Breaking Smartcards Using Power Analysis

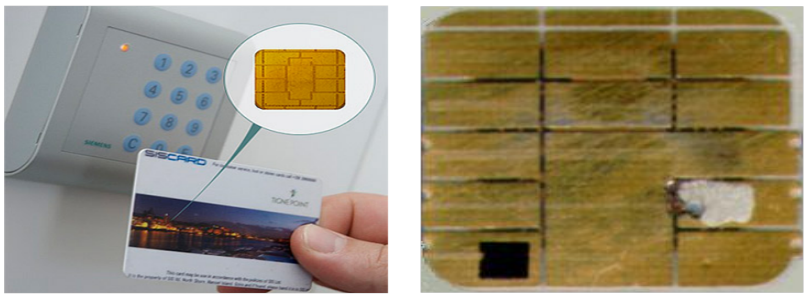
Omar Choudary (osc22)

University of Cambridge

I. Introduction

Smartcards are used Today in many applications, including cash retrieval, shop transactions, on-line banking, Pay-TV services, anti-theft protection and many more. Many of these services attract the interest of people in pirating the smartcards. For example, an attacker might make an illegal copy of a smartcard used in Pay-TV services and sell it. The owner of this illegal copy could then benefit from such services for free.

Smartcards (Figure 1) generally contain at least one microcontroller which features a CPU, memory and data buses. As a result they can be used in many cryptographic applications where hiding a secret (e.g. a private key) is fundamental.



(a) (b)

Fig. 1. *Smartcards: (a) a smartcard used in access control; (b) a tampered smartcard*

In this paper I present several methods for retrieving private information from a smartcard, focusing on the eavesdropping techniques known as Power Analysis.

To illustrate some of the techniques I have done two experiments on Differential Power Analysis (DPA) and Correlation Power Analysis (CPA). The experiments are done in Matlab and represent an approximation of a real scenario. The results demonstrate the validity of the presented attacks.

II. Overview of Smartcard Attacks

There are two main types of smartcard attacks:

- **Invasive Attacks**: these attacks imply physical tampering of the hardware. Such techniques can be used to access the chip's surface directly, and thus we can observe, manipulate, and interfere with the integrated circuit.
- **Non-Invasive Attacks**: they do not damage nor modify the physical structure of the smartcard. As described by Kömmerling and Kuhn [1], such attacks include:
  - **Software Attacks**: use the normal communication interface of the processor and exploit security vulnerabilities found in the protocols, cryptographic algorithms, or their implementation.
  - **Eavesdropping**: techniques to monitor, with high time resolution, the analog characteristics of all supply and interface connections and any other electromagnetic radiation produced by the processor during normal operation. In this paper I'll focus on these techniques.
  - **Fault Generation**: techniques that use abnormal environmental conditions to generate malfunctions in the processor that provide additional access.

Recently it has been developed a type of semi-invasive attack [2]. This attack uses an optical sensor to detect photons emitted by electronic circuits when electrons pass through them. They are called semi-invasive attacks because only the main cover is removed from the chip. There is no need to physically tamper with the device in great detail.

III. Power Analysis Attacks

In this section we'll discuss power analysis attacks, which are a type of eavesdropping attacks. This section represents the main focus of the paper.

The basic setup is the same for all the power analysis attacks. The attacker has physical access to a microcontroller and can record the external data bus as well as the current intensity (See Figure 2).

