A Differential Fault Attack Technique against SPN Structures, with Application to the AES and KHAZAD

Gilles Piret and Jean-Jacques Quisquater

UCL Crypto Group,
Laboratoire de Microélectronique, Université Catholique de Louvain,
Place du Levant, 3, B-1348 Louvain-la-Neuve, Belgium
{piret, jjq}@dice.ucl.ac.be

Abstract. In this paper we describe a differential fault attack technique working against Substitution-Permutation Networks, and requiring very few faulty ciphertexts. The fault model used is realistic, as we consider random faults affecting bytes (faults affecting one only bit are much harder to induce). We implemented our attack on a PC for both the AES and KHAZAD. We are able to break the AES-128 with only 2 faulty ciphertexts, assuming the fault occurs between the antepenultimate and the penultimate `MixColumn`; this is better than the previous fault attacks against AES[6,10,11]. Under similar hypothesis, KHAZAD is breakable with 3 faulty ciphertexts.

Keywords: AES, Block Ciphers, Fault Attacks, Side-channel Attacks

1 Introduction

The idea of using hardware faults happening during the execution of a cryptographic algorithm for breaking it (typically, for retrieving the key) was first suggested in 1997 by D. Boneh, R.A. DeMillo, and R.J. Lipton [7,8]. They succeeded in breaking an RSA CRT with both a correct and a faulty signature of the same message. Shortly after, an adaptation of this idea on block ciphers was proposed by E. Biham and A. Shamir[5].

Application of this principle to tamper resistant devices such as smart cards is a real threat (see e.g. [1,2]): by changing the power supply voltage or the frequency of the external clock, or by applying radiations, a fault can be induced with some probability during the computation. The faults induced by most of these techniques affect one byte[1], as it is the size of a register for current smart cards; however it is often the case that the attacker cannot locate a priori at which stage of the algorithm the fault occurred.

Several authors mounted differential fault attacks against the AES[6,10,11]. In this paper we present a fault attack working against any block cipher with

[1] Although progresses were recently made in inducing faults affecting only one bit[13].

a Substitution-Permutation structure[2]. More precisely, its round function must have the form $\sigma[K^r] \circ \theta_r \circ \gamma_r$ (r is the round number), where:

- The γ_r layer consists in the parallel application of n 8×8 S-boxes (not necessarily identical).
- $\sigma[k]$ denotes the key addition layer:

$$\sigma[k](a) = b \Leftrightarrow b_j = a_j \oplus k_j, 1 \leq j \leq n$$

  where $\oplus$ denotes a group operation. As it is often exclusive or, in the following we will only deal with this case. But our attack could also work against other group operations.
- The diffusion layer θ_r is a mapping that is linear with respect to $\oplus$.
- K^r denotes the r^{th} round key.

We denote the block size by $N_b = 8n$. Note that the fact that the S-boxes are 8×8 is absolutely not mandatory for our attack; we restricted to this parameter as it is common to choose such a size, well fitted with implementation considerations. 4×4 and 2×2 S-boxes can be viewed as 8×8 S-boxes as well, by considering groups of 2 (resp. 4) of them.

The last round of the cipher has the special form $\sigma[K^R] \circ \gamma_R$, as a θ layer at this stage would have no cryptographic significance. Moreover, the first round is preceded by a key addition layer. Thus the whole cipher can be described as:

$$\sigma[K^R] \circ \gamma_R \circ (\bigcirc_{r=1}^{R-1} \sigma[K^r] \circ \theta_r \circ \gamma_r) \circ \sigma[K^0]$$

Remark that the γ_r and θ_r layers need not be identical for all rounds. Only the last two rounds are important for our attack. They are depicted in Fig. 1.

2 The Attack

The Fault Model. We are dealing with *random faults*, in the sense that the faulty value is random and is assumed to be uniformly distributed. They are assumed to occur on one byte. Moreover we assume that the fault occurred somewhere between the before-last layer θ_{R-2} and the last layer θ_{R-1} (i.e. somewhere inside the frame of Fig. 1). Under this condition, the exact stage of the computation at which the fault occurred has no importance, and cannot be guessed by observing the ciphertexts either.
In the remaining of this paper, by $(C; C^*)$ we always denote a right ciphertext C and its corresponding faulty ciphertext C^*. Also, unless otherwise stated, indices will refer to byte positions (for example, C_1 denotes the left-most byte of C).

