Power Analysis Attacks against FPGA Implementations of the DES

Abstract. Since their publication in 1998 [1], power analysis attacks have attracted significant attention within the cryptographic community. So far, they have been successfully applied to different kinds of (unprotected) implementations of symmetric and public-key encryption schemes. Most published attacks apply to smart cards and only a few articles asses the vulnerability of FPGA implementations to power analysis attacks [2, 3]. In this paper, we demonstrate the specificity of this kind of platform in the context of Differential Power Analysis (DPA). First, we show that the original attack described in [1] and its most recent developments [4] are hardly applied to FPGAs because the physical behavior of such devices is different than smart cards. Then we generalize the power consumption model and apply it to FPGAs. Finally, we describe a correlation attack were theoretical predictions of the power consumption are correlated with real measurements in order to make an efficient use of all the collected data. All these techniques are successfully applied to an FPGA implementation of the DES, being the first effective side-channel attack against FPGA implementations of block ciphers.

Keywords: Secure hardware and smartcards, FPGAs, Differential Power Analysis, correlation attacks, Data Encryption Standard.

1 Introduction

The design and implementation of field-programmable gate array (FPGA) technology is rarely described in the literature because much of the information is proprietary. In general, FPGAs may be viewed as a "sea" of programmable logic gates where the logic, but also the routing are user-programmable. As a consequence, the implementation of FPGA designs can be performed at the user site. Synthesis and implementation tools allow to translate the high level description of a design into the programming file for an FPGA. For these reasons, FPGAs are becoming increasingly popular, especially for rapid prototyping. Many recent publications illustrate their growing performance and flexibility for any digital signal processing applications (e.g. video processing, cryptography).

In this paper, we first try to characterize the power consumption of certain commercially available FPGAs. We focus our attention to the static RAM-based (volatile) programming technology that is used in several recent devices (e.g. Xilinx Virtex, [5]). In these FPGAs, programmable logic and connections are controlled by SRAM cells. We present the general architecture of logic blocks, connection blocks and programming circuits. Although this does not exactly correspond to commercial devices, it is probably a good illustration of how an FPGA behaves at the transistor level. Then, we investigate the power consumption of these circuits.

Appeared in *Proceedings of Field-Programmable Logic and its Applications*, Lecture Notes in Computer Science 3203, J. Becker, M. Platzner, and S. Vernalde (eds.), Springer-Verlag, pp. 84–94, 2004. ©2004 Springer-Verlag

For implementation of cryptographic algorithms, not only the performances of a circuit are important, but also its security against implementation attacks. The second part of this paper investigates the specificity of FPGAs in the context of power analysis attacks. In Differential Power Analysis, an attacker uses an hypothetical model of the attacked device to predict its side-channel output. These predictions are then compared to the real measured side-channel outputs in order to recover secret information (e.g. key bits). The quality of the model highly influences the effectiveness of the attack. Based on reasonable assumptions on their power consumption, we demonstrate that the original attack described in [1] and its most recent developments [4] are hardly applied to FPGAs because the physical behavior of such devices is different than smart cards. Then we generalize the power consumption model and apply it to FPGAs. We also describe a correlation attack where theoretical predictions of the power consumption are correlated with real measurements in order to make an efficient use of all the collected data [6]. The resulting attack is more efficient than the popular "multiple-bit" DPA and allows interesting theoretical predictions of the attacks with simulated data. Finally, we successfully apply these techniques to an FPGA implementation of the DES, being the first effective side-channel attack against FPGA implementations of block ciphers.

This paper is stuctured as follows. Section 2 briefly remembers the basics of CMOS circuits power consumption. Section 3 describes how FPGAs may be implemented in CMOS technology and section 4 describes the power consumption of the resulting circuits. Section 5 presents some preliminary tests and section 6 briefly describes the DES and its FPGA implementation. Section 7 defines the original DPA and we illustrate why its model does not fit to FPGAs. Section 8 presents a practical attack where we modified the selection function in order to take the physical behavior of FPGAs into account. Section 9 describes the correlation attack against FPGA implementation of the DES. Finally, conclusions are in section 10.

2 Static CMOS power consumption

The most popular technology used to build programmable logic is static RAM[1]. In SRAM-based FPGAs, the storage cells, the logic blocks and the connection blocks are made of CMOS gates. In the next section, we briefly introduce static CMOS circuits and their power consumption [14–16].

2.1 Static CMOS gates

A static CMOS gate is a combination of two networks, called the $pull-up$ $network$ (PUN) and the $pull-down$ $network$ (PDN). The PUN consists solely

[1] SRAM: Static Random Access Memory, see appendixes, Figure 9.

of PMOS transistors and provides a conditional connection to V_{DD}. The PDN potentially connects the output to V_{SS} and contains only CMOS devices. The PUN and PDN networks should be designed so that, whatever the value of the inputs, one and only one of the networks is conducting in the steady state. As a consequence, theoretically, static CMOS gates have no static power consumption.

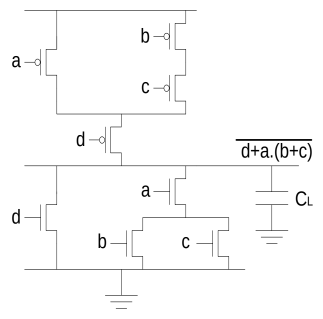


Fig. 1. Static CMOS gate.

2.2 Power consumption of static CMOS circuits

In practice, there are three components that establish the amount of power dissipated in static CMOS circuits. These are:

1. Static dissipation.
2. Short circuit power dissipation.
3. Dynamic dissipation.

Static power consumption: The static dissipation is due to small current leakages in diodes and transistors. It is therefore the product of the device leakage current and the supply voltage. A useful estimate is to allow a leakage current of 0.1nA to 0.5nA per gate at room temperature. Then total static power dissipation P_S is obtained from

$$P_S = \sum_{1}^{n} leakage\ \ current * supply\ \ voltage \tag{1}$$

Where n is the total number of devices. It is worth mentioning that the junction leakage currents are caused by thermally generated carriers. Therefore their value increases with increasing junction temperature, and this occurs in an exponential fashion.

Short circuit power dissipation: Short circuit power consumption refers to the scenario where power is dissipated with the momentary existence of a direct path from the power supply to ground. The scenario occurs when both NMOS and PMOS devices are conducting. For carefully designed circuits, short circuit power dissipation is negligible.