A. Simple Power Analysis

Simple Power Analysis (SPA) is a basic technique which relies only in recording the intensity of the electric current flowing through the microcontroller. As presented by Kocher et al. [4], the main idea behind SPA is that some instructions executed by the microcontroller

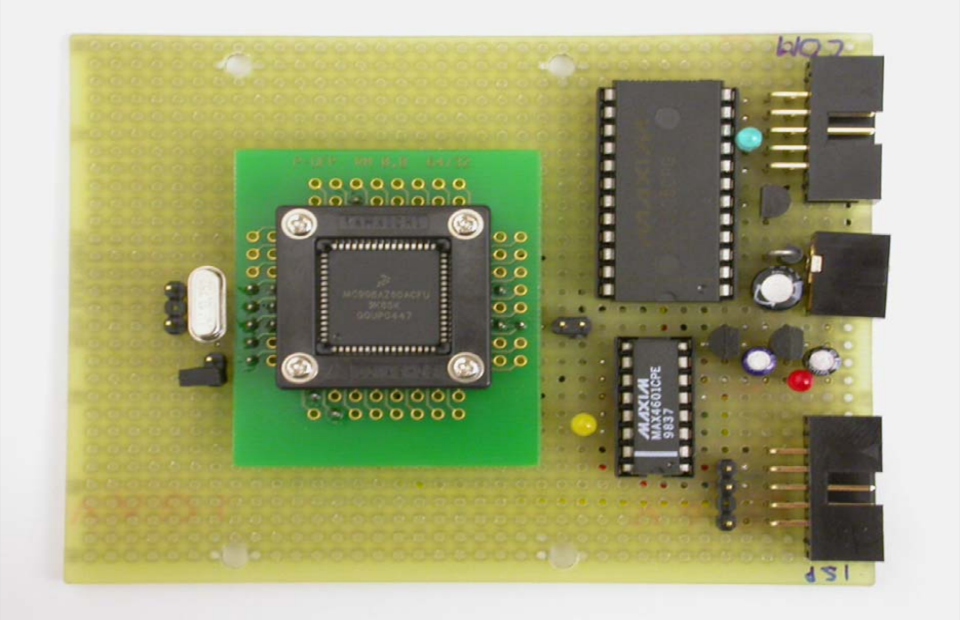


Fig. 2. *Experimental board used in power analysis [3]*

(e.g. jump, RAM access) modify the current intensity in a very noticeable way. As an example, in Figure 3 the three arrows mark register rotations in the DES encryption protocol. The power trace clearly expose these operations by simple observation.

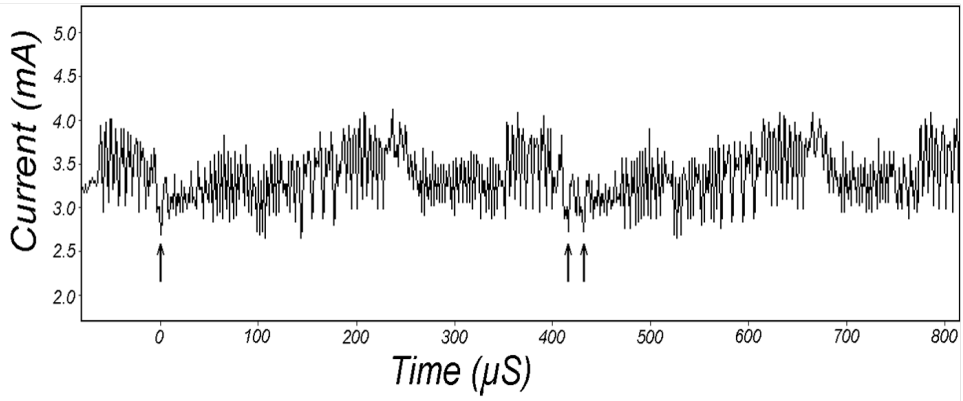


Fig. 3. *Simple Power Analysis on DES encryption: the arrows mark the moments when registers C and D are rotated; figure from [4]*

Skorobogatov and Kuhn [3] have successfully used SPA in retrieving the private password of the MC68HC908AZ60A microcontroller. This password was needed to retrieve the contents of the microcontroller's memory.