[2] Strictly speaking, the designation *substitution-permutation network* implies that the diffusion layer is a bit permutation. However, it becomes more and more used to refer also to ciphers with a more complex diffusion layer. So do we.

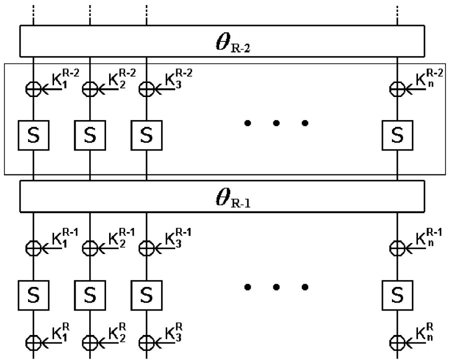

Fig. 1. Last 2,5 rounds of a Substitution-Permutation Network.

Basic Attack. Consider thus 1-byte differences at the input of the linear layer θ_{R-1}. We count $255n$ possible such differences (n different possible locations, and 255 different possible values). Because of the linearity, the number of corresponding possible differences at its output is also $255n$; but while the input difference affected one byte only, the output difference affects several ones, because of the diffusion (if the diffusion layer is optimal, all bytes are affected; with some slower diffusions only a few of them may be affected). Note that the key addition $\sigma[K^{R-1}]$ does not change the set of possible differences.

These considerations lead to a first sketch of attack. For simplicity we assume the θ_{R-1} layer achieves optimal diffusion.

1. Compute the $255n$ possible differences at the output of θ_{R-1}, i.e. the $255n$ values $\theta_{R-1}(x)$, where x has a byte hamming weight of 1. Store them in a list $\mathcal{D}$.
2. Consider a plaintext P, C its corresponding ciphertext, and C^* the faulty ciphertext.
3. Take a guess on round key K^R.
4. Compute the difference $\gamma_R^{-1} \circ \sigma[K^R](C) \oplus \gamma_R^{-1} \circ \sigma[K^R](C^*)$. Check whether it is in $\mathcal{D}$. If yes, add the round key to the list $\mathcal{L}$ of possible candidates.
5. Consider a new plaintext P (with corresponding C and C^*) and go back to step 2 (this time round key guesses only go through the list $\mathcal{L}$ of possible candidates; if the difference computed at step 4 is not in $\mathcal{D}$, remove the candidate from $\mathcal{L}$). Repeat until there remains only one candidate in $\mathcal{L}$.

If the diffusion layer is slow, only a limited number of bytes of the cipher are affected by a given fault. Thus each pair $(C; C^*)$ gives information only on a subset of the round key bytes; the guess is made only on theses bytes. The AES is a good example of this fact.

After the last round key has been found, and if it is not sufficient to retrieve the whole key, the last round is peeled off, and the attack is repeated on the reduced cipher.

Complexity Analysis. We compute the fraction of the round keys K^R that are suggested by a single pair $(C; C^*)$ with difference $\Delta = C \oplus C^*$. Suppose the number of possible differences Δ' before γ_R is 255^n. Among these, $255n$ are elements of $\mathcal{D}$. Thus the fraction of the keys surviving the test is $255n/255^n = n \cdot 255^{1-n}$.

However this computation does not take into account the fact that the number of differences Δ' before γ_R that can cause difference Δ after it is far less than 255^n; this is due to the fact that the XOR distribution table[3] of each S-box contains a lot of 0's. Thus we made the hypothesis that for the observed Δ, the fraction of elements of $\mathcal{D}$ among the corresponding possible Δ' is also about $255n/255^n$.

