SCARE of an Unknown Hardware Feistel implementation

Denis Réal[1,2], Vivien Dubois[1], Anne-Marie Guilloux[1], Frédéric Valette[1], and Mhamed Drissi[2]

[1] CELAR, 35 Bruz, France
{Denis.Real;Vivien.Dubois:Anne-Marie.Guilloux;Frederic.Valette}@dga.defense.gouv.fr
[2] INSA-IETR, 20 avenue des coesmes, 35043 Rennes,France
Rennes, France
{Denis.Real;Mhamed.Drissi}@insa-rennes.fr

Abstract. Physical attacks based on Side Channel Analysis (SCA) or on Fault Analysis (FA) target a secret usually manipulated by a public algorithm. SCA can also be used for Reverse Engineering (SCARE) against the software implementation of a private algorithm. In this paper, we claim that an unknown Feistel scheme with an hardware design can be recovered with a chosen plaintexts SCA attack. First, we show that whatever is the input of the unknown Feistel function, its one-round output can be guessed by SCA. Using this relation, two attacks for recovering the algorithm are proposed : an expensive interpolation attack on a generic Feistel scheme and an improved attack on a specific but commonly used scheme. Then, a countermeasure is proposed.

1 Introduction

Cryptographic algorithms are designed for being robust against logical analysis. However, the activity of any given device (smart cards, FPGA, microprocessor...) filter through side channels such as the processing time, the power consumption or the electromagnetic radiations. Secret values are then vulnerable to statistical attacks such as Differential Power Analysis (DPA) [4] or Correlation Power Analysis (CPA) [1], but also to one-shot attacks such as Template Attacks (TA) [2] [8].

Side Channel Analysis For Reverse Engineering (SCARE) techniques are also employed for recovering a private algorithm. R. Novak proposed a strategy for identifying a substitution table of a secret implementation of the GSM A3/A8 algorithm [6]. The feasibility of a SCARE for symmetric cryptography have been proved by recovering a software DES implementation [3]. Indeed, the intermediate results on one round are available by SCA: they are computed sequentially due to the software design. In this article, we go further with a SCARE attack against an hardware implementation of a generic Feistel Scheme. The SCA feasibility is demonstrated on a hardware DES implemented on an ASIC . However, the side channel leakage exploitation is done for a generic Feistel Scheme. Then, we propose a countermeasure against this generic SCARE attack.

2 The Hardware Implementation of a Generic Feistel Scheme

2.1 Assumption on the Feistel Scheme Design

The logic function is an expensive resource on a cryptographic device, especially on a smart card. As a consequence, the Feistel function is usually implemented once and the input registers are updated at each round. For example, the right part of the first round output is xored with the left part of the plaintext and is assumed to be written on a register whose value was the right part of the plaintext. This realistic design assumption is detailed on Fig. 1.

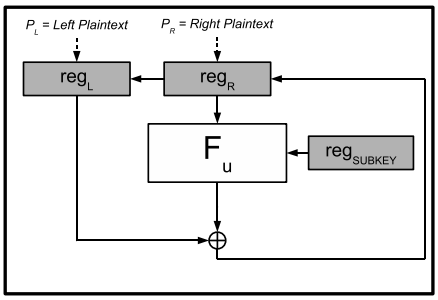

Fig. 1. Assumption on a Generic Feistel Scheme Design

2.2 The power consumption during a Bit Transition

The power consumption of a D Flip-Flop (DFF) depends on its current activity. Due to CMOS transistor behavior, register state changes make the DFF consume power [9]. When the logic value 1 is applied a DFF input while the previous value was 0, a current rise occurs on the rising edge of the clock. Furthermore, the current rise during this transition is higher than the static state holding dissipation. Then, we approximate the circuit behavior by saying that it supplies energy for bit transition and does not supply energy for holding a bit value. Lines supplying energy to the circuit can be observed with electromagnetic near field techniques. The effect of bits transitions occurring at the same instant are also built up on the side channel electromagnetic radiations. This is the power supplying line leakage model. Even if it is experimentally harder, an attacker can also observe the current at the DFF level with the same kind of techniques [5]. The radiations at the DFF level do not behave the same than previously. Then a transition from 0 to 1 induces a radiation opposite to the radiation due to a transition from 1 to 0 due to current motions in the DFF. This is transistor current leakage model. The choice of the leakage model depends on the part of

the circuit targeted by the probe. State of the art electromagnetic techniques permits to target power supplying line: the DFF level is harder to get. The SCA we propose matches with the power supplying line leakage model.

