SCARE of the DES

(Side Channel Analysis for Reverse Engineering of the Data Encryption Standard)

Rémy Daudigny, Hervé Ledig,
Frédéric Muller, and Frédéric Valette

DCSSI Crypto Lab 51, Boulevard de Latour-Maubourg
75700 Paris 07 SP France
{Remy.Daudigny,Frederic.Muller,Frederic.Valette}@sgdn.pm.gouv.fr

Abstract. Side-Channel Analysis for Reverse Engineering (SCARE) is a new field of application for Side-Channel Attacks (SCA), that was recently introduced, following initial results on the GSM A3/A8 algorithm. The principle of SCARE is to use side-channel information (for instance, power consumption) as a tool to reverse-engineer some secret parts of a cryptographic implementation. SCARE has the advantage of being discrete and non-intrusive, so it appears to be a promising new direction of research.

In this paper, we apply the concepts of SCARE in the case of the block cipher DES. We measure the power consumption of a software DES executed on a target smart card and propose new methods to exploit this information. We manage to retrieve many details about the underlying device, including some constants used by the algorithm (*e.g.* permutation tables for the round function and for the key scheduling), but also interesting implementation choices (*e.g.* registers where subkeys are loaded). Of course some information was already known in our case, but situations can be envisaged where the designer would like to keep it secret.

An application of these methods is to reverse-engineer a proprietary algorithm, provided some information about its basic structure is know. Hence it illustrates the power of SCARE and demonstrates yet again the accuracy of Kerckhoff's principle. In addition, a better understanding of a cryptographic implementation can be a first step to mount more sophisticated Side Channel Attacks.

1 Introduction

Side-Channel Attacks (SCA) have been developed since 10 years to analyse the security of cryptographic functions in actual implementations. Compared to traditional attacks which exploit the standard input/output (*i.e.* plaintext and ciphertext) of a cipher to recover some secret data, SCA uses an auxiliary source of information to achieve the same goal. This side-channel information can originate from various types of leakage. The first example was due to Kocher [6] and was based on using timing information. Furthermore it was shown that the

power consumption of a cryptographic device could also reveal some useful information to an attacker [7]. Other attacks of the same vein use electro-magnetic radiations [3, 5] or fault injection [1, 2]. Many variations of these techniques have been studied in the recent years, and the general idea is always to recover some secret keys used by the cipher.

More recently it was proposed to exploit side-channel leakage for reverse engineering. This new idea consists in exploiting (for instance) the power consumption of a device to recover some secret or non-trivial details about the way cryptographic functions were implemented. This new technique is called SCARE (Side-Channel Analysis for Reverse-Engineering) and has initially been applied to the A3/A8 authentication and session key generation algorithm of the GSM standard. An initial attack was proposed by Novak [9] and was later extended by Clavier [4]. These results constitutes a new application of techniques developed for SCA.

In our opinion, SCARE is a very promising direction for future research. We believe it could be useful in two situations. First, there are cryptographic functions where some part of the specification is voluntarily kept secret by the designer. For instance, a proprietary cipher where the general structure is known, but some constants are kept secret. Secondly, think of the implementation itself: some information (although not of cryptographic nature) may need to remain secret. For instance, the techniques used by the developers, the order and the length of the instructions or even the registers in which the data are stored. This information is sensible because a company may want to protect its technology, but also because it can be useful for a side channel attack. In this last case, SCARE is just a preliminary step towards more sophisticated attacks. For instance, a thin understanding of the hardware is often useful to improve power attacks (by focusing on relevant portions of the power traces or by applying a dedicated attack depending on the hardware behaviour).

More generally, SCARE demonstrates once again the accuracy of Kerckhoffs' principle that the security of a cryptosystem should rely on the secrecy of the key and not the cryptosystem itself. It also demonstrates the difficulty to protect a cryptographic implementation in hostile environments. This new reverse-engineering method is expected to have a broad range of applications since it is efficient, discrete and non-intrusive. In the next section, we introduce some background about SCA against DES. Then we describe our algorithmic methods for SCARE, and describe the experimental results we obtained by analysing the behaviour of a target smart card. Finally, we discuss directions for further research.