We conclude that the number of remaining wrong candidates for K^R after N $(C; C^*)$ pairs have been treated is about $256^n(n \cdot 255^{1-n})^N$. The conclusion (for all practical values of n) is that one pair $(C; C^*)$ is not enough to retrieve K^R, but two are (still under the hypothesis that the diffusion layer is optimal; see the AES case in section 3 for an example where it is not).

A Practical Attack. As it is presented, this attack is not really practical, as it implies a guess on the last round key, that is to say a complexity $\sim 2^{N_b}$. We show that slightly modifying the attack considerably reduces this complexity. Once again, for simplicity we assume the diffusion layer considered is optimal. A similar technique, applied only to the bytes affected by the fault, can be used when it is not.

1. Compute the $255n$ possible differences at the output of θ_{R-1}. Store them in a list $\mathcal{D}$.
2. Consider 2 right ciphertext/faulty ciphertext pairs $(C; C^*)$ and $(D; D^*)$.
3. Consider the two left-most bytes of K^R:
   - For each of the 2^{16} candidates, compute[4]:
   $$\gamma_R^{-1} \circ \sigma[\langle K_1^R, K_2^R \rangle](\langle C_1, C_2 \rangle) \oplus \gamma_R^{-1} \circ \sigma[\langle K_1^R, K_2^R \rangle](\langle C_1^*, C_2^* \rangle)$$
   and
   $$\gamma_R^{-1} \circ \sigma[\langle K_1^R, K_2^R \rangle](\langle D_1, D_2 \rangle) \oplus \gamma_R^{-1} \circ \sigma[\langle K_1^R, K_2^R \rangle](\langle D_1^*, D_2^* \rangle)$$
   - Compare the results with the two left-most bytes of the $255n$ differences in list $\mathcal{D}$. Make a list $\mathcal{L}$ of the $\langle K_1^R, K_2^R \rangle$ for which a match is found for both ciphertext pairs.
4. For each $K^\bullet \in \mathcal{L}$, try to extend it by one byte:
   - Remove $K^\bullet$ from $\mathcal{L}$.

[3] See [4] for definition of this concept.

[4] We commit a small abuse in notations by applying σ and γ_R to data of improper length. The right way to understand this is to think that e.g. $\langle C_1^*, C_2^* \rangle$ has been right-appended with 0's, and that only the 2 left-most bytes of the output are considered.

   – For all 2^8 K_3^R, compute:

$$\gamma_R^{-1} \circ \sigma[\langle K_2^\bullet, K_3^R \rangle](\langle C_2, C_3 \rangle) \oplus \gamma_R^{-1} \circ \sigma[\langle K_2^\bullet, K_3^R \rangle](\langle C_2^*, C_3^* \rangle)$$

   and

$$\gamma_R^{-1} \circ \sigma[\langle K_2^\bullet, K_3^R \rangle](\langle D_2, D_3 \rangle) \oplus \gamma_R^{-1} \circ \sigma[\langle K_2^\bullet, K_3^R \rangle](\langle D_2^*, D_3^* \rangle)$$

   – Compare the differences obtained with bytes 2 and 3 of the $255n$ differences in $\mathcal{D}$. If a match is found (again for both ciphertext pairs), add the newly extended key $\langle K^\bullet, K_3^R \rangle$ to $\mathcal{L}$.

5. Repeat step 4 until elements of $\mathcal{L}$ have a length of n bytes.
6. Apply now the first algorithm we gave using the same pairs $(C; C^*)$ and $(D; D^*)$, but consider only the candidates K^R in $\mathcal{L}$ (their number is much smaller than 2^{N_b}).

The idea of this algorithm is that its first 5 steps compute a set of candidates of which the candidates selected by the first algorithm are a subset; otherwise stated, every candidate obtained by applying the first algorithm to pairs $(C; C^*)$ and $(D; D^*)$ will be returned by steps 1→5 of the second algorithm too, but the converse is not true. Thus, the first 5 steps of the second algorithm (that have a low complexity) perform a "first sorting" of the candidates. After that, the size of the set of candidates is quite small, and is affordable for the first algorithm.