3 Side Channel Analysis of a Generic Feistel Scheme

3.1 Side Channel Identification of a Feistel Scheme

SCA techniques permit to identify a Feistel scheme. Indeed, the right part of the plaintext is used once on the 1^{st} round as R_0 round but also on the 2^{nd} round as L_1. Then, a CPA mean computed on the right part of the plaintext contains two peaks (one on the 1^{st} round and the other on the 2^{nd} round). A CPA computed on the left part of the plaintext contains one peak one on the 1^{st} round. Fig. 2 illustrates the effect on a DES hardly implemented. The electromagnetic radiations are measured using near field techniques and a probe sensitive to the vertical magnetic field. Fig. 2.1 shows one ciphering operation. Fig. 2.2 is a CPA on the right part of the plaintext and Fig. 2.3 is a CPA on the left part of the plaintext.

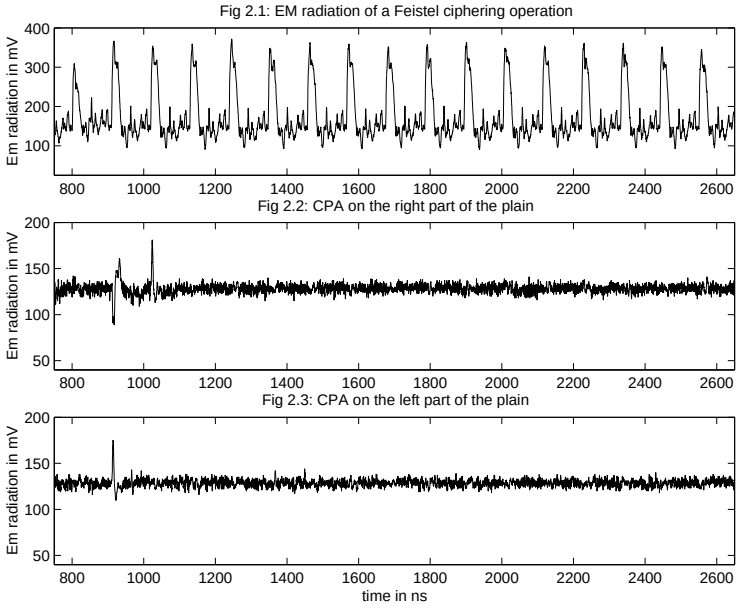

Fig. 2. CPAs on the plaintext

A Feistel Scheme can then be identified with two DPA traces. This method for characterizing a generalized Feistel Scheme can also be used with ciphertexts.

3.2 SCA for to guess the one round output of the Feistel Function

Our SCA is a chosen plaintext attack. We assume that the attacker is able to spy N ciphering operations. Let P_i be the corresponding plaintexts with L_i and R_i their corresponding left and right hand halves, $i < N$. The right part of the plaintext is chosen fixed to R: $R_i = R$. Let R_i^j and L_i^j be the input of the j^{th} round of the feistel scheme : $R_i^0 = R_i = R$, $L_i^0 = L_i$ and $L_i^1 = R_i$. Let F_u be the unknown Feistel function and Y_i^j its corresponding output for the j^{th} round : $Y_i^j = F_u(R_i^j)$. Let Y be the fixed output of F_u for the round 0: $Y_i^0 = F_u(R) = Y$ and $R_i^1 = L_i \oplus Y$. The notations for this chosen plaintexts context are reminded in Fig. 3 and Fig. 4.

