Power Analysis of an FPGA Implementation of Rijndael: Is Pipelining a DPA Countermeasure?

François-Xavier Standaert[1], Sıddıka Berna Örs[2], Bart Preneel[2]

[1]UCL Crypto Group, Laboratoire de Microélectronique
Université Catholique de Louvain,
Place du Levant, 3, B-1348 Louvain-La-Neuve, Belgium
standaert@dice.ucl.ac.be

[2]Katholieke Universiteit Leuven, Dept. ESAT/SCD-COSIC,
Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium.
siddika.bernaors,bart.preneel@esat.kuleuven.ac.be

Abstract. Since their publication in 1998, 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. However, most published attacks apply to smart cards and only a few publications assess the vulnerability of hardware implementations. In this paper we investigate the vulnerability of Rijndael FPGA (Field Programmable Gate Array) implementations to power analysis attacks. The design used to carry out the experiments is an optimized architecture with high clock frequencies, presented at CHES 2003. First, we provide a clear discussion of the hypothesis used to mount the attack. Then, we propose theoretical predictions of the attacks that we confirmed experimentally, which are the first successful experiments against an FPGA implementation of Rijndael. In addition, we evaluate the effect of pipelining and unrolling techniques in terms of resistance against power analysis. We also emphasize how the efficiency of the attack significantly depends on the knowledge of the design.

1 Introduction

Side-channel analysis is becoming a classical topic in cryptographic design, but although numerous papers investigate Differential Power Analysis (DPA) from a theoretical point of view, only a few articles focus on their practical implementation. Moreover, most of the published research is related to smart cards and only a few papers assess the context of hardware and FPGA implementations.

As soon as hardware design is concerned, the questions of effectiveness, clock frequency and area requirements are of primary importance. In this paper, we demonstrate that they also have a very substantial impact on the feasibility of power analysis attacks. For this purpose, we investigated an optimized FPGA implementation of the Advanced Encryption Standard Rijndael [1, 2], presented at CHES 2003. In addition to the practical evaluation of the attack, we present

a number of original observations concerning: (*i*) the effect of pipelining and unrolling techniques in terms of resistance against power analysis attacks; (*ii*) the relationship between the knowledge of a hardware design and the efficiency of power analysis attacks. (*iii*) the effect of high clock frequencies on the measurement setup. Moreover, we characterized some design components (*e.g.* the registers) in terms of *predictablility* and *leakage*. This results in tools that could be used to analyze power analysis attacks in general. Finally, we compare our results with the only published attack against a hardware implementation of Rijndael that we are aware of [3] to validate our conclusions.

This paper is structured as follows. Section 2 presents the hypothesis used to carry out the power analysis attack and Section 3 gives a short description of our Rijndael implementation. Section 4 describes how to perform theoretical predictions on the power consumption in a pipeline design and Section 5 explains how to use these predictions in order to mount a practical attack. Section 6 presents theoretical predictions of the attack and their practical implementation is discussed in Sect. 7. Additional considerations about pipeline and unrolled designs are presented in Sect. 8. Section 9 re-discusses the hypothesis. Finally, conclusions are in Sect. 10.

2 Hypothesis

In Differential Power Analysis, an attacker uses a hypothetical model of the device under attack to predict its power consumption. These predictions are then compared to the real measured power consumption in order to recover secret information (*e.g.* secret key bits). The quality of the model has a strong impact on the effectiveness of the attack and it is therefore of primary importance.

While little information is available on the design and implementation of FPGAs (much of the information is proprietary), we can make assumptions about how commercial FPGAs behave at the transistor level. The most popular technology used to build programmable logic is static RAM[1], where the storage cells, the logic blocks and the connection blocks are made of CMOS gates. For these circuits, it is reasonable to assume that the main component of the power consumption is the dynamic power consumption. For a single CMOS gate, we can express it as follows [5]:

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

where C_L is the gate load capacitance, V_{DD} the supply voltage, $P_{0\to 1}$ the probability of a $0 \to 1$ output transition and f the clock frequency. Equation (1) specifies that the power consumption of CMOS circuits is data-dependent. However, for the attacker, the relevant question is to know if this data-dependent behavior is observable. This was confirmed by the following test.