Faults Occurring at a Wrong Location. As the attacker usually has no control on the fault location, it is important to be able to distinguish pairs $(C; C^*)$ resulting from faults occurring between θ_{R-2} and θ_{R-1} (we call such pairs *right pairs*) from other pairs (these are called *bad pairs*). It is trivial in the case of diffusion layers for which a 1-byte difference in the input implies an output difference affecting only some bytes of the output: in this case it is enough to observe whether some bytes are identical in both C and C^*.

But in the case of optimal diffusion layers, it is not possible to decide whether one only pair $(C; C^*)$ is a right or a bad one. However applying our attack to 2 pairs $(C; C^*)$ one of which is bad will very probably result in no solution for the key K^R. Thus we can indeed distinguish bad pairs $(C; C^*)$ from right ones, but only by considering *pairs* of ciphertext pairs $(C; C^*)$. Nevertheless the attack should be practical: if we consider that 1 ciphertext pair out of 100 is right, which is more than reasonable, we have 10000 pairs to examine before finding two right pairs, which is still accessible.

3 Application to the AES

3.1 Overview of the AES Structure

The AES[9] is an example of a substitution-permutation structure, as it is defined in the introduction. Both its key and block size can be 128, 192, or 256 bits. In

Table 1. The State during AES encryption

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$

this paper we will only deal with the 128-bit block, 128-bit key variant, as it is the most widely used. Our attack can be extended trivially to other variants.

The key addition is performed using exclusive or. The γ layer (identical for all rounds) is made up of the application of 16 identical 8×8 S-boxes. The intermediate computation result, called *state* is usually represented by a 4×4 square, each cell of which is a byte (see Table 1); the θ layer (identical for all rounds) is the composition of two transformations of the state:

1. First, the `ShiftRow` transformation consists in shifting cyclically the rows of the state. Row 0 is not shifted, row 1 is shifted by 1 byte, row 2 is shifted by 2 bytes, and row 3 by 3 bytes. It is pictured in Table 2.
2. Then, the `MixColumn` transformation applies a linear transformation with optimal byte branch number(i.e. 5) to each column of the state. More precisely, application of `MixColumn` to the first column of the state (for example) is computed by:

$$\begin{bmatrix} b_{0,0} \\ b_{1,0} \\ b_{2,0} \\ b_{3,0} \end{bmatrix} = \begin{bmatrix} 02\ 03\ 01\ 01 \\ 01\ 02\ 03\ 01 \\ 01\ 01\ 02\ 03 \\ 03\ 01\ 01\ 02 \end{bmatrix} \cdot \begin{bmatrix} a_{0,0} \\ a_{1,0} \\ a_{2,0} \\ a_{3,0} \end{bmatrix}$$

   where multiplication is performed in GF(2^8) (via definition of an irreducible polynomial of degree 8 over GF(2), see [9] for details).

Table 2. The `ShiftRow` transformation

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$

$\xrightarrow{\texttt{ShiftRow}}$

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,0}$
$a_{2,2}$	$a_{2,3}$	$a_{2,0}$	$a_{2,1}$
$a_{3,3}$	$a_{3,0}$	$a_{3,1}$	$a_{3,2}$

We observe that a 1-byte difference in the state before the θ layer results in a 4-byte difference after it. This property is important for our attack.
Note also that the last round has no `MixColumn`, but well a `ShiftRow`. The reason behind it is purely implementation related. This last `ShiftRow` has no cryptographic significance, and is not relevant to our attack either.

3.2 Previous Works about Fault Analysis on the AES

Several papers have been written on the subject. We summarize here their contributions, by chronological order:

The first paper we know of is the one of Blömer and Seifert[6]. Mainly, two attacks are presented:

- The first one uses a very restrictive fault model: namely, it assumes that one can *force to 0* the value of a *chosen* bit. It is worth noting that applying this technique to the memory cells storing the key makes it trivial to retrieve, even without being able to choose precisely the location of the bit set to 0. This has been demonstrated by Biham and Shamir[5], and is true for any algorithm. [6] shows however that the fault model can be slightly relaxed. 128 faulty encryptions of plaintext **0** are required to retrieve the key using this technique.
- The second attack is implementation-dependent, and has several variants depending on the implementation. Its principle is to turn the timing attack on AES suggested by Koeune and Quisquater[12] into a fault based cryptanalysis. The fault model used also depends on the implementation. The authors claim that about 16 faulty ciphertexts (with the fault occurring at a carefully chosen location) are needed to retrieve one key byte.