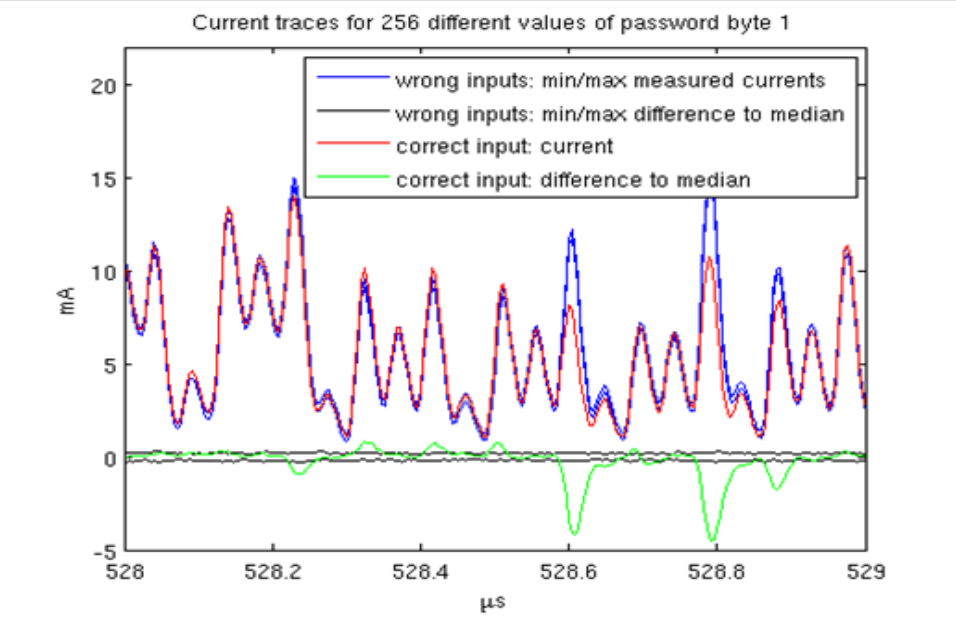


Fig. 4. *SPA attack on the MC68HC908AZ60A microcontroller: plotting the intensity for all 256 possible values of a password-byte it is possible to determine the out-lier which corresponds to the correct byte; figure from [3]*

In order to find the 8-byte long password, they sequentially tried all possible values for the 8 bytes. For a given byte all the possible 256 values were sent to the microcontroller. By analyzing the resulting intensity after each possible value they could determine precisely which was the correct value (See Figure 4).

B. Differential Power Analysis

An improved attack based on power analysis is Differential Power Analysis (DPA) [4]. This method relies on the fact that current intensity is dependent on the data processed by the microcontroller, not only on the instructions executed. However, unlike SPA, the differences in intensity caused by the variations in data cannot be observed directly.

In order to visualize the dependence between the data processed and the power trace we need to repeat the same cycle of encryption operations many times (possibly more than 1000 operations are needed). The basic process to successfully apply the DPA attack is presented in Figure 5. We input random plaintext (known) to the cipher algorithm and we record the output and the power trace (only the input plaintext or the output ciphertext are needed). We need a prior knowledge of the algorithm in order to estimate the resulted data after the first (or last) iteration of one encryption operation.