Let three 4096-bit vectors be defined as follows. Initially, $a_0 = 00000...001$ and $b_0, c_0 = 00000...000$. Then:

$$a_{i+1} = SL(a_i), \qquad b_{i+1} = b_i \oplus a_i, \qquad c_{i+1} = c_i \oplus b_i,$$

[1] For all the experiments, we used a Xilinx Virtex XCV800 FPGA [4].

where SL is the shift left operator and consecutive values (x_i, x_{i+1}) are separated by a register. It is easy to see that:

- a is a bit-vector with a constant Hamming weight ($H(a) = 1$). The position of the 1-bit inside the vector is incremented/decremented from 0 to 4095.
- b is a bit-vector for which the Hamming weight is incremented/decremented from 0 to 4095.
- c is a bit-vector for which the number of bit switches between two consecutive states is incremented/decremented from 0 to 4095.

A design that generates these three vectors was implemented in the FPGA.

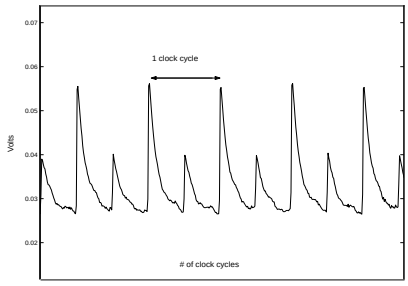

Fig. 1. One single power trace

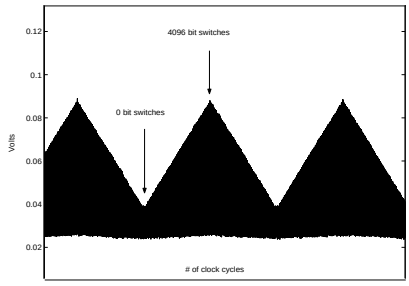

Fig. 2. Preliminary test

Figure 1 illustrates[2] a single power trace. Figure 2 illustrates the power consumption of vectors a, b and c during about 20 000 clock cycles. From this experiment, we conclude that the power consumption clearly depends on the number of transitions in registers.

Based on these considerations, we used the following **hypothesis** to mount power analysis attacks against FPGAs: "an estimation of a device power consumption at time t is given by the number of bit transitions inside the device registers at this time". Predicting the transitions in registers is reasonable since registers usually consume the largest part of the power in a design.

3 Hardware Description

A short description of the Rijndael algorithm is given in the Appendix A. The architecture used to investigate DPA against Rijndael was presented last year at CHES 2003 [6]. We briefly describe its significant details.

SubBytes: The substitution box (S-box) is implemented as a 256 x 8 multiplexer and takes advantage of specific properties of the FPGA. Note that two pipeline stages are inserted for efficiency purposes, as represented in Appendix B. In SubBytes, this S-box is applied to the 16 bytes of the state in parallel.

[2] Measurement setups for DPA have already been intensively described in the open literature. In Fig. 1, we observe the voltage variations over a small resistor inserted in the supply circuit of the FPGA. Every trace was averaged 10 times in order to remove the noise from our measurements.

MixAdd: In [6], an efficient combination of MixColums and the key addition is proposed, based on an optimal use of the FPGA resources. The resulting MixAdd transform allows MixColumns and AddRoundKey to be computed in two clock cycles, the key addition being embedded with MixColumns in the second cycle.

Complete architecture: The complete architecture is represented in Fig. 3, where all the registers are 128-bit long[3]. It is a loop architecture with pipeline, designed for optimizing the ratio $Throughput\ (Mbits/s)/Area\ (slices)$. It is important to remark that the multiplexer model for the S-box implies that its first part uses four 128-bit registers. The resulting design implements the round (and key round) function in 5 clock cycles and the complete cipher in 52 clock cycles.

4 Predictions in a pipeline design

The question we assess in this paper is to know whether pipelining has any influence on DPA resistance. We also investigate a practical design that is the result of efficiency optimizations. Loop architectures are a relevant choice for investigation because they satisfy the usual area and throughput requirements for block cipher applications. However, unrolled architectures will also be explored in a further section.

Based on the hypothesis of Sect. 2, the first step in a power analysis attack is to make theoretical predictions on the power consumption. This can be done using a selection function D that we define as follows. Let X_i and X_{i+1} be two consecutive values inside a target register. An estimation of the register power consumption at the time of the transition is given by the function $D = H(X_i \oplus X_{i+1})$. An attacker who has to predict the transitions inside the registers of an implementation therefore needs to answer two basic questions:

1. Which register transitions can we predict?
2. Which register transitions leak information?

Answering these questions determines which registers will be targeted during the attack. As an attacker can use the plaintexts (*resp.* ciphertexts) and predict transitions by partial encryption (*resp.* decryption), it is also important to evaluate both scenarios.