Dynamic power consumption Dynamic power consumption is due to the charge and discharge of the load capacitance C_L (see Figure 1) and is the most important part in the power consumption of a CMOS device. It can be evaluated by observing that during the low-to-high transition, C_L is loaded with a charge $C_L V_{DD}$. This charge requires an energy from the supply equal to $C_L V_{DD}^2$ ($=Q \times V_{DD}$). The energy stored on the capacitor equals $C_L V_{DD}^2/2$. This means that only half of the energy supplied by the power source is stored on C_L. The other half has been dissipated by the PMOS transistor. Notice that the percentage of energy dissipation is independent of the size (and hence the resistance) of the PMOS device. During the discharge phase, the charge is removed from the capacitor, and its energy is dissipated in the NMOS device. In summary, each switching cycle (L$\rightarrow$H and H$\rightarrow$L transition) takes a fixed amount of energy, equal to $C_L V_{DD}^2$. If the gate is switched on and off f times per second, the power consumption equals:

$$P_D = C_L V_{DD}^2 f \tag{2}$$

As a consequence, the power dissipation is data dependent and is a function of the switching activity of a circuit. This point is essential for security as it is the basic assumption made by most attackers.

2.3 Switching activity of logic gates

Power in static CMOS circuits is mainly consumed during the switching of the gates and can be expressed as:

$$P_D = C_L V_{DD}^2 f_{0\rightarrow 1} \tag{3}$$

with $f_{0\rightarrow 1}$ the frequency of energy-consuming transitions (or 0$\rightarrow$ 1 transitions for static CMOS). Computing the dissipation of a complex gate is complicated by the $f_{0\rightarrow 1}$ factor, also called the *switching activity*. One concern is that the switching activity of a network is a function of the nature and the statistics of the input signals. If the input signals remain unchanged, no switching happens and the dynamic power consumption is zero. On the other hand, rapidly changing signals provoke plenty of switching and hence dissipation. These factors can be incorporated by introducing a slight modification in equation 3:

$$P_D = C_L V_{DD}^2 P_{0\rightarrow 1} f \tag{4}$$

with f the average event rate of inputs and $P_{0\rightarrow 1}$ the probability that an input transition results in a 0$\rightarrow$1 (or consuming) event.

Note that the dependence of power consumption on circuit activity highly depends on the technology considered. The power consumption in static CMOS circuits is proportional to switching activity. In the case of dynamic circuits, power is only dissipated when the output is set to zero which results in a different power consumption model.

3 Field Programmable Gate Arrays

3.1 General description

In general, FPGAs [17–26] may be viewed as a "sea" of programmable logic gates where the logic, but also the routing are user-programmable, as illustrated in Figure 2. In RAM-based FPGAs, SRAM cells are used to store the configuration bits that program the different components of the device (e.g. logic blocks, connection blocks, ...). A basic SRAM cell is described in the appendixes (Figure 9). Configurable logic blocks are usually made of 4-input lookup tables[2] and storage

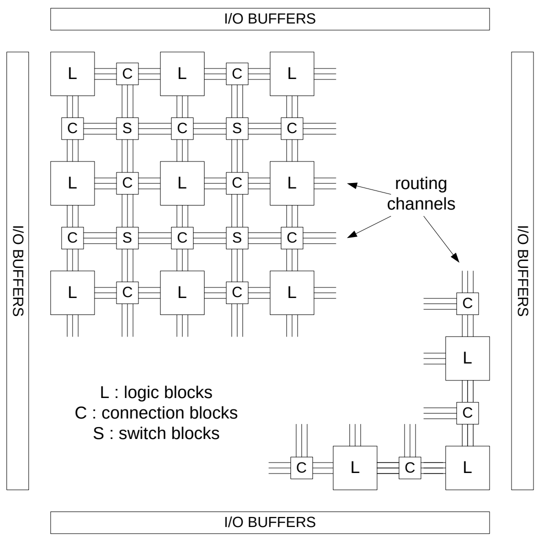


Fig. 2. FPGA: high level view.

elements (D flip flops). Additional logic elements (e.g. XOR gates, multiplexors) are often provided in the logic blocks of recent devices and allow efficient implementations of arithmetic functions, RAM blocks, ... Next to the logic blocks, FPGAs also contain programmable routing (connection blocks, switch blocks) and I/O buffers. Programable connections of FPGAs are used to connect the logic blocks to routing channels and to provide connectivity between the different routing channels. Two solutions are usually considered to implement these connections. A first solution is to use switches, i.e. pass transistors controlled by a RAM cell. Another possibility is to use multiplexors. The principal advantage of multiplexors is that a single bit of control manages several switches and therefore the use of RAMs is more efficient. The major drawback is the critical

[2] LUTs are programmable components that allow to implement any function of 4 input bits and 1 output bit. They are usually implemented as 16 x 1 multiplexors.

path of the signal through multiplexors. Finally, Embedded memories or multipliers are present in certain recent FPGAs but not represented in Figure 2. As an illustration, a high-level view of the logic block of Xilinx Virtex FPGAs is given in the appendixes (Figure 10) with the general architectures for all the usual components of FPGAs : lookup tables (Figure 11), flip flops (Figure 12), buffers (Figure 13) and connection blocks (Figures 14 and 15).

3.2 Programming technology

Different techniques for programming FPGAs (i.e. to load the configuration bits into SRAM cells) were identified in the open literature. The simplest one is to connect all the bits into a long shift register as illustrated in the appendixes (Figure 16). This requires only one pin to input the programming data but is the slowest method. Faster configuration speeds are obtained by organizing programming bits into memories and to manage them as parallel data.

4 Power consumption of FPGAs

Based on previous descriptions, we may clearly asses that the power consumption of FPGAs will not significantly differ from the one of static CMOS integrated circuits. However, a notifiable feature of FPGAs is that they are made of different resources (e.g. logic blocks, connections) of which the power consumption differ because of different effective load capacitances. As a consequence, the power consumption of FPGA designs will not only depend on their switching activity but also on the internal resources used.

Recent works [27–29] tried to identify these important resources in the FPGA architecture and to characterize their power consumption. Using a large set of real designs, it is possible to evaluate the effective capacitance of each resource. Then the power consumption of a design is estimated by using switching activity and capacitance values. As a motivating example, we refer to the results presented in [29]. According to this work, the power dissipation share of routing, logic and clocking resources in Xilinx Virtex-II FPGA family are 60%, 16% and 14% respectively[3]. The effective capacitance of all resources are given in Table 1. In this table, the segmented routing architecture include wires that travels two logic blocks (called Doubles), six logic blocks (called Hexes) and the length of the chip (called Longs), in both vertical and horizontal dimensions. There are also two sets of switches to connect the wire segments to the inputs and outputs of each logic blocks. We refer to these sets as Input Crossbar (IXBar) and Output Crossbar (OXBar). These results show that, other than global wiring for clocking, the resources with the largest effective capacitance are inputs to LUTs. The components with the second highest effective capacitance are interconnects.