In [11], Giraud presents two fault attacks on the AES. Both require the ability to obtain several faulty ciphertexts originating from the *same* plaintext (contrary to our attack):

- The first one assume it is possible to induce a fault on only one bit of an intermediate result. More precisely, it exploits faults induced on one bit before the last γ layer (while we exploit faults occurring before the last diffusion layer). Under these conditions, about 50 faulty ciphertexts are necessary to retrieve the full key (provided the location of the fault can be chosen).
- The second attack, more realistic, exploits faults on bytes. It requires the ability of inducing faults at several chosen places, including the key schedule. The author claims that 250 faulty ciphertexts are needed (it is assumed that the attacker can choose the stage of the computation where the fault takes place, but not the exact byte), and that the time needed for computation is about 5 days.

Finally, P. Dusart, G. Letourneux, and O. Vivolo[10] take advantage of byte faults occurring after the `ShiftRow` layer of the 9^{th} round. Thus the fault model and the hypothesis on the fault location are exactly the same as in our attack. However the way they exploit faults is different from ours: they use the particular form of the `Mixcolumn` transformation and of the AES S-box to write and solve a system of equations (one by S-box) of which the unknown value is the one of the fault (i.e. of the byte difference engendered by the fault). Suggestions for 4 key bytes follow. The authors show that 5 well-located faults are necessary to retrieve 4 key bytes.

3.3 Our Results

As observed above, a 1-byte difference at the input of the θ layer of AES results in a 4-byte difference at its output. Concretely, it means that a fault on one byte before the θ_{R-1} layer will give information on only 4 bytes of the last round key (the other bytes of both ciphertexts C and C^* being identical). More precisely, with the different bytes of the state numbered as in Table 1:

- A fault on byte $a_{0,0}$, $a_{1,1}$, $a_{2,2}$, or $a_{3,3}$ will release information about round key bytes $K^R_{0,0}$, $K^R_{1,3}$, $K^R_{2,2}$, $K^R_{3,1}$.
- A fault on byte $a_{0,1}$, $a_{1,2}$, $a_{2,3}$, or $a_{3,0}$ will release information about round key bytes $K^R_{0,1}$, $K^R_{1,0}$, $K^R_{2,3}$, $K^R_{3,2}$.
- A fault on byte $a_{0,2}$, $a_{1,3}$, $a_{2,0}$, or $a_{3,1}$ will release information about round key bytes $K^R_{0,2}$, $K^R_{1,1}$, $K^R_{2,0}$, $K^R_{3,3}$.
- A fault on byte $a_{0,3}$, $a_{1,0}$, $a_{2,1}$, or $a_{3,2}$ will release information about round key bytes $K^R_{0,3}$, $K^R_{1,2}$, $K^R_{2,1}$, $K^R_{3,0}$.

Consider a fault occurring on one of the bytes $a_{0,0}$, $a_{1,1}$, $a_{2,2}$, or $a_{3,3}$. We compute that with one pair $(C; C^*)$ about 1036 candidates for $(K^R_{0,0}, K^R_{1,3}, K^R_{2,2}, K^R_{3,1})$ remain (see complexity analysis in section 2). If two pairs are exploited, we are in principle left with the right candidate only. Thus with 8 faults at carefully chosen locations, we are able to recover the whole key.

However it is possible to do better. Suppose a fault occurs on one byte somewhere between θ_{R-3} and θ_{R-2} (rather than between θ_{R-2} and θ_{R-1}). The corresponding difference after the θ_{R-2} layer has 4 non-zero bytes. Each of them can be exploited as described previously, and releases information about a different part of the last round key. For example, a fault on $a_{0,0}$ before θ_{R-2} will result in a non-zero difference on $a_{0,0}$, $a_{1,0}$, $a_{2,0}$, and $a_{3,0}$ after it. Thus using faults occurring somewhere between θ_{R-3} and θ_{R-2} allows us to kill 4 birds with one stone. As a consequence, only 2 such faults are needed to retrieve the whole AES-128 key.