- N : The number of plaintexts.
- P_i : The i^{th} plaintext with $i < N$.
  - L_i : The corresponding left part.
  - $R_i = R$: The corresponding fixed right part .
- P_i : The unknown Feistel Function.
  - L_i^j : The left input of the j^{th} round : $L_i^0 = L_i$.
  - R_i^j : The right input of the j^{th} round : $R_i^0 = R$.
- Y_i^j : The output of F_u at the the j^{th} round.
  - Y : The fixed output of F_u after the round 0 $Y = F_u(R)$.
  - X : The CPA object, classification according to the hamming weight of $L_i \oplus X$.

Fig. 3. Notations

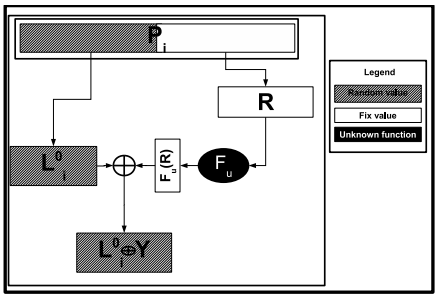

Fig. 4. SCA context

We consider 4-bits words: the leakage of 4-bits can usually be properly observed while an exhaustive search on 4 bits in not very expensive.This choice corresponding to the S-box size of the DES is purely coincidental. Let $L_{i,0}$, X_0

and R_0 be the first 4 bits of L_i, R and X. Here, we propose an SCA for guessing X_0, the generalization to the other words of X being trivial. For each possible value α of X_0, we compute the side channel trace Γ_α being the mean trace computed using the plaintexts P_i whose first word is α: we obtain then 16 traces. Let X_0 be the index of the trace Γ_α which is minimal at the $t = T_0 + T_d$. We claim that $X_0 = Y_0 \oplus R_0$. Indeed, according to the assumption made on the Feistel design scheme, we clearly see that, at the instant $t = T_0 + T_d$ the value $L_i \oplus Y$ overwrites the value R on the register 2. The value of L_i that minimizes the number of bit transition occurring on the register 2 is $L_i = Y \oplus R$. The classification done computing the traces Γ_α is random regarding all the 4-bits words of L except the first one explaining we are able to isolate its leakage from the logical noise. Then we can guess the value of Y : $Y = X \oplus R$. This SCA attack permit to have the relation $Y = F_u(R)$ with X and Y known and F_u the unknown feistel function. Fig. 5 shows the results with $R_0 = 0 \times 3$ and $Y_0 = 0 \times B$ and $N = 5000$. Fig. 5.1 illustrates a ciphering operation useful for finding the instant $t = T_0 + T_d$. Fig. 5.2 shows the mean traces $\Gamma_{0\times 8}$ and $\Gamma_{0\times 7}$: $\Gamma_{0\times 8}$ is the minimum at $t = T_0 + T_d$ while, as expected $\Gamma_{0\times 7}$ is the maximum ($7 \oplus 8 = 0 \times F$). Fig. 5.3 shows the 14 other mean trace Γ_α. The difference before the instant $t = T_0 + T_d$ cannot be used because the signal is not time-stationary.

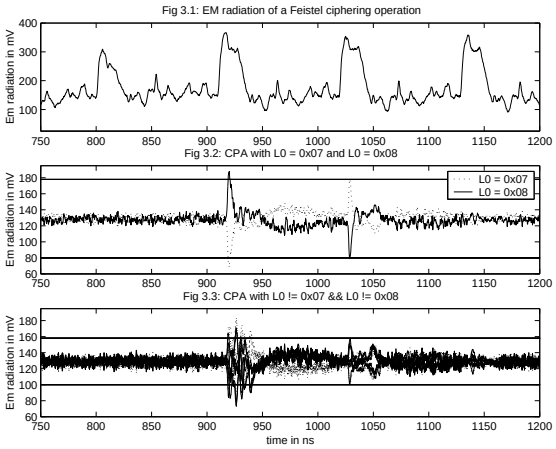

Fig. 5. Γ_α mean traces

Then, a low cost SCA permits to compute the output of the unknown Feistel function F_u of any input word R. This SCA aims at being general. In the case of measurements with a low level of noise as is observed here, the attacker can search $F_u(R)$ bit by bit. Indeed, he can compare two side channel traces linked to two plaintexts whose difference is one bit of the left halve : the correct bit value corresponds to the lower radiation. Then the attacker can expect to get $F_u(R)$ with 64 chosen plaintexts for DES.