[3] These are general guidelines as these results are obviously dependent on the application considered.

Type	Resource	Capacitance (pF)
Interconnect	IXbar	9.44
Interconnect	OXbar	5.12
Interconnect	Double	13.20
Interconnect	Hex	18.40
Interconnect	Long	26.10
Logic	LUT inputs	26.40
Logic	FF inputs	2.88
Logic	Carry	2.68
Clocking	Global wiring	300
Clocking	Local wiring	0.72

Table 1. Effective capacitance summary.

Since LUTs are driven by either input ports or output of CLBs, these results underline that logic block outputs drive much larger capacitive loads than any of their internal signals (e.g. FF inputs).

Finally, more accurate estimations about the most consuming components of an FPGA design can be derived from the delay information that is generated by most implementation tools [30]. As an input delay represents the delay seen by a signal driving that input due to the capacitance along the wire, large (resp. small) delay values indicate that the wire has a large (resp. small) capacitance. Based on the reports automatically generated by implementation tools, one may expect to recover a very accurate information about the signals that are driving high capacitances. The knowledge of the implementation netlists with delay information is therefore susceptible to provide an attacker with advantages that could be used to improve the efficiency of an attack.

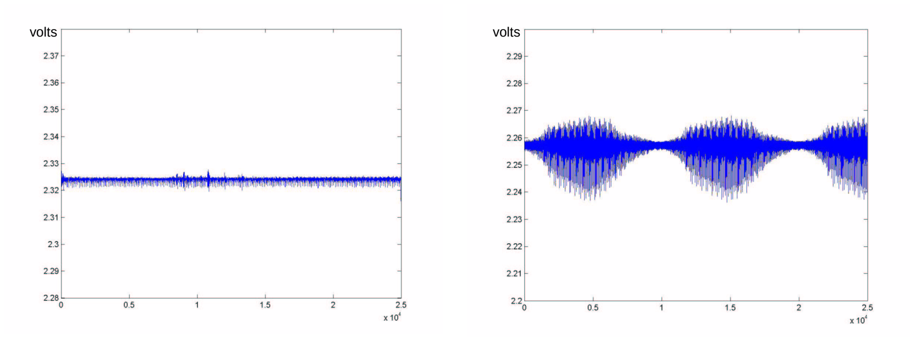
5 Preliminary test

A simple test design may confirm that the power consumption is dependent on the switching activity. Three vectors are defined and implemented in the FPGA[4]:

1. a is a bit-vector with a constant Hamming weight $H = 1$. The position of the 1-bit inside the vector is incremented/decremented from 0 to 100.
2. b is a bit-vector for which the Hamming is incremented/decremented from 0 to 100.
3. c is a bit-vector for which the number of bit switches between two consecutive states is incremented/decremented from 0 to 100.

A design that generates these 3 bit-vectors is given in the appendixes (Figure 17). Figure 3(a) illustrates the power consumption of vectors a and b. Figure 3(b) illustrates the power consumption of vectors a, b and c. From this experiment, we observe that:

[4] Xilinx Virtex XCV1000-4-BG560.



(a) Hamming (b) Switch

Fig. 3. Preliminary test.

1. The most consuming vector is c.
2. We can distinguish the number of bit switches by observing the power traces (e.g. peaks in Figure 3(b) correspond to 100 bit switches between two states).

6 The Data Encryption Standard

In 1977, the Data Encryption Standard (DES) algorithm [7] was adopted as a Federal Information Processing Standard for unclassified government communication. Although a new Advanced Encryption Standard (AES, [8]) was selected in October 2000, DES is still largely in use. DES encrypts 64-bit blocks with a 56-bit key and processes data with permutations, substitutions and XOR operations. It is an iterative block cipher that applies a number of key-dependent transformations called rounds to the plaintext. This structure allows very efficient hardware implementations.

Basically, the plaintext is first permuted by a fixed permutation IP. The result is next split into the 32 left bits and the 32 right bits, respectively L and R that are sent to 16 applications of a round function. The ciphertext is calculated by applying the inverse of the initial permutation IP to the result of the 16-th round.

The secret key is expanded by the key schedule algorithm to 16 x 48-bit subkeys K_i and in each round, a 48-bit subkey is XORed to the text. The key schedule consists of known bit permutations and shift operations. As a consequence, finding any subkey bit directly involves that the secret key is corrupted.

Finally, the round function is represented in the grey part of Figure 4(a) and is easily described by:

$$L_{i+1} = R_i \tag{5}$$

$$R_{i+1} = L_i \oplus f(R_i, K_i) \tag{6}$$

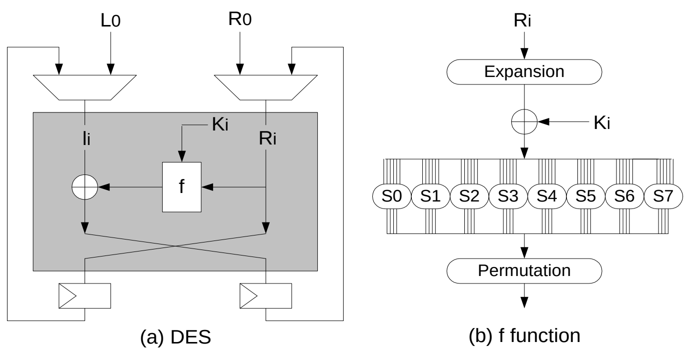


Fig. 4. Data Encryption Standard.

where f is a nonlinear function detailed in Figure 4(b): the R_i part is first expanded to 48 bits with the E box, by doubling some R_i bits. Then, it performs a bitwise modulo 2 sum of the expanded R_i part and the 48-bit subkey K_i. The output of the *XOR* function is sent to eight non-linear S-boxes (S). Each of them has six input bits and four output bits. Their result are permuted by the box P.

The design we used to carry out the experiments is a sequential DES [9] that takes one clock cycle to perform one round. It is represented in Figure 4(a).

7 Original attack

In its original form [1], the Differential Power Analysis of DES requires a selection function $D(C, b, K_{Sb,15})$ that we define as computing the value of a bit b which is a part of intermediate vector L_{15} (Figure 5(a)). As b results of a partial decryption through the last round of the algorithm, it can be derived from the ciphertext C and the 6 key bits entering in the same s-box as bit b.