We implemented our attack on a PC. The results obtained well matched our estimates.
When one fault between θ_{R-2} and θ_{R-1} was considered, the average number of candidates for 4 bytes of K^R obtained was 1046 (instead of the expected 1036). A more surprising point (a priori) was that 2 pairs $(C; C^*)$, both giving information on the same 4 bytes of K^R, allowed to retrieve a unique value for these bytes in only 98% of the cases; otherwise two possible values for these 4 bytes remained (or even four, but it was very rare). These deviations from the expected results are due to the fact that we were making very few hypothesis on the θ layer and the S-boxes in our complexity analysis. Thus our estimations did not take into account particular features of these components. We give a more detailed explanation for the 98% figure in appendix A.
Using 2 faults between θ_{R-3} and θ_{R-2}, the number of candidates left for the whole key never exceeded 16, and we obtained one only candidate in 77% of

the cases. The time needed to complete the attack is a few seconds. Also, when applying the attack to 2 ciphertext pairs one of which is bad (i.e. corresponds to a fault occurring before θ_{R-3}), the set of candidates returned by our algorithm was always empty.

4 Application to KHAZAD

4.1 Brief Description of KHAZAD

KHAZAD is a 64-bit block 128-bit key block cipher submitted to the NESSIE European project by P.S.L.M. Barreto and V. Rijmen[3]. It has 8 rounds, whose structure is the one described in section 1, with exclusive or used for key addition. Its γ layer (identical for all rounds) is made up of the application of 8 identical involutive 8×8 S-boxes. Its θ layer (also identical for all rounds) has optimal byte branch number (i.e. 9) and is also involutive.

4.2 Our Attack Applied to KHAZAD

Two faults occurring between θ_{R-1} and θ_{R-2} are enough to retrieve K^R (as each fault gives information on all bytes of K^R; remember that θ is optimal). However knowledge of K^R is not enough to retrieve the whole key. Thus once K^R is known the last round is peeled off. Then a fault occurring between θ_{R-2} and θ_{R-3} is exploited to select about $256^8 \cdot (8 \cdot 255^{-7}) \simeq 2105$ candidates for K^{R-1}. We conclude the attack by searching exhaustively among these candidates; knowledge of K^R and K^{R-1} allows to compute the main key.

Our implementation of the attack showed that using 2 right pairs $(C; C^*)$ we obtain one unique candidate for K^R in about 90% of the cases (otherwise 2 candidates remain, sometimes 4). One reason for this bad score happens to be related to the choice of the S-boxes: it seems that the worse an S-box is with respect to differential cryptanalysis, the better it resists our fault attack. As an illustration, we applied our attack to a modified version of KHAZAD using the AES S-boxes; then a unique candidate is obtained from 2 right pairs $(C; C^*)$ with probability 96%. Appendix A sketches an explanation for this.
Note that once again, the number of faulty ciphertexts needed to retrieve the key is not affected by these figures; only the time complexity of the attack (which remains small anyway) is. Also, when trying to recover K^R with 2 ciphertext pairs one of which is bad, the set of candidates returned by our algorithm was always empty.

5 Conclusion

The basic idea of our attack is to use the diffusion property of the last θ layer, in order to determine whether the difference before the last nonlinear layer γ

possibly originates in a fault or not. This provides us with a distinguishing criteria for the last round key. The fault model used is the most liberal and realistic one: we simply need random faults occurring on bytes. The ability to choose the location of the fault is not important either: of course only faults occurring at a given location (between θ_{R-2} and θ_{R-1} in the general case, between θ_{R-3} and θ_{R-1} in the case of AES) are exploitable, but those occurring elsewhere can be discarded.

We give in Table 3 a summary of existing faults attacks against the AES.