4 The Cryptographic Attack derived from the SCA

In this section, we compute the number of (chosen) input-output pairs required to fully recover the unknown function F_u. We first compute this number for a general function F_u of given degree and input/output size. Secondly, we show that this number can be made much smaller when F_u follows a specific but commonly used design.

4.1 Simple Interpolation Attack

F_u is a vectorial function with n input/output bits. Its bitwise coordinates are polynomials of the n input bits of degree denoted d. For each input-output pair, we use the input and the j-th bit of the output to reconstruct the j-th coordinate of F_u. When the same input-output pairs can be used to reconstruct the individual coordinates of F_u, the number of input-output pairs that we need can be computed from the reconstruction of a single coordinate of F_u. Since each such polynomial has about $n^d/d!$ coefficients, we need this number of input-output pairs to be able to resolve the unknown coefficients by linear algebra. The computational cost of recovering one coordinate of F_u is therefore $n^{3d}/(d!)^3$ binary operations and n times this number to recover F_u entirely. For $n = 64$ and $d = 8$, this amounts to roughly 2^{32} input-output pairs and 2^{102} binary operations.

4.2 Improved Attack on a Class of Commonly Used Schemes

The specification of a general function F_u requires a large amount of storage for practical choices of degree and input size. For this reason, functions admitting a compact description are often considered in that place. A commonly encountered class of functions is built by composition of a random function of a reduced number of bits $d \ll n$ with a linear compression from n bits to d bits. In this case, each of the n coordinates of F_u has the shape $f \circ S$ where f is an unknown function from d bits to 1 bit and S is an unknown linear function from n bits to d bits. We next show how to recover f and S from a few input-output pairs of $f \circ S$.

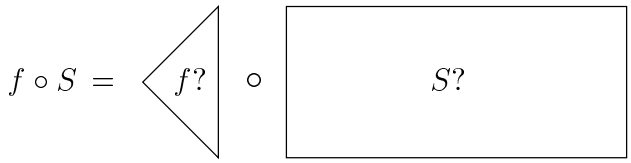

Fig. 6. A representation of $f \circ S$

First, we observe that considering a composed function $f \circ S$ the problem of recovering f and S has multiple solutions. Indeed, for any linear permutation ϕ

of the rows of S, the transformed $f \circ \phi^{-1}$ and $\phi \circ S$ are an alternative solution. As a consequence, when the first d columns of S are linearly independent, we can suppose that they actually form the Identity matrix. Then, considering the inputs of $f \circ S$ with their last $n-d$ bits to 0 actually provides us with input-output pairs of f. There are 2^d inputs of $f \circ S$ with their last $n-d$ bits to 0 and this is what we need to compute f by the interpolation method. Finally, the cost of recovering f is 2^d input-output pairs of $f \circ S$ and 2^{3d} binary operations.