To implement the DPA attack, an attacker first observes m encryptions and captures m power traces T_i and their associated ciphertexts C_i. No knowledge of the plaintext is required. With a guess of 6 key-bits $K_{Sb,15}$, the function D can be computed for each i and we can obtain two sets of traces: one corresponding with $D_i = 0$ and the other with $D_i = 1$. Each set is then averaged to obtain two average traces A_0 and A_1 and we can compute the difference $\Delta = A_0 - A_1$.

If $K_{Sb,15}$ is correct, the computed value for D will equal the actual value of target bit b with probability 1. As the power consumption is correlated to the data, the plot of Δ will be flat, with spikes in regions where D is correlated to the values being processes. If $K_{Sb,15}$ is incorrect, Δ will be flat everywhere. Finally, a multiple bit attack is denoted when the selection function outputs d bits with $d > 1$. It allows to improve the SNR of the attack. Basically, if a single-bit DPA attack using N traces has a signal to noise ratio SNR_1, then an *all-zeros-or-all-ones* d-bit DPA attack using N traces will have a ratio $SNR_d = d \times SNR_1$.

The single and multiple-bit DPA are described by algorithm 1.

According to [10], the selection function was chosen because, at some point during a software DES implementation, the software needs to compute its value. When this occurs or any time data containing the selection bits is manipulated, there will be a slight difference in the amount of power dissipated, depending on wether these bits are 0s or 1s. However, in the case of RAM-based FPGA implementations, this function does not correctly fit to the physical behavior of the devices. In a multiple-bit attack, one tries to distinguish bit vectors of different Hamming weights, although it is clear from Figure 3 that the most significant consumption differences are related to the switching activity between two states. In the next section, we propose to modify the selection function in order to take the physical behavior of FPGAs into account.

Algorithm 1 DPA Attack on DES

```
1. Guess the bits of $K_{16}$ involved in the selection function $D$.
2. Initialize $A_0 = A_1 = 0$.
3. Get a ciphertext $C_i$ and its corresponding power trace $T_i$.
4. Reverse-calculate the selection function $D$.
5. If $D = 0$ (or $D =$ "00..0" in case of multiple bit attacks) then
        6. Add power trace to $A_0$.
7. Else if $D = 1$ (or $D =$ "11..1" in case of multiple bit attacks)
        8. Add power trace to $A_1$.
9. Else do nothing.
10. If not enough averages, goto 3.
11. DPA bias signal: $\Delta = A_0 - A_1$.
```

8 Modified attack: an improved selection function

In its original form, the selection function is defined as computing the value of a bit b which is part of the intermediate vector L_{15}. In case of multiple bit attacks, d bits are computed and we denote them by $D = L_{15}[p_0, p_1, ..., p_{d-1}]$, where p_i is the position of the *ith* bit guessed. Distinguishing $D =$ "00...0" from $D =$ "11...1" therefore means to distinguish vectors of different Hamming weights. A modified selection function can be defined as followed. Let D_1 be the original selection function that involves bits $L_{15}[p_0, p_1, ..., p_{d-1}]$. As L_{16} is part of the ciphertext, we can access it and we denote a function $D_2 = L_{16}[p_0, p_1, ..., p_{d-1}]$. Then we can define different selection functions correlated with the switching activity of the device:

1. $D' = D_1 \oplus D_2$ takes all transitions into account.
2. $D'' = \overline{D_1}.D_2$ only considers transitions $0 \rightarrow 1$.
3. $D''' = D_1.\overline{D_2}$ only considers transitions $1 \rightarrow 0$.

Based on these selection function, we mounted successful 4-bit attacks against FPGA implementations of the DES. However, as a multiple bit attack only considers the texts that give rise to 0 or d switches, it is far from optimal and a

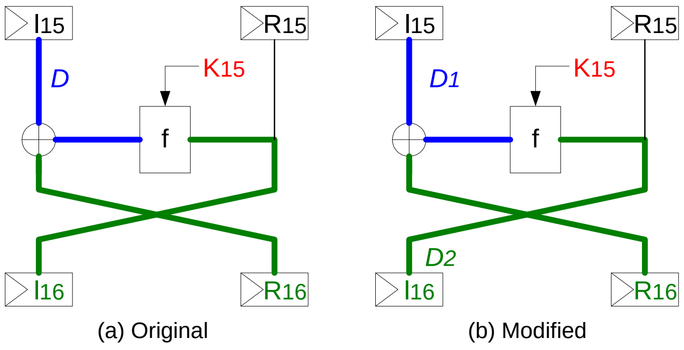


Fig. 5. Selection functions D.

lot of texts are actually not used. In the next section, we propose an improved attack based on the correlation between theoretical power consumption files and practical traces.

Remark that in the case of software implementations in smart cards, the same model is used implicitly. Reference [4] clearly explains that the DPA model is actually based on the Hamming distance of the data handled with regard to an unknown but constant reference state. This constant reference state simply corresponds to the address of an instruction. As a software implementation will load the instruction before loading the data, a DPA attack actually models the switching activity between two states, but one of these states (i.e. the instruction address) is constant. Our selection function (with two variable states) is therefore a generalization of the original DPA model.

9 Modified attack: prediction files and correlation attack

9.1 Background

In DPA, an attacker uses an hypothetical model of the attacked device to predict its side-channel output. The quality of this model is dependent on the knowledge of the attacker and highly influences the efficiency of a practical attack. These predictions are then compared to the real, measured side-channel outputs of the device (e.g. in d-bit DPA, the attacker assumes that the power consumption of the device with d bits switching will be higher than when none of these bit are switching).

The comparisons between model predictions and real measurements may be performed by applying statistical methods, e.g. the *distance-of-mean test* and the *correlation analysis*. For the correlation analysis, the model predicts the amount of side-channel leakage for a certain moment of time in the execution of the algorithm and a certain key hypothesis [6]. Then we correlate these predictions to the real side-channel output. To measure this correlation, the Pearson correlation coefficient can be used. Let T_i denote the *ith* measurement data (i.e. the *ith* trace) and T the set of traces. Let P_i denote the prediction of the model for the *ith* trace and P the set of such predictions. Then we calculate:

$$C(T,P) = \frac{E(T.P) - E(T).E(P)}{\sqrt{Var(T).Var(P)}}. \tag{7}$$

Where $E(T)$ denotes the mean of the set of traces T and $Var(T)$ its variance. If this correlation is high, it is usually assumed that the prediction of the model, and thus the key hypothesis, is correct.