4.1 Definitions

i. The *predictability* of a register is related to the number of key bits one should know to predict its transitions. For block ciphers, this depends on the size of the S-boxes and the diffusion layer. In practice, it is assumed that it is possible to guess up to 16 key bits, and the diffusion layer usually prevents guessing of more than one block cipher round. In Rijndael, S-boxes are 8-bit wide and their outputs are thus *predictable* after the first (*resp.* final) key addition. However, every MixColumns output bit depends on 32 key bits and is therefore computationally intensive to guess.

[3] Except the first part of Mixadd that is 176-bit long.

ii. We denote a register as a *full* (*resp. empty*) register if its transitions leak (*resp.* do not leak) secret information. For example, it is obvious that an input (*resp.* output) register does not leak any secret information as it only contains the plaintext (*resp.* ciphertext). A surprising consequence of the hypothesis introduced in Sect. 2 is that the registers following an initial (*resp.* final) key addition do not leak information either. To illustrate this statement, we use the following key addition:

AddKey
$\{ \ result = input \oplus key; \ \}$

Let assume that the *result* is actually stored in an FPGA register R . Let two consecutive inputs of the key addition be denoted as $input_1$ and $input_2$. Using the previously defined selection function, the register power consumption may be estimated by:

$$\begin{aligned} P_R \propto H(result_1 \oplus result_2) &= H(input_1 \oplus key \oplus input_2 \oplus key) \\ &= H(input_1 \oplus input_2) \end{aligned} \tag{2}$$

Equation 2 clearly specifies that the register R is *empty*. In practice, registers of our Rijndael implementation will actually remain *empty* as long as the state has not passed through the non-linear S-box. Thereafter, the power consumption depends on $H(sbox(input_1 \oplus key) \oplus sbox(input_2 \oplus key))$ and therefore on the key.

Remark that this observation strongly depends on the hypothesis and selection functions used to perform the attack, what we will discuss further in Sect. 9. Another surprising observation is that the register R may still leak secret information if reset signals are used. This is due to the constant state that reset signals introduce. Then, we have:

$$\begin{aligned} P_R \propto H(\text{“}all\ zeroes\text{”} \oplus result_1) &= H(\text{“}all\ zeroes\text{”} \oplus input_1 \oplus key) \\ &= H(input_1 \oplus key) \end{aligned} \tag{3}$$

which makes the power consumption dependent on the key again. As a consequence, a secure hardware implementation should not apply reset signals to its inner registers in order to delete this additional information leakage. Note that a similar observation has been used to attack smart card implementations, where the constant state actually corresponds to a constant instruction address.

4.2 Predictions in Rijndael

Figure 3 illustrates *predictable* and *full* registers when our AES design is filled with 5 different texts, denoted 1,2,. . . ,5, during the first eight clock cycles of an encryption. As an example, during the first cycle, register $R1$ contains the plaintext 1 while all the other registers are undefined. During the second cycle, $R1$ contains the plaintext 2, $R2$ contains the plaintext 1 and the other registers are undefined. Remark that in the eighth cycle, the multiplexer starts to loop and register $R3$ therefore contains data corresponding to plaintext 1 again.

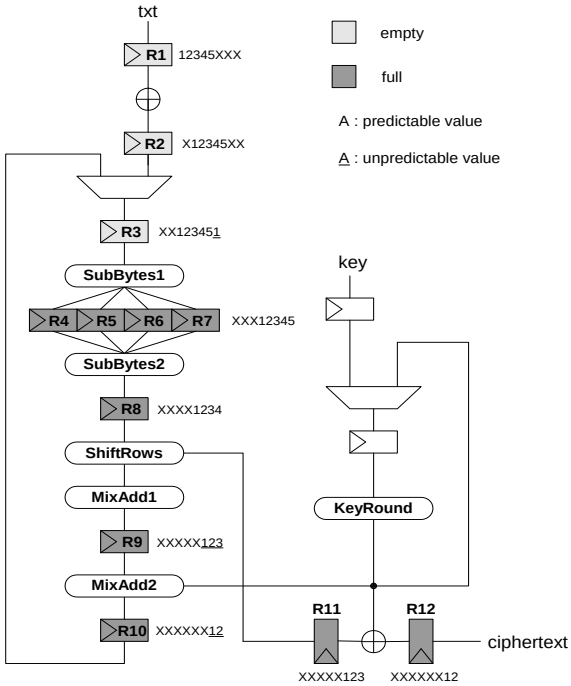

Fig. 3. Encryption predictions.

Similarly, Figure 4 illustrates *predictable* and $full$ registers when our AES design is filled with 5 different texts, denoted 1,2,. . . ,5, during the last six clock cycles of an encryption. As an example, the register $R12$ contains the first ciphertext in the second cycle, ciphertext 2 in the third cycle and ciphertext 3 in the fourth cycle.