2   Side-Channel Attacks and the DES

DES (Data Encryption Standard) is a well-known encryption standard adopted by the NBS in 1977 [10], and replaced since then by AES. DES is a 64-bit block cipher with 56-bits key based on a Feistel structure. We skip details about the algorithm specifications, which are very well known. The reader can refer to [10] to find more information.

Today DES is still widely used especially in cryptographic hardware devices thus it has been an important target for side-channel analysis, following the initial paper by Kocher [6]. Several attacks are known, the simplest of which is Simple Power Analysis (SPA), where the attacker observes directly one power trace of a DES execution to retrieve the value of manipulated data. Power consumption curves are typically not that easy to analyse, therefore advanced attacks are needed.

An interesting and famous technique is Differential Power Analysis (DPA) [7]. DPA is a statistical attack, requiring to analyse several messages with their corresponding power traces. More precisely, assume we obtain M power traces: $(\mathcal{T}_1, \ldots, \mathcal{T}_M)$ where the power consumption at time t is given by $\mathcal{T}_i(t)$. We assume this consumption is correlated with the data manipulated at time t. For instance if a S-box is computed, $\mathcal{T}_i(t)$ is correlated with the output of this S-box. However it is impossible to infer the data directly from one power trace, due to the noise in the measurement. A statistical attack is therefore necessary.

We consider a known-plaintext attacker. He targets the output of Sbox S_1 of round 1. We call this 4-bit output (a, b, c, d). It depends on 6 key bits and 6 plaintext bits. If the attacker guesses these 6 key bits, he can predict the value of a for each message. The corresponding power traces can thus be sorted in two groups G_0 and G_1 according to the predicted value of a. Group G_i corresponds to the value $a = i$. The "differential curve" is computed as

$$D(t) = \left| \frac{1}{|G_0|} \sum_{\mathcal{T}_i \in G_0} \mathcal{T}_i(t) - \frac{1}{|G_1|} \sum_{\mathcal{T}_i \in G_1} \mathcal{T}_i(t) \right|$$

This curve should present peaks, if we assumed the correct key bits and should be close to 0 otherwise. Indeed, for the instants t where the S-box S_1 is computed, the power consumption is correlated with a. Therefore, for the correct key, there is an important difference between the consumption of groups G_0 and G_1. Otherwise, there should be no significant difference between the two groups, hence the differential curve $D(t)$ should be close to 0. Of course, M needs to be large enough to eliminate the noise in the experiments.

To summarize, the attacker learns 6 key bits by statistical treatment of the power traces. There exists a similar attack that requires only knowledge of the ciphertext. In practice DPA is often more complicated: countermeasures can be implemented, so the analysis of power traces is rather subtle. Basically the attack can be improved if we know more about:

- which portions of the traces correspond to the computation of S_1. Then we can restrict the analysis to significant data.
- the electrical behaviour of the cryptographic device. There may be better ways to exploit the curves, assuming we understand well the correlation between internal data and power consumption.

Therefore it may be useful, before applying this attack to spend some time to understand better the target device. Then optimized versions of the attack can be applied. This is an important motivation for SCARE.

3   Methods and Goals of SCARE

3.1   Goals

In the context of SCARE, we no longer consider the point of view of an attacker but the one of a **reverse-engineer**. The primary goal of this new adversary is not to recover the secret key. He wants to know more about the implementation, which englobes several possible goals:

– **Learn secret constants**
   Sometimes, the general structure of an algorithm is known but some partial information is kept secret. For instance if a company develops a proprietary algorithm, it might prefer, for various reasons, to keep some constants secret. This situation has been encountered with the GSM authentication and session key generation algorithm A3/A8 [4, 9]. Some tables were kept secret, but it was later demonstrated that SCARE could be used to retrieve them.
– **Learn more about the algorithm**
   Consider a secret proprietary block cipher. Several reasonable assumptions about its general structure (Feistel cipher, SP network) can be made. A reverse-engineer expects to obtain more details using SCARE. For instance, how many S-boxes are used, what is the size of these S-boxes, . . .
– **Learn information about the implementation techniques**
   The way an algorithm was implemented can sometimes be considered as a secret by itself. It reveals secrets about the technology used by a company or methods used by the developers of its products.
– **Understand better the device**
   Knowing which instruction corresponds to which portion of the power trace improves the efficiency of many attacks (like DPA). Moreover, it is helpful to understand better the correlation between intermediate data and power consumption (for instance, what consumption model should be used).