9.2 A correlation attack using simulated data

One of the important issues in making a practical power analysis attack is how to obtain relatively noise free measurements. The more noisy the obtained measurements are, the worse the statistical evaluation works and the more measurements are needed. Therefore, an interesting first step in evaluating a device against side-channel attacks is to simulate its behavior with theoretical noise-free data.

Our target for the correlation attack is the 4 bits of the register L that are effected by the 6 MSBs of the round key 16. It corresponds to the output bits of S-box $S0$ (Figure 4(b)). A theoretical prediction of the attack can be performed by running it with known simulated data.

In the first step of this simulated attack, we produce a global simulated power consumption file. For this purpose, we choose N random plaintexts and one fixed, but random key. After each encryption round (clock cycle), the program writes the number of bits that switches between the previous and the current values of registers L and R to this file. Hence, the program produces a file which contains a $N \times 16$ matrix M_1 ($N = 10\,000$), with values between 0 and 64. We denote M_1 as the global prediction matrix. Remark that since the same key is used for all the measurements, the power consumption of the key schedule is fixed and may be considered as a DC component that we can neglect.

In the second step, we produce a simulated power consumption file for the 4 bits that are targeted by our attack. Hence, we calculate an $N \times 2^{kg}$ matrix[5] M_2, which is made of numbers between 0 and 4. Each column of the matrix M_2 contains the power consumption predictions for the targeted bit-changes in a certain round r and a particular guess of the kg targeted key bits for N random plaintexts. In a practical attack, only the first and last round are accessible. We denote M_2 as the selected prediction matrix.

Then we compute the correlation coefficient between one column (corresponding to the round targeted by the correlation analysis) of the global prediction matrix and all the columns (corresponding to all the round r key guesses) of the selected

[5] kg is the number of bits targeted by the correlation analysis. In our example, $kg = 6$.

prediction matrix as follows.

$$c_i = C(M_1(1:N, r), M_2(1:N, i)), \qquad 0 \leq i \leq 2^{kg} - 1. \tag{8}$$

If the attack is successful, we expect that only one value, corresponding to the correct key guess, leads to a high correlation coefficient. Figure 6 shows that this

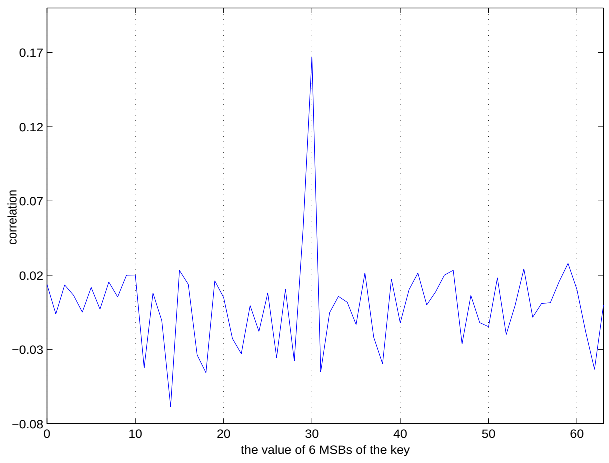


Fig. 6. A correlation attack using simulated data

expectation is fulfilled and the correct 6 MSBs of the last round key guess are $1A_{hex} = 30_{dec}$.

As the useful information for an attacker is to know the minimum number of plaintexts that are necessary to determine the correct key, we also calculated this correlation coefficient for different values of N as follows.

$$c_{i,j} = C(M_1(1:j, r), M_2(1:j, i)), \qquad 0 \leq i \leq 2^{kg} - 1, 1 \leq j \leq 2\,000. \tag{9}$$

As shown in Figure 7, after approximately 400 plaintexts the right kg key-bits can be distinguished from a wrong guess. We may therefore say that the attack is **theoretically successful** after about 400 texts.

9.3 A correlation attack using the measured data

When attacking a device practically, the selected prediction file stays unchanged while we replace the global prediction file by the real measurements. We let the FPGA[6] encrypt the same $N = 10\,000$ plaintexts with the same key as we did

[6] The measurement setup consists of a Xilinx Virtex XCV800-4-HQ240 FPGA, an FPGA board and a TDS Tektronix 714L sampling oscilloscope with a Tektronix CT-1 current probe to measure the supply current. The FPGA uses two separate power supplies, a 3.3V supply for I/O and a 2.5V supply for the core cells. Only the core power supply has been measured.

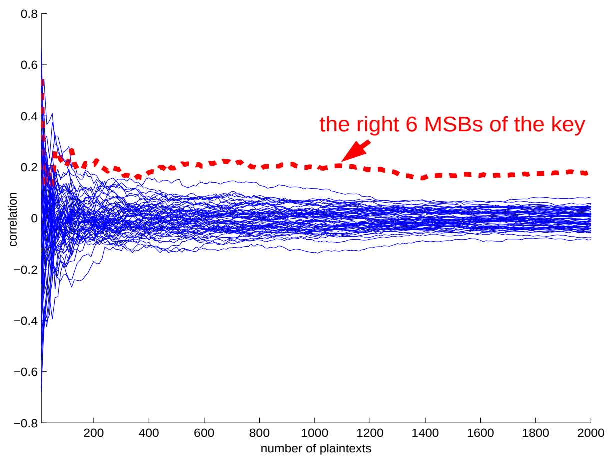


Fig. 7. A correlation attack using simulated data for different N values.

in the first step of section 9.2. While the chip was operating, we measured the power consumption for 16 consecutive clock cycles, corresponding to 16 rounds of the DES. With these measurements, we produced a $N \times (16.D)$ matrix, M_3, where D is the number of data points measured during one clock cycle.

In order to identify the correct kg MSBs of the round key, we used the correlation coefficient again. Additionally, we applied a pre-processing technique to reduce the noise in the acquired measurements. The pre-processing essentially consists of averaging. We have calculated the mean values of the measurement data in the different clock cycles (i.e. rounds) as follows:

$$e_{i,j} = E(M_3(i, D \times (j-1) + 1 : D \times j)), \qquad 1 \leq i \leq N, 1 \leq j \leq 16. \tag{10}$$

$M_3(i, D \times (j-1) + 1 : D \times j)$ is the vector which is made of the ith row and the columns between $D \times (j-1) + 1$ and $D \times j$ of M_3. Then we defined a matrix $M_4(i) = e_{i,16} - e_{i,15}$, where $1 \leq i \leq N$, that contains the result of this computation. We used these pre-processed measurements data as input for our correlation analysis:

$$c_i = C(M_4, M_2(1 : N, i)), \qquad 0 \leq i \leq 2^{kg} - 1. \tag{11}$$

As it is shown in Figure 8, the highest correlation occurs when the key guess is $1A_{hex} = 30_{dec}$. This value corresponds to the correct 6 MSBs of the round key 16. As a consequence, the attack is **practically successful**, i.e. the selected prediction matrix is sufficiently correlated with the real measurements and we can extract the key information.