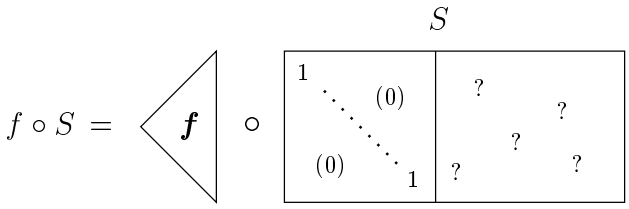

Fig. 7. Updated representation of $f \circ S$

We are now left at finding the coefficients of the $n-d$ last columns of S. For this, we sequentially resolve the coefficients of the k-th column for $k = d+1, \ldots, n$ by using inputs of $f \circ S$ with their k-th bit to 1 and their last $n-k$ bits to 0. At each step, the coefficients of the previous columns are known and each input-output pairs of $f \circ S$ with the prescribed bits to 0 defines an algebraic relation in the d unknown coefficients of the current column. Since we want to keep to a minimum the number of input-output pairs of $f \circ S$ that we use, we do not use these algebraic relations to solve the coefficients of the current column. Instead, we do exhaustive search on these coefficients and use the algebraic relations to identify their correct values. Since each algebraic relation is satisfied with probability about $1/2$, we need on average about d input-output pairs of $f \circ S$ to discriminate the correct values of all d coefficients. Applying this method to the $n-d$ unknown columns of S costs $(n-d)d$ input-output pairs of $f \circ S$ and $(n-d)2^d$ binary operations.

In all, our improved method allows to recover the complete description of $f \circ S$ from $2^d + (n-d)d$ chosen input-output pairs and using $2^{3d} + (n-d)2^d$ binary operations. For $n = 64$ and $d = 8$, this amounts to roughly $2^{9.5}$ input-output pairs and 2^{24} binary operations.

5 Proposition of Countermeasure

Hardware countermeasures are either expensive to design, either not efficient against statistical SCA [7]. We propose then the logic countermeasures based on the attacker capability.

5.1 Countermeasure with the supplying line leakage model

The current rise during the transitions from 0 to 1 or 1 to 0 is much higher than the static dissipation observed for the holding of state. The proposed SCA is based on the knowledge of the first value of a register. But, if the targeted register is precharged with a random before the secret-dependent value be written, the proposed SCA cannot be done. That countermeasure implementation costs one register more but also costs a random generator and specific means to load it in the register. However, random generator are usually implemented on smart cards. Fig. 8 illustrated that the precharging countermeasure design. During the initialization stage, the plaintext is written in a register while the random value is loaded in the other one. At the first round of the algorithm, right part of the plaintext is processed but the result is now written on the randomized register. A random value is needed at the 1^{st} round, the cryptographic algorithm making this register precharged value be random for the other rounds. This design divides the data rate by 2. As the 2 first and 2 last round of the algorithm are sensitive to our way of attacking, the designer can improve the data rate by precharging only these round which costs 4 clock periods.

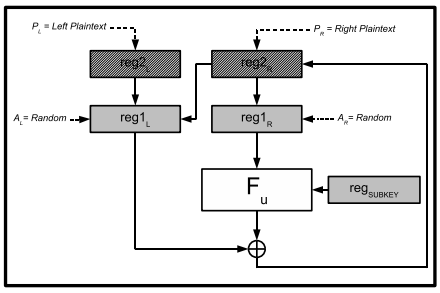

Fig. 8. Precharging countermeasure

Fig. 9 presents another way of designing that countermeasure. The Feistel function is unrolled on two round. The initialization consists then on loading a random value on the rising clock edge and the plaintext on the next rising clock edge. The data rate is not divided by 2 anymore, just 1 clock period is added. However, this design multiplies the logic by 2. With this design, two more logic had been implemented, the cost is then area and consumption not time execution anymore.

Precharging is a well-known countermeasure. However it seems to be inefficient in the case of DFF leakage model.