3.2   Methods

In [4, 9], dedicated methods have been proposed in the case of the A3/A8 algorithm. They allowed to recover secret tables which were part of the algorithm specifications but kept secret. In this section, we propose and apply new methods to reverse-engineer a commercial smart card where a software DES implementation is available.

   The first operation consists in monitoring the power consumption of a DES encryption. We mention that similar attacks could be envisaged if we monitor the electro-magnetic radiations. For each message, we obtain a trace which is represented by a collection of values, each of them associated with a certain time index t. The preliminary stage consists in synchronizing the curve in order for a given time index t to uniquely correspond to a given instruction. This pre-processing is usually done prior to most Side Channel Attacks. Afterwards, we denote by $\mathcal{T}_i(t)$ the power consumption of the i-th trace at time index number t. Roughly, we can assume that each time index corresponds to one clock cycle.

Next, the goal of the reverse-engineer is to determine when each data is manipulated. If he knows nothing about the underlying algorithm, all he can do is detect when the algorithm inputs (such as the plaintext or the key) are manipulated. If he knows more, he can focus on any intermediate value, as long as no unknown material is needed to predict it. For instance, if we replace the DES permutation table by a random permutation, a reverse-engineer can predict intermediate values of the first round, until the new permutation is performed.

To determine when each data is manipulated, the reverse-engineer applies a statistical analysis to the power traces. For each intermediate bit he can predict, he tries to determine the time index t where it is manipulated. We call this result the **scheduling information** of the considered implementation. Afterwards, the reverse-engineer analyzes this information, hoping to learn the unknown material about the target algorithm. This second phase is detailed in Section 4.

Method to obtain scheduling information. Pick an intermediate bit a. The input consists in M messages and their corresponding power traces $(\mathcal{T}_j)_{1 \leq j \leq M}$. We want to determine the clock cycles where a is manipulated. If no information about the cipher is initially known, we can start by choosing a to be a bit from the plaintext.

- Build two groups G_0 and G_1 according to the value of a in each encryption (G_i corresponds to $a = i$).
- For each clock cycle i compute the following value V_i

$$V_i = \frac{1}{|G_0|} \sum_{\mathcal{T}_j \in G_0} \mathcal{T}_j(i) - \frac{1}{|G_1|} \sum_{\mathcal{T}_j \in G_1} \mathcal{T}_j(i)$$

- If $|V_i| > \lambda$ for some appropriate threshold λ we decide that the bit a is manipulated at time i. In our experimental settings, we fixed the threshold afterwards, in order to keep only a short number of significant time index.

To summarize, the indicator V_i just measures how much the consumption at clock cycle i is affected by a. Hence large values of $|V_i|$ reveal when the bit a is manipulated.

An example. We present some experimental results obtained on our target smart card. Of course, we know the specification of DES. However, we first worked as if the algorithm was secret, hence all we know is the plaintext. Therefore we focused on obtaining the **scheduling information of the plaintext bits**. The initial permutation of DES only modifies the order in which plaintext bits are manipulated, so it does not really change the analysis.

We use $M = 1000$ power consumption traces. We pick a to be any plaintext bit, say the first input bit of the first S-box S_1 for instance. For each clock cycle i, we determine if a is manipulated or not by testing if $|V_i| > \lambda$. The unit of the V_i is arbitrary and has been normalized to range in the set $[0, 200]$. The threshold λ has been set to 10, so that only a small number of clock cycles i verify

$$|V_i| > \lambda$$

Table 1. A sample of experimental results.