More key bits may be found using the same set of measurements. The attacker only has to modify the selected prediction matrix and to target different key bits. We may also repeat the same experiment with a target of 8 bits (in place of

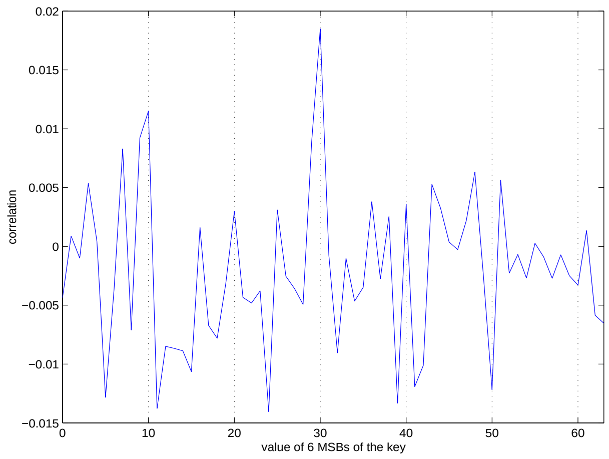


Fig. 8. A correlation attack with real measurements

4 bits in previous sections) corresponding to the output bits of two S-boxes (Figure 4(b)). Because we have to predict 12 key bits, we expect that the correlation coefficient for the right key guess will be higher than in a 4-bit experiment. According to the figures 18,19,20 given in the appendixes, this expectation is true. However, Figure 19 illustrates that more plaintexts are needed to determine the correct key guess. This phenomenon is well known in classical cryptanalysis and is due to the fact that wrong key guesses may be correlated with the correct one, e.g. in linear cryptanalysis [31].

10 Conclusions and further research

We showed the specificity of SRAM-based FPGAs in the context of Differential Power Analysis. Although the original attack developed in [1] can hardly be applied to these reconfigurable devices, we generalized the model of power attacks in order to take the physical behavior of FPGAs into account. The resulting attack is effective with reasonable measurement material. It is more efficient than the popular "multiple-bit" DPA and allows interesting theoretical predictions of the attacks with simulated data. Moreover, the power consumption model and therefore the efficiency of the attack are susceptible to be improved in different ways, for example by taking advantage of implementation netlists and delay information as we suggest in section 4.

As FPGAs does not present significant differences with ASICs, most of the conclusions presented in this work could probably be extended to other CMOS-based circuits. Other block ciphers (especially AES Rijndael) are also susceptible to be defeated with similar methods. Finally, the theoretical model developed in this work could also be used to investigate the electromagnetic side-channel of CMOS circuits. These last points should deserve additional research.

References

1. P.Kocher, J.Jaffe, B.Jun, *Differential Power Analysis*, in the proceedings of CRYPTO 99, Lecture Notes in Computer Science 1666, pp 398-412, Springer-Verlag.
2. S.B.Ors, E.Oswald, B.Preneel, *Power-Analysis Attacks on an FPGA - First Experimental Results*, in the proceedings of CHES 2003, Lecture Notes in Computer Science, vol 2279, pp 35-50, Springer-Verlag.
3. F.X. Standaert, L.van Oldeneel, D.Samyde, J.J. Quisquater, *Power Analysis of FPGAs, How Practical is the Attack?*, in the proceedings of FPL 2003, Lecture Notes in Computer Science, vol 2278, pp 701-711, Springer-Verlag.
4. E.Brier, C.Clavier, F.Olivier, *Optimal Statistical Power Analysis* , IACR e-print archive 2003/152.
5. Xilinx: *Virtex 2.5V Field Programmable Gate Arrays Data Sheet*, http://www.xilinx.com.
6. E.Oswald, *On Side-Channel Attacks and the Application of Algorithmic Countermeasures*, Phd Thesis, Institute for Applied Information Processing and Communications (IAIK), TU-Graz, 2003.
7. National Bureau of Standards. *FIPS PUB 46*, The Data Encryption Standard. U.S. Departement of Commerce, Jan 1977.
8. NIST Home page, http://csrc.nist.gov/CryptoToolkit/aes/.
9. G.Rouvroy, F.X.Standaert, J.J.Quisquater, J.D.Legat, *Design Strategies and Modified Descriptions to Optimize Cipher FPGA Implementations: Fast and Compact Results for DES and Triple-DES*, in the proceedings of FPL 2003, LNCS 2778, pp 181-193, Springer-Verlag, 2003.
10. T.S.Messerges, E.A.Dabbish, R.H.Sloan, *Examining Smart-Card Security under the Threat of Power Analysis Attacks*, IEEE transactions on computers, Vol.51, N5, May 2002.
11. P.Kocher, J.Jaffe, and B.Jun, *Introduction to Differential Power Analysis and Related Attacks*, Cryptography Research 607 Market Street, 5th Floor San Francisco, CA 94102, www.cryptography.com.
12. L.Goubin, J.Patarin, *DES and Differential Power Analysis*, in the proceedings of CHES 1999, Springer LNCS, vol 1717, pp 158-172.
13. J.S.Coron, P.Kocher, D.Naccache, *Statistics and Secret Leakage*, in the proceedings of Financial Crypto 2000, Springer LNCS, vol 1972, pp 157-173.
14. J.M.Rabaey, *Digital Integrated Circuits*, Prentice Hall International, 1996.
15. N.Weste, K.Ashraghian, *Principles of CMOS VLSI Design: A system Perspective*, Addison-Wesley, 1985.
16. S.M. Kang, Y.Leblebici, *CMOS Digital Integrated Circuits: Analysis and Design*, McGraw-Hill, 1996.
17. J.Oldfield, R.Dorf, *Field Programmable Gate Arrays*, Wiley, 1995.
18. S.Brown, R.Francis, J.Rose, Z.Vranesic, *Field Programmable Gate Arrays*, Kluwer Academic Publishers, 1992.
19. J.Rose, T.J.Francis, D.Lewis, P.Chow, *Architecture of Field Programmable Gate Arrays: The Effect of Logic Block Functoinality on Area Efficiency*, IEEE Transactions on Solid State Circuits, Vol 25, Num 5, pp 1217, October 1990.
20. J.Rose, S.Brown, *Flexibility of Interconnection Structures for Field Programmable Gate Arrays*, IEEE Transactions on Solid State Circuits, Vol 26, Num 3, pp 277, March 1991.
21. P.Chow et al, *A 1.2μm CMOS FPGA Using Cascaded Logic Blocks and Segmented Routing*, in FPGAs, W.Moore and W.Luk editors, Abingdon, UK, pp 91, 1991.