In the next section, we explain how theoretical predictions of the power consumption can be used to attack an FPGA implementation of Rijndael.

5 Description of a correlation attack

A correlation attack [3, 7] against an FPGA implementation of Rijndael is divided into three steps. Let N be the number of plaintext/ciphertext pairs for which the power consumption measurements are accessible. Let K be the secret encryption key. When simulating the attacks, we assume that K is known to the attacker. In case of practical attacks, it is of course unknown.

Prediction phase: For each of the N encrypted plaintexts, the attacker first selects the target registers and clock cycle for the previously defined selection function D. In Fig. 3, we see that between cycles 7 and 8, registers $R4, R5, R6, R7, R8, R11$ and $R12$ are $full$ and have *predictable* and defined values. Similarly, in Fig. 4, we observe that between cycles 1 and 2, registers $R3, R4, R5, R6, R7$ and $R10$ are $full$ and have *predictable* and defined values.

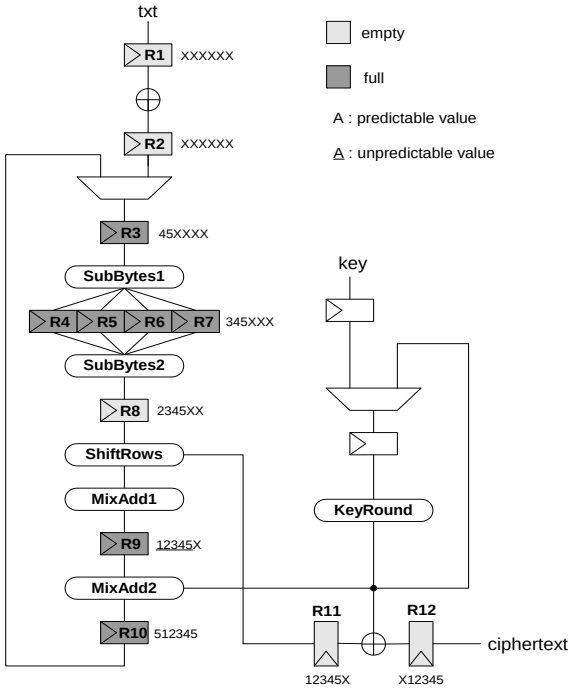

Fig. 4. Decryption predictions.

Depending on the knowledge of the design, these registers can therefore be targeted. Due to the size of the Rijndael S-box, the predictions are performed on 8 bits and may be repeated for every 8-bit part of a register R_i.

Let t be the number of 8-bit registers targeted by the attacker. Then, he predicts the value of D (*i.e.* the number of bit switches inside the target registers in the targeted clock cycle) for the 2^8 possible key guesses and N plaintexts. The result of the prediction phase is an $N \times 2^8$ **selected prediction matrix**, containing integers between 0 and $8 \times t$. For simulation purposes, it is also interesting to produce the **global prediction matrix** that contains the number of bit switches inside all the 12 registers[4] of the design, for all the cycles. That is, if the encryption is performed in 52 clock cycles, we obtain a $N \times 52$ matrix, containing integers between 0 and $12 \times 128 = 1536$. This is only feasible if the key is known. In accordance with the hypothesis of Sect. 2, these matrices give estimations for the power consumption of the device.

Measurement phase: During the measurement phase, we let the FPGA encrypt the same N plaintexts with the same key, as we did in the prediction phase. While the chip is operating, we measure the power consumption for the 52 consecutive clock cycles. Then, the power consumption trace of each encryption

[4] 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 as a first approximation.

is averaged 10 times in order to remove the noise from our measurements and we store the maximum values of each encryption cycle so that we produce an $N \times 52$ matrix with the power consumption values for all the texts, cycles. We denote it as the **global consumption matrix**.

Correlation phase: In the correlation phase, we compute the correlation coefficient between a column of the global consumption matrix (corresponding to the cycle targeted by the prediction phase) and all the columns of the selected prediction matrix (corresponding to all the 2^8 key guesses). If the attack is successful, we expect that only one value, corresponding to the correct key guess, leads to a high correlation coefficient.

