained, as we can see hereafter, for a table who provides the ranking of the 6 first guesses, obtained with only 40 curves:

$SBox_1$		$SBox_2$		$SBox_3$		$SBox_4$		$SBox_5$		$SBox_6$		$SBox_7$		$SBox_8$	
K	ρ_{max}	K	ρ_{max}	K	ρ_{max}	K	ρ_{max}	K	ρ_{max}	K	ρ_{max}	K	ρ_{max}	K	ρ_{max}
24	92%	19	90%	8	87%	8	88%	5	91%	50	92%	43	89%	2	89%
48	74%	18	77%	18	69%	44	67%	32	71%	25	71%	42	76%	28	77%
01	74%	57	70%	05	68%	49	67%	25	70%	05	70%	52	70%	61	76%
33	74%	02	70%	22	66%	02	66%	34	69%	54	70%	38	69%	41	72%
15	74%	12	68%	58	66%	29	66%	61	67%	29	69%	0	69%	37	70%
06	74%	13	67%	43	65%	37	65%	37	67%	53	67%	30	68%	15	69%

There are different countermeasures to protect data against CPA and DPA, but none can be considered as absolutely secure.

Comparison with DPA

The practical implementation of DPA (designed by Messerges) against the DES substitutions works quite well, but sometimes they produces some ghost peaks.

Ghost peaks are DPA peaks which appear for wrong guesses. The true DPA peak given by the right guess may be smaller than some ghost peaks, and even null or negative. This is quite confusing for an attacker …

The reason why wrong guesses may generate DPA peaks is that the distribution of an SBox output bit for two different guesses are deterministic and so possibly partially correlated.

The ambiguity of DPA does not lie in imperfect estimation but in wrong basic hypotheses.

All the countermeasures

Personal opinion:

I found that this paper is pretty understandable for beginners until the seventh part. I didn't succeed to understand well the differences between DPA and CPA (except for restrictions of DPA and ghosts' peaks).

However, it taught me a lot on Hamming distance, CPA, DPA, and I didn't know side channels attacks could be so accurate! I also retained that CPA is more efficient and robust than DPA, which is himself better than SPA.

I also learned that reverse engineering (process identification, bit tracing) is essential to conduct a statistical power analysis, and that protection against DPA also work to protect against CPA.