Cycle i	Value V_i	Decision
1074	7	Bit NOT manipulated at clock cycle 1074
1075	14	Bit manipulated at clock cycle 1075
1076	50	Bit manipulated at clock cycle 1076
1077	24	Bit manipulated at clock cycle 1077
1078	−8	Bit NOT manipulated at clock cycle 1078
1079	14	Bit manipulated at clock cycle 1079
1080	76	Bit manipulated at clock cycle 1080
1081	41	Bit manipulated at clock cycle 1081
1082	8	Bit NOT manipulated at clock cycle 1082
1083	9	Bit NOT manipulated at clock cycle 1083

A significant portion of the results is represented in Table 1. For most of the other time indexes, the outcome is that the considered bit was not manipulated. Further analysis has revealed that the depicted interval corresponds to the computation of S-box S_1. A larger sample is represented in Appendix A. The sign of V_i may also reveal some partial information (for instance, a change of sign probably means that 1 was XORed to the manipulated bit), however we do not take into account this information.

4   Analysis of Our Results

In this section, we describe how to interpret the scheduling information obtained in Section 3.2. We first work as if the DES specifications were unknown, so we only exploit the scheduling information of the plaintext bits. This allows us to determine the expansion table of DES. Next, we demonstrate similar methods to retrieve other information about the algorithm.

4.1   Application to the Expansion Table

The round function of DES starts by expanding the input from 32 to 48 bits. The DES specification contains a table describing this expansion [10] (see Table 2). In this section, we show how to retrieve this table using SCARE. Of course a similar analysis could be applied to a cipher using a secret expansion table.

Table 2. The DES Expansion Table.

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

Let us focus on scheduling information corresponding to plaintext bits (this analysis can even be performed when nothing is initially known about the cipher). Scheduling information tells us when each single bit is manipulated. But we can also observe **the groups of bits which are manipulated together**. Such groups are likely to correspond to S-box inputs.

Table 3. Scheduling information of the first round input.

Cycle	First Round Input								
	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	...	Bit 31	Bit 32
1073	-	-	-	-	-	-		-	-
1074	-	-	-	-	-	-		-	-
1075	Y	-	-	-	Y	-		-	-
1076	Y	Y	Y	Y	Y	-		-	Y
1077	Y	Y	Y	Y	Y	-		-	Y
1078	-	Y	Y	Y	Y	-		-	Y
1079	Y	Y	Y	-	Y	-		-	Y
1080	Y	Y	Y	Y	Y	-		Y	Y
1081	Y	Y	Y	Y	Y	-		Y	-
1082	-	Y	Y	Y	Y	-		-	-
1083	-	Y	Y	Y	Y	-		-	-
1084	-	-	-	-	-	-		-	-
1085	-	-	-	-	-	-		-	-

As an illustration, Table 3 contains a relevant portion of the scheduling information. For sake of simplicity, the exact values of the indicator V_i are replaced by 'Y' (YES) when $|V_i|$ exceeds the threshold $\lambda = 10$ and '-' otherwise. In this sample, input bits number 1, 2, 3, 4, 5 and 32 are clearly manipulated together around clock cycles $1075, \ldots, 1082$. These 6 bits form the first line of the DES expansion table.

We observe that the bit number 31 is also manipulated at cycles 1080 and 1081. This phenomenon can be due to noise in the experiment. Another possible interpretation is that the bit 31 was stored in the same register than bit 32 and is manipulated, while bit 32 is loaded.

A similar property could be observed for input bits number 4, 5, 6, 7, 8 and 9 in the following clock cycles. This group forms exactly the second line of the DES expansion table. Similarly we can learn the rest of the expansion table.

More generally, the structure of the round function can be detected this way. Indeed SCARE reveals which plaintext bits are manipulated together. This reveals the S-box structure. For instance, an attacker knowing nothing about DES could suspect that it uses 8 S-boxes, each of them applied to 6 input bits.

4.2 Application to the S-Box Tables

We have not implemented any attack to recover the DES S-boxes. However it is likely that the methodology of SCARE could also be used here. It has already

been verified experimentally in the case of the A3/A8 algorithm [4, 9]. Secret
substitution tables have been recovered in the case, using the power traces of
the cryptographic device.

4.3   Application to the S-Box Outputs

Now, we suppose that we know the expansion table and the DES S-box. Hence,
we can predict the 32 output bits of the S-box layer and analyze the scheduling
information concerning these bits. We observe several significant intervals, but
focus first on a particular interval which was already analyzed in Table 3. We
know it corresponds to the S-box computations. Other significant intervals are
further investigated later in the paper.

Table 4. Scheduling information of the S-box outputs.