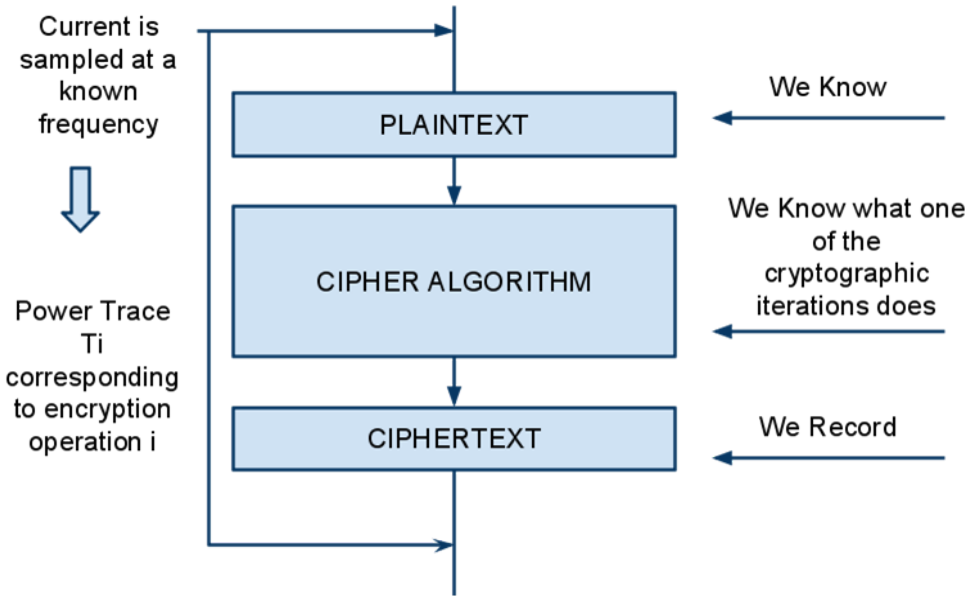


Fig. 5. *DPA process overview*

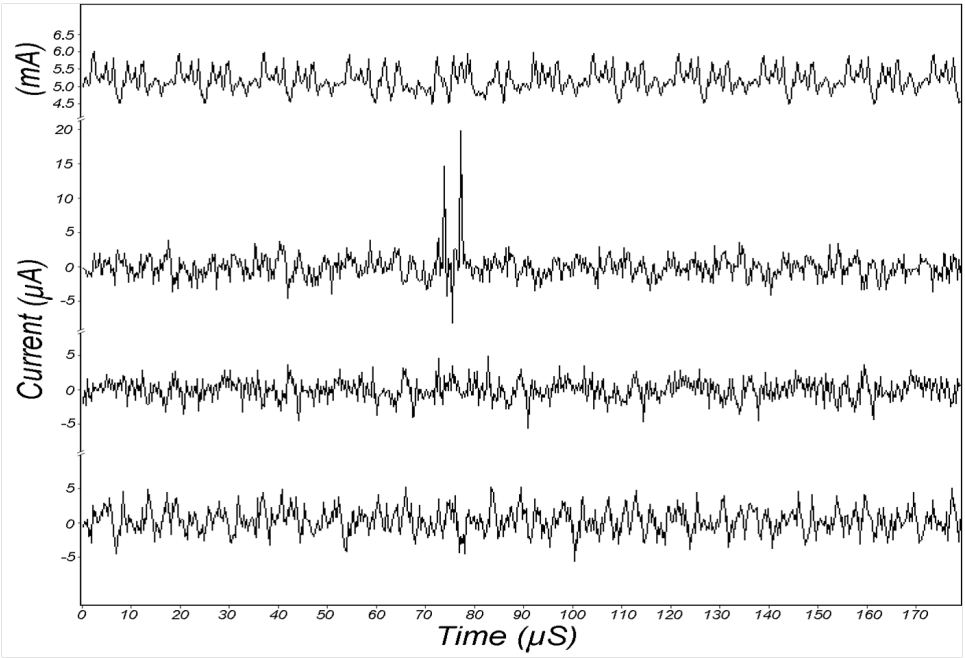


Fig. 6. *DPA attack on DES encryption standard; figure from [4]*

If we note m the number of encryption operations, $p_{1..m}$ the input plaintext for each of the operations, k the number of samples (iterations within the cryptographic algorithm) per operation, $T_{1..m}[1..k]$ the power traces for the m operations, and $D(p_i, b)$ the expectation of

bit b after the one of the iterations in operation m_i over plaintext p_i, we can compute the `differential trace`:

$$\Delta_D[j] = \frac{\sum_{i=1}^{m} D(p_i, b)T_i[j]}{\sum_{i=1}^{m} D(p_i, b)} - \frac{\sum_{i=1}^{m}(1 - D(p_i, b)T_i[j])}{\sum_{i=1}^{m}(1 - D(p_i, b))} \quad (1)$$

In Figure 6 we can see an example of DPA attack on the DES encryption algorithm. On top it is presented the reference power trace. The second row represents the differential trace for a correct estimation of a bit, while the last two rows represent the differential trace when using a wrong estimation.