22. P.Chow et al, *The Design of an SRAM-based Field Programmable Gate Array - Part 1 : Architecture*, IEEE Transactions on Very Large Scale Integrated Circuits, Vol 7, Num 2, pp 191, June 1999.
23. P.Chow et al, *The Design of an SRAM-based Field Programmable Gate Array - Part 2 : Circuit Design and Layout*, IEEE Transactions on Very Large Scale Integrated Circuits, Vol 7, Num 3, pp 321, September 1999.
24. S.Brown and J.Rose, *Architecture of FPGAs and CPLDs: A Tutorial*, IEEE Design and Test of Computers, Vol. 13, No. 2, pp 42, 1996.
25. D.Lewis et al, *The Stratix Routing Architecture*, FPGA 2003, Monterey, California, 2003.
26. K.Padalia, *Automatic Transistor-Level Design and Layout Placement of FPGA Logic and Routing from an Architectural Specification*, University of Toronto Engineering Science Bachelor's Thesis, May 2001.
27. A.D.Garcia, *Etude sur l'estimation et optimisation de la consommation de puissance des circuits logiques programmables du type FPGA*, Phd Thesis, Ecole Nationale Superieure des Telecommunications, Paris, 2000.
28. K.Weiss et al, *Power Estimation Approach for SRAM-based FPGAs*, FPGA 2000, Monterey, California, 2000.
29. L.Shang, A.Kaviani, K.Bathala, *Dynamic Power Consumption in Virtex2 FPGA Family*, FPGA 2002, Monterey, California, 2002.
30. L.T. Mc Daniel, *An Investigation of Differential Power Analysis Attacks on FPGA-based Encryption Systems*, Master Thesis, Virginia Polytechnic Insitute and State University, May 29, 2003.
31. M.Matsui, *Linear Cryptanalysis Method for DES Cipher*, Lecture Notes in Computer Science, Advances in Cryptology, proceedings of EUROCRYPT'93, pp. 386-397, 1993.

Appendixes.

Static RAM cell: The static RAM cell used to store configuration bits in FPGAs is usually a 6-transistor cell based on a SR flip flop. $\overline{BL}$ acts as the reset signal and BL as the set signal. During a write operation, a 0 (resp. 1) is written in the cell by setting BL to 0 (resp. 1) and $\overline{BL}$ to 1 (resp. 0). During a read operation, we assume that both bit lines are precharged to 5V before the read operation is initiated. The read-cycle is started by asserting the word line, enabling both pass transistors M_5 and M_6. Then, during the read event, the values stored in Q and $\overline{Q}$ are transferred to the bit lines.

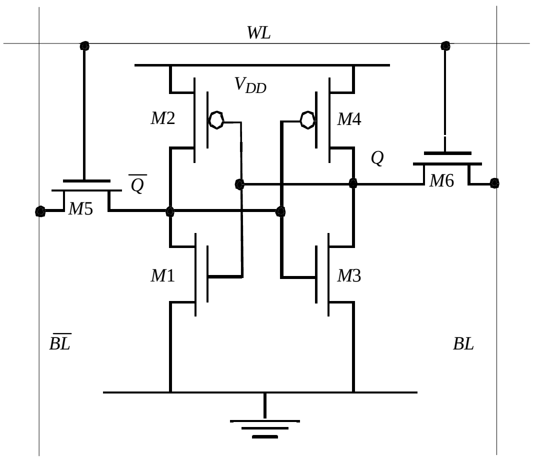


Fig. 9. Static RAM cell.

Fig. 10. Virtex slice.

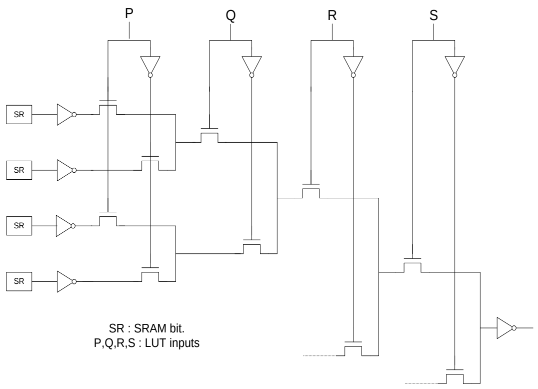


Fig. 11. Top quarter of a lookup table.

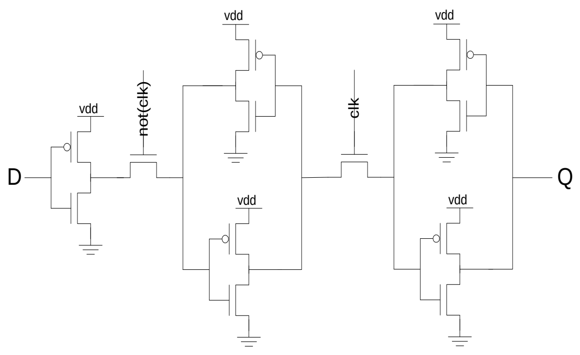


Fig. 12. D flip flop.

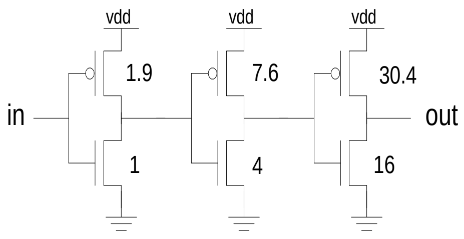


Fig. 13. Buffer.

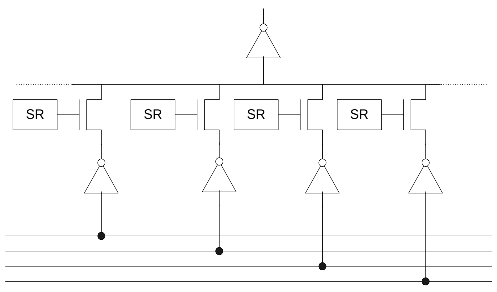


Fig. 14. Connections using pass transistors.