Cycle	First Round Output								
	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8	...
1074	-	-	-	-	-	-	-	-	
1075	-	-	-	-	-	-	-	-	
1076	Y	Y	Y	Y	Y	-	Y	-	
1077	Y	Y	Y	Y	-	-	-	-	
1078	Y	Y	Y	Y	-	-	-	-	
1079	-	-	-	-	-	-	-	-	
1080	Y	Y	Y	Y	-	-	Y	-	
1081	Y	Y	Y	Y	-	-	-	-	
1082	Y	Y	Y	Y	-	-	-	-	
1083	-	-	-	-	-	-	-	-	
1084	-	-	-	-	-	-	-	-	
1085	Y	Y	Y	Y	-	-	-	-	
1086	Y	Y	Y	Y	-	-	-	-	
1087	Y	Y	Y	Y	-	-	-	-	
1088	Y	Y	Y	Y	-	-	-	-	
1089	Y	Y	Y	Y	-	-	-	-	
1090	Y	Y	Y	Y	-	-	-	-	
1091	Y	Y	Y	Y	-	-	-	-	
1092	-	-	-	-	-	-	-	-	
1093	-	-	-	-	-	-	-	-	

A sample of scheduling information is represented in Table 4. Focus on S-
box S_1 (the corresponding output bits are indexed $1, 2, 3, 4$). The table confirms
that S_1 is computed from clock cycle 1076 to 1091. Similarly S_2 is computed
between clock cycle 1140 and 1160 (as indicated in Table 5). A surprising effect
appears: at the moment S_2 is computed, the output bits of S_1 are manipulated
again. This phenomenon occurs especially between clock cycles 1154 and 1160.
We observe a similar behaviour for other pairs of S-box, (S_3, S_4), (S_5, S_6) and
(S_7, S_8).

Table 5. Scheduling information of the S-box outputs.

Cycle	First Round Output							
	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8 ...
1138	-	-	-	-	-	-	-	-
1139	-	-	-	-	-	-	-	-
1140	Y	-	Y	Y	Y	Y	Y	Y
1141	-	-	-	-	Y	Y	-	Y
1142	-	-	-	-	Y	Y	Y	Y
1143	-	-	-	-	-	-	-	-
1144	-	-	-	-	-	Y	-	-
1145	-	-	-	-	Y	Y	Y	Y
1146	-	-	-	-	Y	Y	Y	Y
1147	-	-	-	-	Y	Y	Y	Y
1148	-	-	-	-	Y	Y	Y	Y
1149	-	-	-	-	-	Y	-	-
1150	-	-	-	-	-	Y	-	-
1151	-	-	-	-	Y	Y	-	Y
1152	-	-	-	-	Y	Y	Y	Y
1153	-	-	-	-	Y	Y	Y	Y
1154	Y	Y	Y	Y	Y	Y	Y	Y
1155	Y	Y	Y	Y	Y	Y	Y	Y
1156	Y	Y	Y	Y	Y	Y	Y	Y
1157	Y	Y	Y	Y	Y	Y	Y	Y
1158	Y	Y	Y	Y	Y	Y	Y	Y
1159	Y	Y	Y	Y	Y	Y	Y	Y
1160	Y	Y	Y	Y	Y	Y	Y	Y
1161	-	-	-	-	-	-	-	-
1162	-	-	-	-	-	-	-	-

Our interpretation is that **the outputs of adjacent S-boxes are stored in the same register**. Therefore we are probably dealing with a 8-bit architecture and the implementation choice is to store the output of adjacent S-boxes in the same register.

4.4   Application to the Permutation Table

The last step of DES round function is a permutation of the 32 output bits. This permutation is also described by a table in the DES specification [10] (see Table 6).

The permutation of round 1 is computed just after the S-box layer. So we focus on this time interval and on the manipulation of S-box output bits (*i.e.* the input bits of the permutation). However the corresponding scheduling information is rather long. So we provide only a summary in Table 7. We focus on 8 among the 32 target bits which are manipulated first, and an interval is given where each of them is manipulated.

These 8 bits actually correspond to two lines (the 5-th and the 6-th) of the permutation table. Our interpretation is that the permutation is performed

Table 6. The DES Permutation Table.