An efficient way to perform the correlation between theoretical predictions and real measurements is to use the Pearson coefficient. Let M_i denote the ith measurement data (*i.e.* the ith trace) and M 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:

$$C(M,P) = \frac{E(M.P) - E(M).E(P)}{\sqrt{Var(M).Var(P)}}. \tag{4}$$

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

Finally, theoretical predictions of the attack can be performed by using the global prediction matrix instead of the global consumption matrix. As the global prediction matrix contains the number of bit switches inside all the registers, it represents a theoretical noise free measurement and may help to determine the minimum number of texts needed to mount a successful attack, *i.e.* an attack where the correct key guess leads to the highest correlation coefficient. This is investigated in the next section.

6 An attack using simulated data

In this section, we study the influence of the number of registers predicted on the efficiency of the attack. Different scenarios can be considered that correspond to different abilities of the attacker. In the most basic case, the attacker does not have any information about the design and has to make assumptions about its implementation. A reasonable assumption is that the S-box outputs will be stored in registers[5]. Therefore, the attacker will only predict the switching activity of 8 bits in $R8$ (in encryption) or $R3$ (in decryption). In the first step of the simulated attack, we produce the **selected prediction matrix** and **global prediction matrix** as defined in the previous section. Thereafter, we perform the correlation phase between these two matrixes. If the attack is successful, we expect that only one value, corresponding to the correct key guess, leads to a high correlation coefficient.

[5] This is usually the case in Rijndael because S-boxes are the most time (and space) -consuming parts of the algorithm.

As the attacker is interested to determine the minimum number of plaintexts necessary to extract the correct key, we calculated this correlation coefficient for different values of $N : 1 \leq N \leq 4096$. As shown in Fig. 5.(A), after approximately 1500 plaintexts the right 8 key bits can be distinguished from a wrong guess. We may therefore say that the attack is **theoretically successful** after about 1500 texts.

In a more advanced scenario, the attacker has access to some implementation details (for example the scheme of Fig. 3) and may determine the *predictable* and $full$ registers. Based on the complete predictions of Fig. 3, the correlation coefficient values for every key guess and different numbers of traces are represented in Fig. 5.(B). We observe that the correct key guess is distinguishable after about 500 plaintexts, but stays closely correlated to 3 other candidates. The explanation of this phenomenon can be found in the implementation details of the substitution box represented in the annexes (Figure 6). As the S-box is a large multiplexer with two pipeline stage, 6 input bits are actually used to select the values in registers $R4, R5, R6, R7$. Thereafter, two last bits select the final result of $R8$. As a consequence, if the key guess is such that the first 6 input bits of the S-box remain unchanged, the values stored in registers $R4, R5, R6, R7$ will be the same. Only the S-box output in register $R8$ will differ. As there are 4 such key guesses, we will have 4 closely correlated candidates, including the correct one, what we can clearly observe in Fig. 5.(B).

A solution to this problem is to use the decryption predictions of Fig. 4. Then, even if only one bit differs at the output of the S-box (in $R8$), it will not result in the same intermediate register transitions. Based on these predictions, the correlation coefficient values for every key guess and different number of traces are represented in Fig. 5.(C), where the correct key candidate is clearly distinguishable after about 500 traces.

7 An attack using practical measurements

When attacking a device practically, the selected prediction matrix remains unchanged while we replace the global prediction matrix by the real measured global consumption matrix. Therefore, we let the FPGA encrypt 4096 plaintexts with the same key as we did in the previous section and produced the matrix as described in Sect. 5.

To evaluate the quality of our theoretical predictions, we made a preliminary experiment and computed the correlation coefficient between one (in practice the 26th) column of the **global prediction matrix** and every column of the **global consumption matrix**. Figure 5.(D) clearly illustrates that the highest correlation value appears for the predicted round, and therefore confirms that our predictions are correlated with real measurements.

In order to identify the correct 8 MSBs of the final round key, we used the correlation coefficient again. As it is shown in Fig. 5.(E), the correct key guess is distinguishable after about 1000 traces. As a consequence, the attack is **prac-**

tically successful, *i.e.* the selected prediction matrix is sufficiently correlated with the real measurements and we can extract the key information. Remark that comparing Figures 5.(C) and 5.(E) allows us to evaluate the effect of the measurement phase. Compared with smart cards, the sampling process was made more difficult by the high clock frequency of the Rijndael design (around 100 MHz). Note also that the noise was removed from the measurements by an averaging process, but this step could be removed or reduced if the measurement setup was improved. Nevertheless, due to the specificities of our acquisition device[6], the averaging was directly done during the measurement step and did not increase the memory requirements of the attack. If we compare these results with the only published power analysis attack against ASIC implementations of Rijndael [3], the quality of our measurements seems to be better. Moreover, we need significantly less plaintexts for the attack to be practically successful.