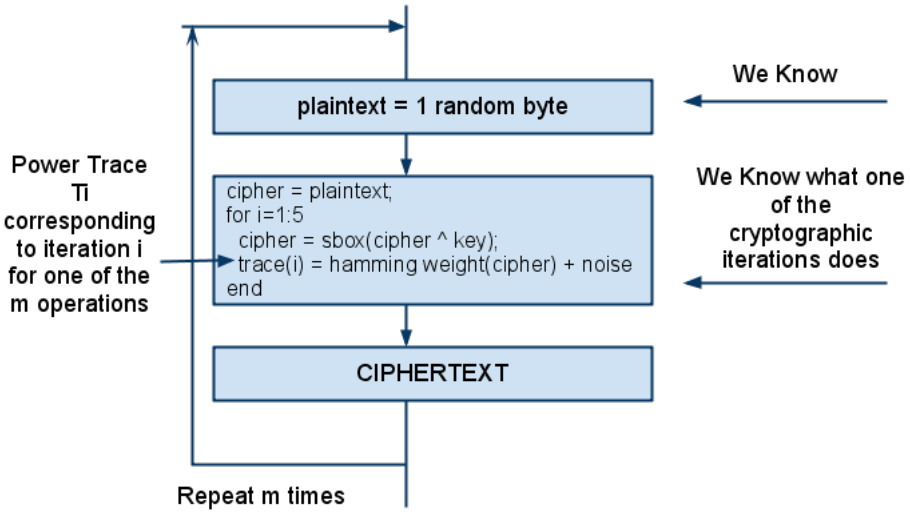


Fig. 7. *Overview of our DPA attack experiment*

I have used a simple encryption algorithm in order to test the DPA attack. This algorithms contains five iterations per encryption operation. On each iteration the algorithm encrypts an input byte (cipher) by using an XOR operation with a private key (key) and then makeing a lookup into a random permutation table (sbox).

The DPA attack on this simple algorithm can be seen in Figure 7. I simulate the power trace as the Hamming weight of the resulted byte plus some random noise.

All my experiment was done in Matlab. First I've chosen a random key. Then I've run the simple encryption algorithm for a large number of m encryption operations. For a given