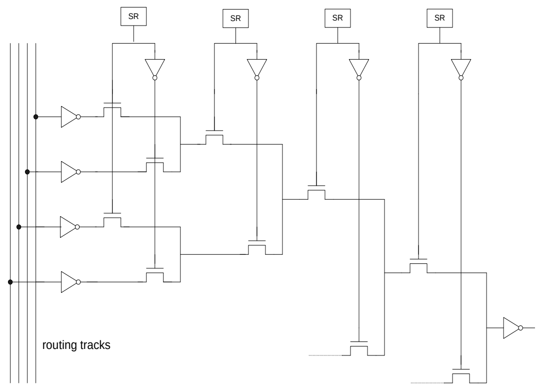


Fig. 15. Connections using multiplexors.

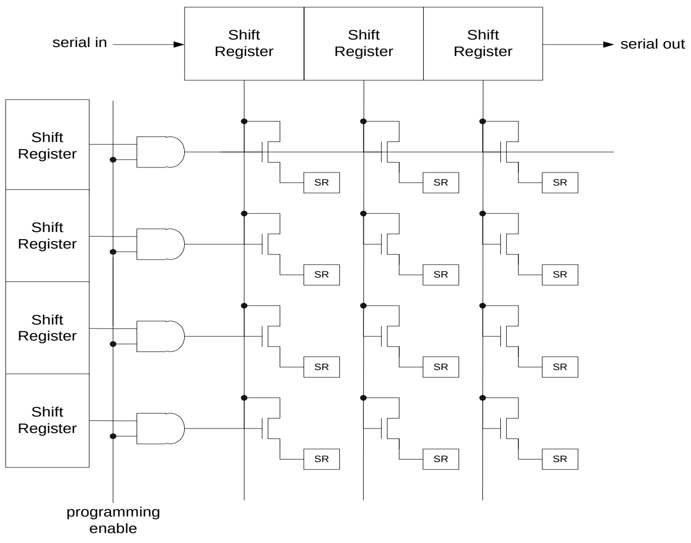


Fig. 16. Programming the FPGA.

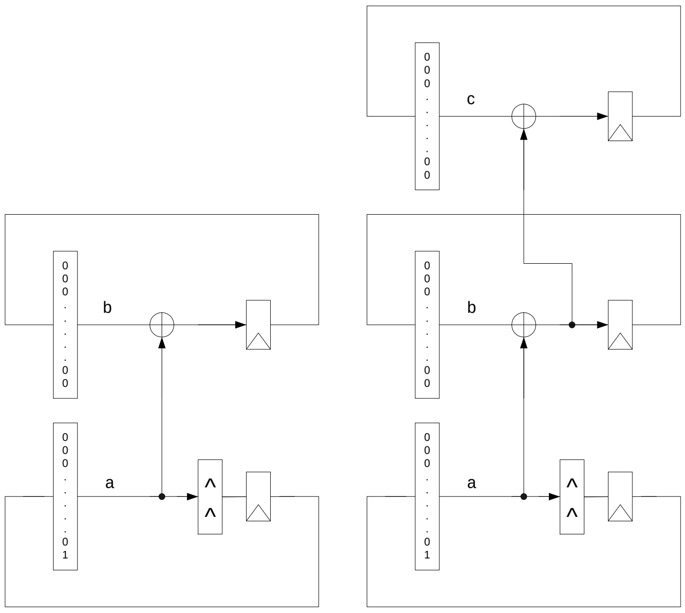


Fig. 17. Preliminary test design.

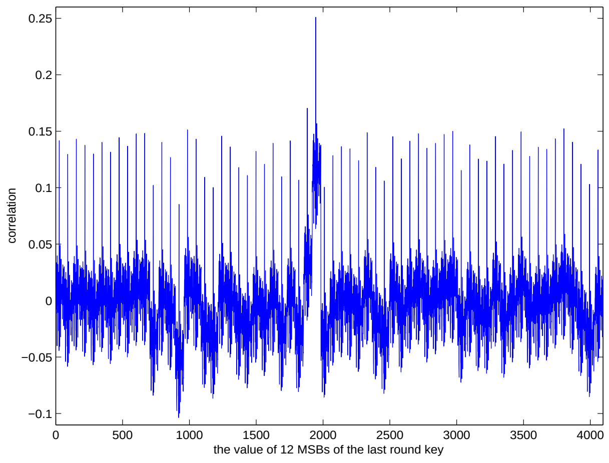


Fig. 18. A correlation attack with real measurements, 12-bit experiment.

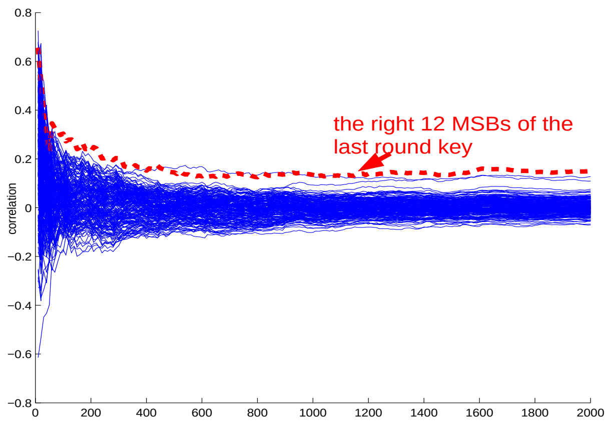


Fig. 19. A correlation attack using simulated data for different N values, 12-bit experiment.

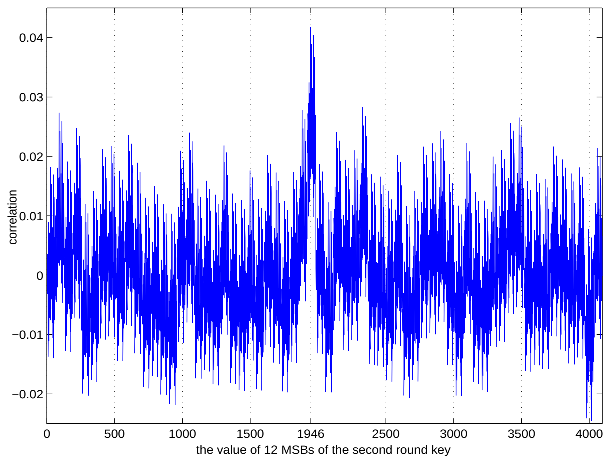


Fig. 20. A correlation attack with real measurements, 12-bit experiment.