Finally, it is important to note that more key bits may be found using exactly the same set of measurements. The attacker only has to modify the **selected prediction matrix** and target different key bits. The full key can therefore be recovered computing the correlation between the **global consumption matrix** and 16 predictions, each one revealing 8 key bits.

8 Adding more pipeline

Previous sections emphasized that pipelining a loop implementation of Rijndael does not provide any efficient protection against DPA. However, the predictions of Sect. 6 also reveal that when only one register (*e.g.* R8 in Fig. 4.(A)) is predicted, we need significantly more traces than when several registers are predicted. The efficiency of an attack against a loop implementation is notably due to the fact that most registers are predictable, because only one round is implemented. In case of unrolled and pipelined implementations, the situation strongly differs, as only the outer rounds are partially predictable. As a consequence, the inner rounds may be viewed as noise generators and therefore act as a well known DPA countermeasure. Although noise addition does not fundamentally counteract power analysis attacks (the signal is still present and may still be recovered), it has the advantage of decreasing the correlation between predictions and measurements. Moreover, if the noise is added in the form of unrolled pipeline stages, it does not reduce the efficiency of an implementation. Finally, the method introduced in Sect. 5, allows us to theoretically predict the effect of unrolled architectures with pipelining on resistance against DPA.

A first step to predict the effect of pipeline stages is to investigate the theoretical number of bit switches in a register. In the following, we will assume that the rounds of a block cipher behave like a random number generator. In practice, this is only true after a few rounds, when the diffusion is complete. Based on this hypothesis, we may predict the probability $P(x, n)$ of having x bit switches between two states S_1, S_2 in an n-bit register:

[6] Tektronix TDS 7104 oscilloscope.

$$P(x,n) = P(H(S_1 \oplus S_2) = x) = \frac{C_{n,x}}{2^n} \quad (5)$$

As a consequence, the number of bit switches is distributed as a binomial which can be approximated by a Gaussian distribution with parameters $\mu = n/2$ and $\sigma^2 = n/4$. It is therefore possible to predict the number of bit switches in registers of arbitrary size.

For example, in the design of Fig. 3, we observe that one round is implemented in 5 cycles, using eight 128-bit registers. Its transitions may be simulated as a Gaussian distributed random noise with parameters $\mu = 512$ and $\sigma^2 = 256$. In general, if an n-round unrolled implementation is considered, we can add Gaussian distributed random noise with parameters $\mu = (n-1).512$ and $\sigma^2 = (n-1).256$ to our previously computed global prediction matrix and then compute the correlation with the selected prediction matrix.

The result of an attack using simulated data with 10 rounds unrolled and 5 pipeline stages per round is illustrated in Fig. 5.(F), where we used the same **selected prediction matrix** as in the previous section. While the correct key guess still has the best correlation value, we clearly observe that the correlation value was significantly reduced if we compare with Fig. 5.(C), making a practical attack much more difficult.

9 Hypothesis (2)

Looking back at the hypothesis of Sect. 2, it is important to evaluate how the work presented in this paper could be improved and how representative are our results. To the question "Are power analysis attacks realistic against efficient FPGA implementations of Rijndael?" we may certainly answer "yes". While attackers usually investigate "toy" implementations for side-channel attacks, we took a real and optimized design with high clock frequencies and evaluated the significance of pipelining techniques in terms of DPA resistance. From an attacker's point of view, we have investigated the simplest possible hypothesis and built a practical attack based on these simple assumptions. However, the question "How to counteract these power analysis attacks?" is still open in different ways. When countermeasures are considered, it is important to note that our measurements can be improved in several ways; moreover, the attack model should be taken into account.

As an illustration, we limited the transition predictions to the registers of the Rijndael design. However, it is clear that registers are not the only leaking parts in FPGAs and transitions in other components could be predicted in order to improve the attack. Similarly, looking back at Equation (1), a more accurate prediction of the FPGA power consumption could be done by evaluating the load capacitance values. A notifiable feature of FPGAs is that they are made of different resources (*e.g.* logic blocks, connections) of which the power consumption differs because of different effective load capacitances. As a consequence,

the power consumption of FPGA designs does not only depend on their switching activity but also on the internal resources used. In practice, 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 [8]. 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 relevant; it will allow an attacker to improve the attack.

Finally, more advanced attack scenarios are possible, *e.g.* taking the key scheduling into account, or using more complex power consumption models. The measurement setup could also be improved and therefore the gap between theoretical predictions of the attacks and practical results would be reduced. To conclude, this paper used a simple leakage model and we could already recover secret information. However, as far as Boolean masking (or other countermeasures) are concerned, it is certainly not sufficient to mask the transitions in registers only and there are other leakage sources to investigate and prevent.