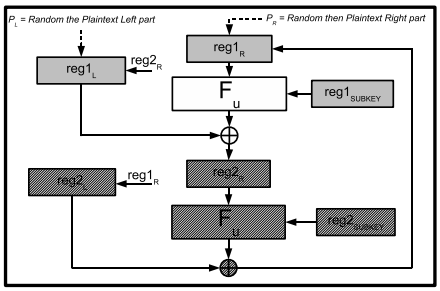

Fig. 9. Unrolling implementation

5.2 Improved Countermeasure in the case of DFF leakage model

If the falling or the rising transition can be distinguished, the proposed SCA is still possible, at the difference that the attacker does not search the minimal mean radiation but the medium mean radiation. This SCA bypasses the previous countermeasure. Indeed, the outputs on one round of the Feistel could be divided into three groups and instead of two groups. Then, at the bit level, an attacker who observe no transition cannot say anything. But a transition from 0 to 1 means that the observed bit of $L \oplus F_u(R)$ is 1 and a transition from 1 to 0 means that the observed bit of $L \oplus F_u(R)$ is 0. Then, the proposed attack work with the register precharging, it just costs more experiments. The previous countermeasure is then improved by working randomly with the plaintext or its complementary. Due to Feistel properties, the final result can be calculated whatever be the kind of input data. Then the attacker who observes a 0 in a register does not know if this bit is inverted or not. This countermeasure resists to 1^{st} order SCA. If F_u can be expressed as $F_u(R) = F_2(K \oplus A(R))$ with A an affine function, the countermeasure implementation can be improved as represented on Dig. 10.

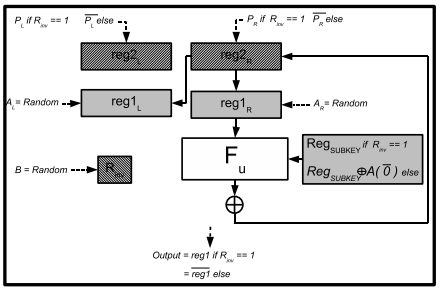

Fig. 10. Complementary Precharging

6 Conclusion

State of art in SCARE attack usually target software implementation. The attack we presented in this article is, to our knowledge, the first attack for reverse engineering a general Feistel with an hardware design. All the results proposed here can be applied to a generalized Feistel Scheme. We presented also an improved countermeasure mixing complementary register and precharging. This countermeasure should resist to 1^{st} order SCA. It has not been design for to resist to 2^{nd} order SCA. However, for that a 2^{nd} order SCA attack be a real threat, its feasibility should be studied in future works and its complexity clearly identified.

References

1. É. Brier, C. Clavier, and F. Olivier. Correlation power analysis with a kleakage model. In M. Joyeand and J-J. Quisquater, editors, *Cryptographic Hardware and Embedded Systems – CHES 2004*, volume 3156 of *LNCS*, pages 16 – 29. Springer-Verlag, 2004.
2. S. Chari, J. R. Rao, and P. Riohatgi. Template Attacks. In B. S. Kaliski Jr, Ç. K. Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2002*, volume 2523 of *LNCS*, pages 13 – 28. Springer-Verlag, 2002.
3. R. Daudigny, H. Ledig, F. Muller, and F. Valette. SCARE of the DES (side channel analysis for reverse engineering of the data encryptation standart). In J. Ioannidis, A. Keromytis, and M. Yung, editors, *Applied Cryptography and Network Security. International conference – ACNS 2005*, volume 3531 of *LNCS*, pages 393 – 406. Springer-Verlag, 2005.
4. T. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In M. Wiener, editor, *Advances in Cryptology – Crypto '99*, volume 1666 of *LNCS*, pages 388 – 397. Springer-Verlag, 1999.
5. S. Mangard, T. Popp, B. M. Gammel, and B. Jun. Side Channel Lekage of Masked CMOS Gates. In J. R. Rao and B. Sunar, editors, *CT-RSA 2005*, volume 3376 of *LNCS*, pages 361 – 365. Springer Verlag, 2005.
6. R. Novak. Side-channel attack on substitution blocks. In J. Zhou and M. Yung Y. Han, editors, *Applied Cryptography and Network Security. International conference – ACNS 2003*, volume 2846 of *LNCS*, pages 307 – 318. Springer-Verlag, 2003.
7. D. Réal, C. Canovas, J. Clédière, M. Drissi, and F. Valette. Defeating Classical Hardware Countermeasures: a New Processing for Side Channel Analysis. In *Desing Automation Test in Europe International conference – DATE 2008*, 2008.
8. C. Rechberger and E. Oswald. Practical Template Attacks. In C. H. Lim and M. Yung, editors, *International Workshop on Information Security Application – WISA*, volume 3325 of *LNCS*, pages 440 – 456. Springer, 2004.
9. S. M. Sze. *Semiconductor Devices: Physics and Technology*. John Wiley and Sons, Inc., 2002.