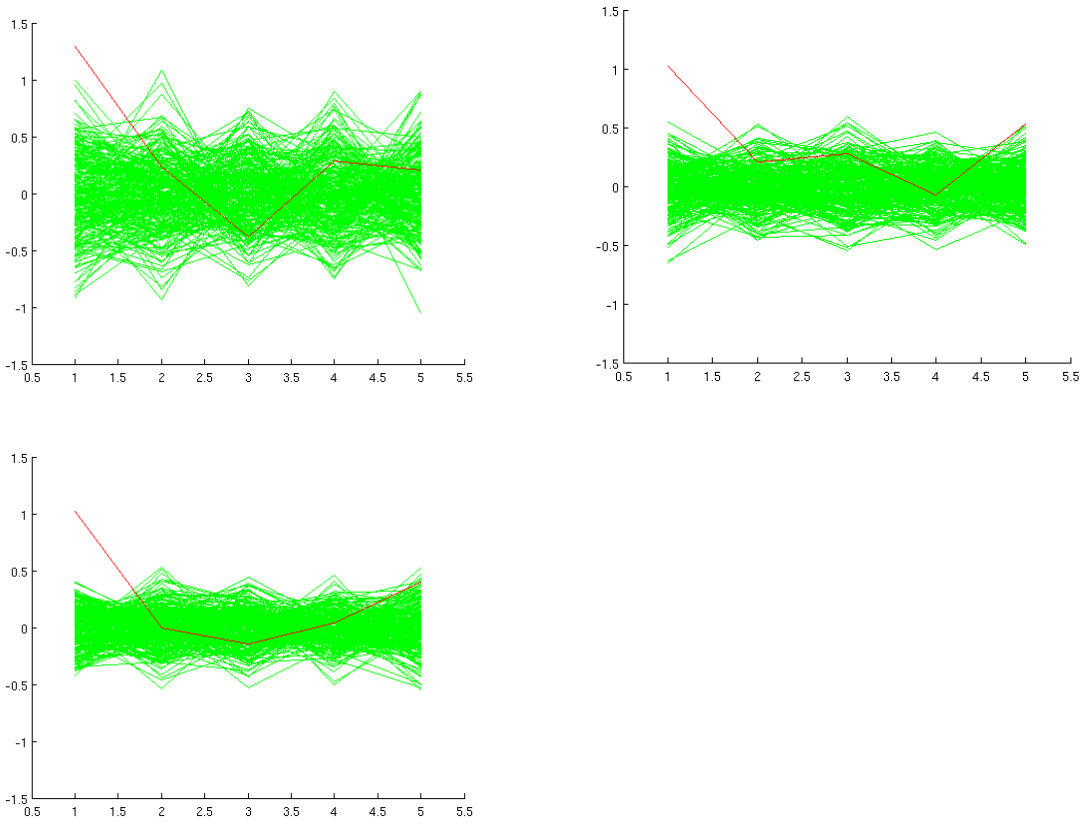


Fig. 8. *Results of our DPA attack for different values of m; on the X-axis we represent the 5 iterations of an encryption operation; the Y-axis represents the amplitude of the differential trace*

input plaintext byte p and random chosen password byte $rkey$, the estimation of bit b was simply the first iteration of our algorithm: $D(p, b) = bitand(2^b, sbox(rkey \wedge p))$.

By plotting the differential trace over a large number of encryption operations for each possible key we can easily detect the correct key. Figure 8 show these plots for different values of m. For values over 100 the correct key is clearly visible.

C. Correlation Power Analysis

Looking more carefully at the DPA attack experiment we can observe that the power trace is highly correlated to the Hamming weight of the encrypted data after the first iteration (we only added some noise). Thus we could think of an attack using this correlation. It has been actually proved that the power trace in a real scenario is correlated to the data being processed [5]. This information is used in the so called Correlation Power Analysis (CPA) attacks.

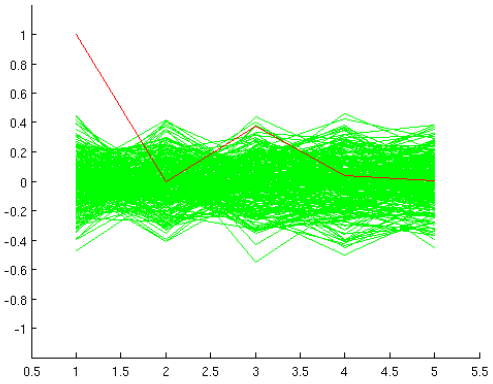


Fig. 9. *Results of the CPA attack for m=50*

As I already had the simulation in place, I made a simple implementation of the CPA attack. Instead of computing an estimate of a given bit for an encryption operation, I computed the Hamming weight of the resulted byte after the first iteration of the encryption algorithm. That is, for a given plaintext byte p, random key $rkey$ and expected data $d = sbox(rkey \wedge p)$, we compute the Hamming weight of d. The correlation of d with the power trace over all the m encryption operations can be easily found in matlab by using the function `coerrcoef`.

By plotting the correlation coefficient for each possible key we obtained the results in Figure 9, where m is 50. As it can be seen in Figure 10, the CPA attack gives better results (at least in our case) than DPA.

IV. Acknowledgments

I need to thank Dr. Markus Kuhn for his enormous help and directions in the experiments.

References

[1] O. Kömmerling and M. G. Kuhn, "Design principles for tamper-resistant smartcard processors," in *WOST'99: Proceedings of the USENIX Workshop on Smartcard Technology on USENIX Workshop on Smartcard Technology*. Berkeley, CA, USA: USENIX Association, 1999, pp. 2–2.