16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

Table 7. Summary of scheduling information for the permutation.

Output Bit number	Significant interval
2	$1814 - 1817$
3	$1886 - 1890$
8	$1826 - 1832$
9	$1888 - 1892$
14	$1850 - 1856$
24	$1838 - 1841$
27	$1874 - 1877$
32	$1862 - 1865$
...	...

in a byte-oriented way, *i.e.* the bits corresponding to two consecutive lines are extracted and stored in the same register. Moreover look at the order in which the bits of Table 7 are manipulated: 2 then 8, 24, 14, 32, 27, 3 and finally 9. This is exactly the order of the bits in the two corresponding lines of the DES permutation table. Our interpretation is that the corresponding byte is computed from the most to the least significant bit.

To summarize, SCARE can retrieve the content of a permutation table, as illustrated with the example of DES.

4.5   Application to the Key Scheduling

Inbetween rounds 1 and 2, there is a long time interval where the output bits of the first round function are apparently manipulated. This effect is surprising because these bits are not needed at this point. We think that **this time interval corresponds to the key scheduling** and that **power consumption is correlated with S-box output bits because the registers where they were previously stored are overridden by the round key**. Indeed it is well known that the power consumption is correlated with the new value written in a register, but also with its previous content.

In Table 8, we represent a summary of the scheduling information. The byte formed with bits $9, \ldots, 16$ is manipulated starting from clock cycle 2698. Then its bits successively disappear from the scheduling information, *i.e.* they stop being manipulated. We believe it is because they are successively overridden

Table 8. Sample of scheduling information for the key scheduling.

Output Bit number	Significant interval
1	$2718 - 2985$
2	$2718 - 2985$
3	$2718 - 2837$
4	$2718 - 2796$
5	$2718 - 2873$
6	$2718 - 2724$
7	$2718 - 2985$
8	$2718 - 2941$
9	$2698 - 2952$
10	$2698 - 2952$
11	$2698 - 2737$
12	$2698 - 2776$
13	$2698 - 2860$
14	$2698 - 2952$
15	$2698 - 2709$
16	$2698 - 2904$
. . .	. . .

by round key bits. Let us look at the order in which this phenomenon occurs. This order is actually related to the permutated choice table PC-2 of DES (see Table 9). This table is used to extract the bits forming the round key.

Table 9. The DES Permutated Choice PC-2 Table.

14	17	11	24	1	5
3	28	16	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

In Table 8, the first bit to be overridden (in our target byte) is the bit 15, then 11, 12, 13, 16, 14, etc . . .. This should be put into correspondence with the 3-rd line of PC-2 (see Table 10). The order in which the bits disappear corresponds to the order of extraction, assuming that high indexes are extracted first.

A similar behaviour is observed with 3 other lines from PC-2. Therefore we learn 4 lines of PC-2 by just looking at the order in which the previous data is overridden. Of course, this could be used to retrieve a secret extraction table. Moreover we learn that the round key bits are stored in registers where intermediate data were previously stored. This is an undesirable property and could be used in other power attacks.

Table 10. Correspondence between PC-2 and the Output bits.

Output bits	9	10	11	12	13	14	15	16
Order to disappear	-	-	2	3	4	6	1	5
3-rd line of PC2	-	-	23	19	12	4	26	8
Order of extraction	-	-	2	3	4	6	1	5

4.6   Application for Side Channel Attacks

All the information derived in previous sections allows the reverse-engineer to know precisely when each instruction is executed on the device. For a power attack, this is useful information. Indeed the data analysis can be restricted to significant portions of the traces. For instance, to apply DPA on the S-box S_1 of the first round, we can isolate a small significant time interval for the analysis. This has two advantages

- The size of the data to handle is smaller, which improves the speed of the analysis
- Meaningless portions of the curves are not taken into account, which reduces the underlying noise. So the probability of success of DPA improves, especially with a limited number of traces

As an example, from the previous analysis, we can deduce how DPA should be applied on this device. Since S-box S_2 outputs are combined with S-box S_1's outputs, basic DPA on S_2's outputs is likely to give noisy results. It should be more efficient to apply DPA to the S-box S_1 or to the permutation. This information may also be helpful to mount more sophisticated attacks, like the recently proposed Davies-Murphy Power Attack [8]. In this last case, we know that only the pairs of S-box (S_1, S_2), (S_3, S_4), (S_5, S_6) or (S_7, S_8) can be valid targets.