10 Conclusions

We have investigated a power analysis attack against a practical FPGA implementation of Rijndael and have exhibited the effect of pipelining and unrolling techniques in this context. It is first demonstrated that pipelining a loop implementation does not provide an effective countermeasure if an attacker has access to the design details because most of the registers in the pipeline remain predictable. Then we illustrate how the combination of pipelining and unrolling techniques may counteract power analysis attacks as a random noise generator. We also provide a theoretical model allowing the simulation and comparison of the attacks in different contexts. In practice, we have mounted the first successful attack against an efficient FPGA implementation of Rijndael. Finally, a clear discussion of the hypothesis used to perform power analysis is provided with some proposals for further improvements.

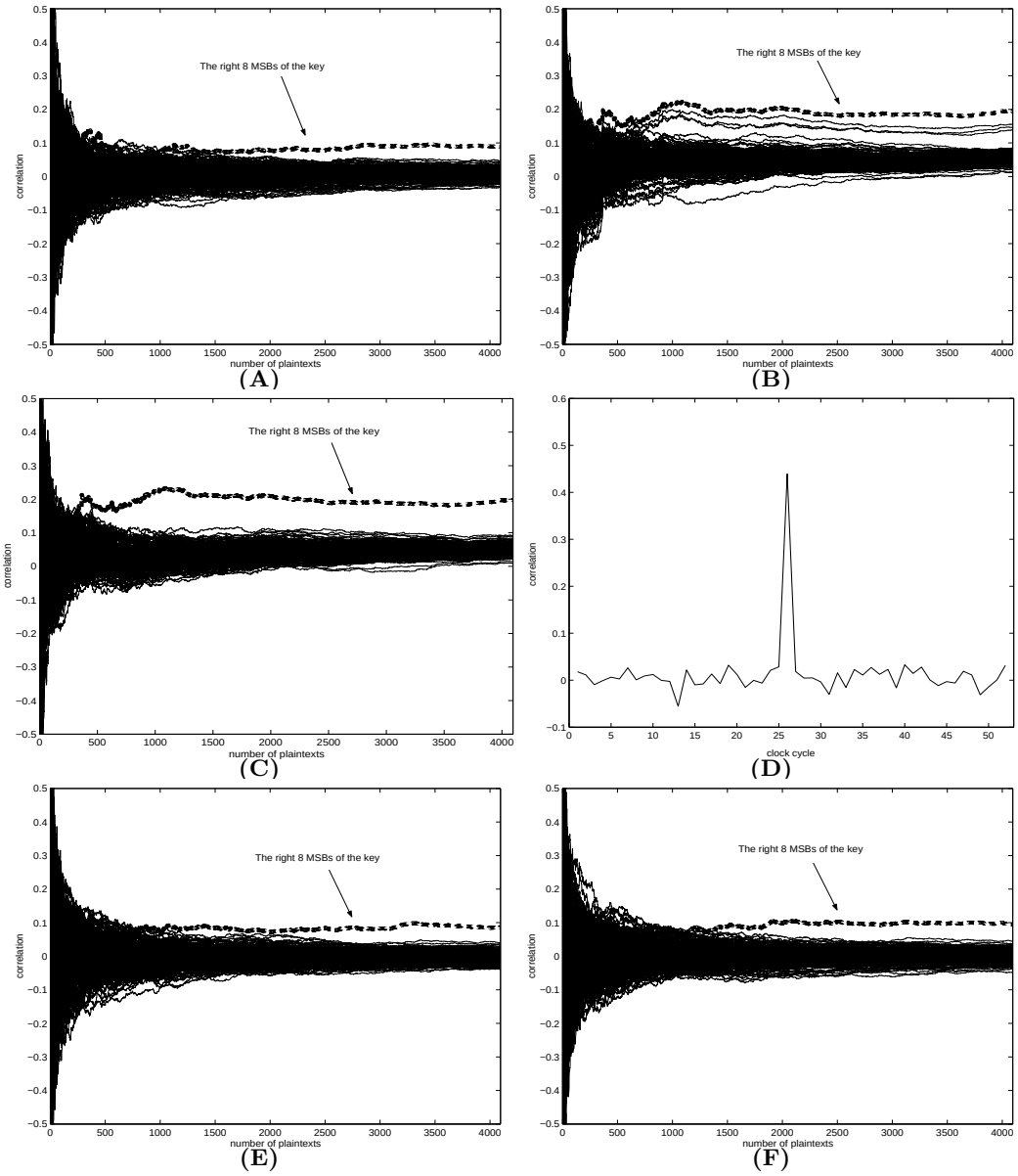

Fig. 5. **(A)** A simulated attack using register R8 only.
(B) A simulated attack using complete encryption predictions (R4,R5,R6,R7,R8,R11,R12).
(C) A simulated attack using complete decryption predictions (R3,R4,R5,R6,R7,R10).
(D) Correlation between global predictions for cycle 26 and measurements ($N = 4096$).
(E) A correlation attack with real measurements.
(F) A simulated attack against an unrolled implementation.