[2] S. Skorobogatov, "Using optical emission analysis for estimating contribution to power analysis," in *6th WORKSHOP ON FAULT DIAGNOSIS AND TOLERANCE IN CRYPTOGRAPHY - FDTC 2009*, 2009.

[3] S. Skorobogatov and M. Kuhn, "Power analysis of the Motorola MC68HC908AZ60A microcontroller," University of Cambridge, Tech. Rep., 2005.

[4] P. C. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *CRYPTO '99: Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*. London, UK: Springer-Verlag, 1999, pp. 388–397.

[5] E. Brier, C. Clavier, and F. Olivier, *Correlation Power Analysis with a Leakage Model*. Springer Berlin / Heidelberg, 2004, ch. Correlation Power Analysis with a Leakage Model.

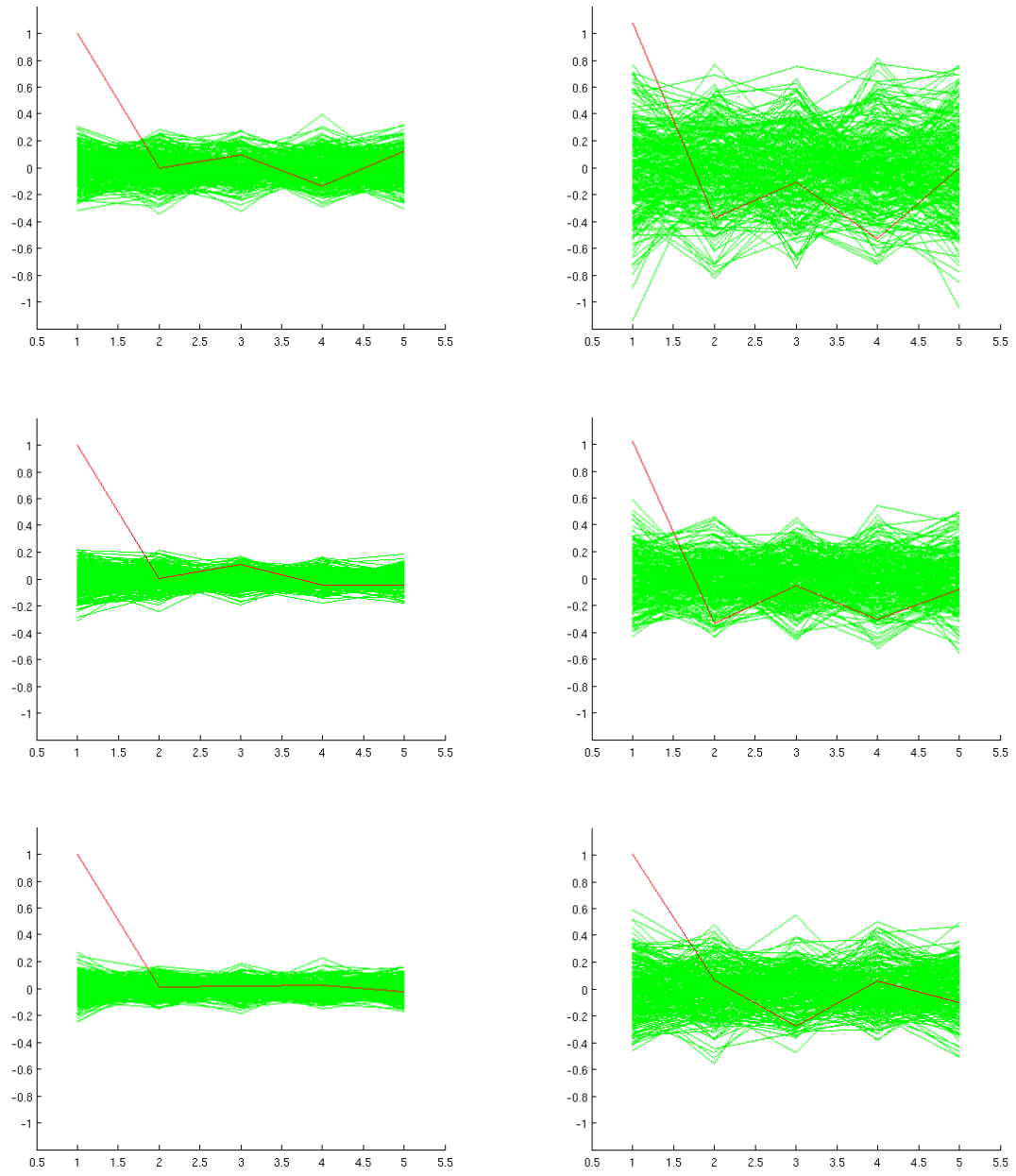


Fig. 10. *Results of CPA (a) and DPA (b), for m=100 (top), m=1000 (middle) and m=10000 (bottom). The X-axis represents the 5 iterations of each encryption operation in our algorithm; the Y-axis represents the amplitude of the correlation for CPA and the amplitude of the differential trace for DPA*

APPENDIX
MATLAB CODE

spwanoise.m:

```
function spwanoise(iterations, noise)
% spwanoise(iterations,noise)
%
% iterations is the number of encryption operations to perform
% noise is the random noise to be added to the power trace to simulate a
% real situation

numrounds = 5;

% known to attacker
sbox = uint8(randperm(2^8)-1);
plaintext = uint8(floor(rand(1, iterations) * 256));
cipherout = [];

% unknown to attacker
key = uint8(42);

% encryption and recording of power trace of toy "cipher"
trace = [];
ciphertext = plaintext;
for round=1:numrounds
  ciphertext = sbox(bitxor(ciphertext, key)+1)-1;