5   Conclusion

SCARE (Side Channel Analysis for Reverse Engineering) is a new field of application of Side Channel Attacks (SCA), where one tries to reverse-engineer an implementation, based only on the power consumption of the cryptographic device. We proposed new methods for SCARE and applied them to the famous block cipher DES.

    We managed to retrieve experimentally many information about a target implementation, including constant tables used by the cipher, details of the architecture, registers where data are stored or the order of execution of the instructions. These results against DES suggest that it would be difficult to protect the secrecy of a proprietary algorithm regarding SCARE. We also consider this analysis to be an interesting preliminary step before applying power attacks to the device.

References

1. E. Biham and A. Shamir. Differential Fault Analysis of Secret Key Cryptosystems. In B. Kaliski, editor, *Advances in Cryptology – Crypto'97*, volume 1294 of *Lectures Notes in Computer Science*, pages 513–525. Springer, 1997.
2. D. Boneh, R. DeMillo, and R. Lipton. On the Importance of Checking Cryptographic Protocols for Faults (Extended Abstract). In W. Fumy, editor, *Advances in Cryptology – Eurocrypt'97*, volume 1233 of *Lectures Notes in Computer Science*, pages 37–51. Springer, 1997.
3. V. Carlier, H. Chabanne, E. Dottax, and H. Pelletier. Electromagnetic Side Channels of an FPGA Implementation of AES. Cryptology ePrint Archive, Report 2004/145, 2004. `http://eprint.iacr.org/`.
4. Christophe Clavier. Side Channel Analysis for Reverse Engineering (SCARE) - An Improved Attack Against a Secret A3/A8 GSM Algorithm. Cryptology ePrint Archive, Report 2004/049, 2004. `http://eprint.iacr.org/`.
5. K. Gandolfi, C. Mourtel, and F. Olivier. Electromagnetic Analysis: Concret Results. In Ç. Koç, D. Naccache, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems (CHES) – 2001*, volume 2162 of *Lectures Notes in Computer Science*, pages 251–261. Springer, 2001.
6. P. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Others Systems. In N. Koblitz, editor, *Advances in Cryptology – Crypto'96*, volume 1109 of *Lectures Notes in Computer Science*, pages 104–113. Springer, 1996.
7. P. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In M. Wiener, editor, *Advances in Cryptology – Crypto'99*, volume 1666 of *Lectures Notes in Computer Science*, pages 388–397. Springer, 1999.
8. S. Kunz-Jacques, F. Muller, and F. Valette. The Davies-Murphy Power Attack. In P.-J. Lee, editor, *Advances in Cryptology – Asiacrypt'04*, volume 3329 of *Lectures Notes in Computer Science*, pages 451–467. Springer, 2004.
9. R. Novak. Side-Channel Attacks on Substitution Blocks. In J. Zhou, M. Yung, and Y. Han, editors, *Applied Cryptography and Network Security (ACNS) 2003*, volume 2846 of *Lectures Notes in Computer Science*, pages 307–318. Springer, 2003.
10. National Institute of Standards and Technology (NIST). Data Encryption Standard (DES) FIPS Publication 46-3, October 1999. Available at `http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf`.

A   A Larger Sample of Experimental Results

The sample represented in Figure 1 illustrates the scheduling information we obtain in practice. It corresponds to the 32 input bits of the first round (*i.e.* 32 bits from the plaintext), and the corresponding information between clock cycles 1074 and 1116. We used coloured thresholds as a visual tool. Red corresponds to high values of the indicator V_i, and green to intermediate values. The content of Table 1 is actually extracted from this larger sample.

Large values in the tables correspond to clock cycles where plaintext bits are manipulated. One can observe that there are isolated signals probably corresponding to noise in the experiments. However some very significant portions clearly appear. For instance, bits number $1, 2, 3, 4, 5$ and 32 are clearly manipulated between clock cycles 1076 and 1081. This is likely to correspond to the computation of S_1.

Fig. 1. A large sample of experimental results.