References

1. J. Daemen, V. Rijmen, *"The Design of Rijndael. AES – The Advanced Encryption Standard,"* Springer-Verlag, 2001.
2. FIPS 197, *"Advanced Encryption Standard,"* Federal Information Processing Standard, NIST, U.S. Dept. of Commerce, November 26, 2001.
3. S.B.Ors, F.Gurkaynak, E. Oswald, B. Preneel *Power-Analysis Attack on an ASIC AES implementation*, in the proceedings of ITCC 2004, Las Vegas, April 5-7 2004.
4. Xilinx: *Virtex 2.5V Field Programmable Gate Arrays Data Sheet*, http://www.xilinx.com.
5. J.M.Rabaey, *Digital Integrated Circuits*, Prentice Hall International, 1996.
6. F.-X.Standaert, G.Rouvroy, J.-J.Quisquater, J.-D.Legat, *Efficient Implementation of Rijndael Encryption in Reconfigurable Hardware: Improvements and Design Tradeoffs*, in the proceedings of CHES 2003, Lecture Notes in Computer Science, vol 2779, pp 334-350, Springer-Verlag, 2003.
7. E.Brier, C.Clavier, F.Olivier, *Optimal Statistical Power Analysis* , IACR e-print archive 2003/152.
8. 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.
9. 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.

A Short description of Rijndael

Rijndael is an iterated block cipher with a variable block length and a variable key length. The block length and the key length can be independently specified to 128, 192 and 256 bit. This paper focusses on the 128-bit version. The algorithm consists of a serial of 10 applications of a key-dependent round transformation to the cipher state and the round is composed of four different operations. In pseudo C, we have:

```
Round(state,roundkey)
{
SubBytes(state);
ShiftRows(state);
MixColumns(state);
AddRoundKey(state,roundkey);
}
```

SubBytes is a non-linear byte substitution operating on each byte of the state independently. ShiftRows is a cyclic shift of the bytes of the state. In MixColumns, the columns (1 column = 4 bytes) of the state are considered as polynomials over GF(2^8) and multiplied modulo $x^4 + 1$ with a fixed polynomial. Finally, AddRoundKey is a bitwise XOR with the bits of the key.

Rijndael's initial 128-bit key is expanded to eleven 128-bit roundkeys by means of a key scheduling algorithm. Although the key scheduling is also implemented

in hardware, its description is not necessary for the understanding of the paper and we will consider it as a black box.

Finally, the complete cipher consists of an initial roundkey addition, 10 rounds and a final round where MixColumns has been removed. In pseudo C, we have:

```
Rijndael(state,cipherkey)
{
KeyExpansion(cipherkey,expandedkey[0..10]);
AddRoundKey(state,expandedkey[0]);
for (i=1;i<10;i++)
        {
        Round(state,expandedkey[i]);
        }
SubBytes(state);
ShiftRows(state);
AddRoundKey(state,expandedkey[10]);
}
```

A more detailed view of the Rijndael algorthm can be found in [1].

B Implementation of the substitution box

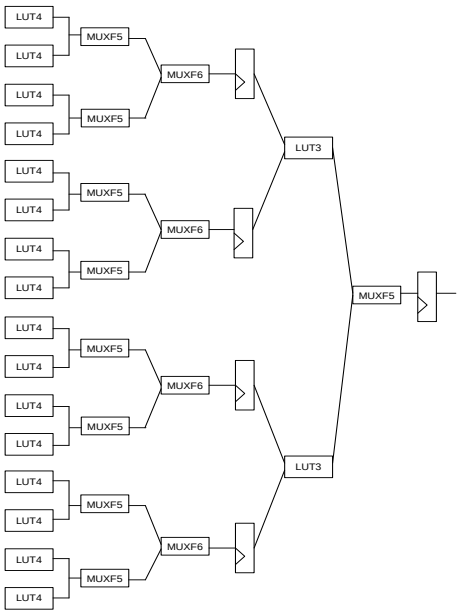

Fig. 6. One output bit of the substitution box.