  noised = sum(dec2bin(ciphertext)-'0',2) + rand(iterations,1)*noise;
  trace = [trace; noised'];
  cipherout = [cipherout; ciphertext];
end

% differential power analysis
```

```
% reconstruct LSB in first round
numkeys = 256;
keys=uint8(0:numkeys-1)';
[plaintextm,keym] = meshgrid(plaintext, keys);

lsb = bitand(1,sbox(bitxor(plaintextm, keym)+1)-1);

tracepos = [];
traceneg = [];
for i=1:numkeys
    pos = trace(:, find(lsb(i,:)));
    posavg = mean(pos, 2);
    tracepos = [tracepos; posavg'];

    neg = trace(:, find(lsb(i,:) == 0));
    negavg = mean(neg, 2);
    traceneg = [traceneg; negavg'];

end

diff = tracepos - traceneg;

% search for key that leads to peak

ax = uint8(1:numrounds);
figure;
hold on;
axis([0.5 (numrounds+0.5) -1.2 1.2]);
plot(ax,diff,'g');
plot(ax,diff(43,:),'r'); % we know key at index 43 is the correct
```

```
%We try to do some correlation analysis

%compute hamming weight for all keys/iterations
rh = hammingweight(sbox(bitxor(plaintextm, keym)+1)-1);

correlation =[];
for i=1:numkeys
    cr = [];
    for round=1:numrounds
        cf = corrcoef(rh(i,:),trace(round,:));
        cr = [cr cf(1,2)];
    end
    correlation = [correlation; cr];
end

%show
ax = uint8(1:numrounds);
figure;
hold on;
axis([0.5 (numrounds+0.5) -1.2 1.2]);
plot(ax,correlation,'g');
plot(ax,correlation(43,:),'r');
```

hammingweight.m:

```
function H = hammingweight(X)
% h = hammingweight(X)
% where X is a matrix
%
% This function computes the Hamming weight of all values in X, that is t
% sum of "1" bits in each element

[m,n] = size(X);
```

```
H = [];
for i=1:m
    H = [H; sum(dec2bin(X(i,:))-'0',2)'];
end
```