Table 3. Comparison of existing fault attacks against the AES

Ref.	Fault Model	Fault Location	# Faulty Encryptions
[6]	Force 1 bit to 0	Chosen	128
[6]	Fct of impl.	Chosen	256
[11]	Switch 1 bit	Any bit of chosen bytes	~ 50
[11]	Disturb 1 byte	Anywhere among 4 bytes	~ 250
[10]	Disturb 1 byte	Anywhere between θ_{R-2} and θ_{R-1}	~ 40
This paper	**Disturb 1 byte**	**Anywhere between $\theta_{\mathbf{R-3}}$ and $\theta_{\mathbf{R-2}}$**	**2**

Amongst these attacks, the most similar to ours is the one of P. Dusart & *al.*[10]. The difference mainly lies in the way faults are exploited. [10] exploits the particular structure of the AES S-box and `MixColumn`, while we do not. The consequence is that their attack is not adaptable to other algorithms; ours can be used to attack KHAZAD(as we showed in section 4), but also ciphers like Serpent or Anubis[5]. On the other hand, note that an algorithm such as Safer++ is not directly vulnerable to our attack, due to the use of two different group operations for key mixing.

In our attack against AES, note that while 2 well-located faults are needed for easy retrieving of the key, one only well-located fault reduces the size of the key space to be explored to $1046^4 \simeq 2^{40}$.

As a final remark, it is amusing to note that it is the very simple and elegant structure of SPN structures that makes our attack so efficient... It is not clear whether ciphers with a more intricate structure could be broken with so few ciphertext pairs.

References

1. R. Anderson and M. Kuhn. Tamper resistance – a cautionary note. In *Proc. of the second USENIX workshop on electronic commerce*, pages 1–11, Oakland, California, Nov. 18-21 1996.
2. R. Anderson and M. Kuhn. Low cost attacks on tamper resistant devices. In *Proc. of 1997 Security Protocols Workshop*, volume 1361 of *Lecture Notes in Computer Science*, pages 125–136. Springer, 1997.

[5] But this last must be rewritten in order to comply with our description of an SPN structure.

3. P.S.L.M. Barreto and V. Rijmen. The Khazad Legacy-Level Block Cipher. Available at http://www.cryptonessie.org.
4. E. Biham and A. Shamir. Differential cryptanalysis of DES-like cryptosystems. *Journal of Cryptology*, 4(1):3–72, 1991.
5. E. Biham and A. Shamir. Differential Fault Analysis of Secret Key Cryptosystems. In B. Kaliski, editor, *Advances in Cryptology - CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 513–525. Springer, 1997.
6. J. Blömer and J.-P. Seifert. Fault based cryptanalysis of the Advanced Encryption Standard. To appear in *Financial Cryptography '03*, LNCS. Springer, 2003. Also available at http://eprint.iacr.org/, 2002/075.
7. D. Boneh, R. A. DeMillo, and R. J. Lipton. On the Importance of Checking Cryptographic Protocols for Faults (Extended Abstract). In W. Fumy, editor, *Advances in Cryptology - EUROCRYPT '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 37–51. Springer, 1997.
8. D. Boneh, R. A. DeMillo, and R. J. Lipton. On the Importance of Eliminating Errors in Cryptographic Computations. In *Journal of Cryptology 14(2)*, pages 101–120, 2001.
9. J. Daemen and V. Rijmen. AES proposal: Rijndael. In *Proc. first AES conference*, August 1998. Available on-line from the official AES page: `http://csrc.nist.gov/CryptoToolkit/aes/rijndael/Rijndael.pdf`.
10. P. Dusart, G. Letourneux, and O. Vivolo. Differential Fault Analysis on A.E.S. Available at http://eprint.iacr.org/, 2003/010.
11. C. Giraud. DFA on AES. Available at http://eprint.iacr.org/, 2003/008.
12. F. Koeune and J.-J. Quisquater. A timing attack against Rijndael. Technical report, available at http://www.dice.ucl.ac.be/crypto/techreports.html, 1999.
13. S. Skorobogatov and R. Anderson. Optical fault induction attacks. In Burton S. Kaliski, Çetin K. Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002*, volume 2523 of *Lecture Notes in Computer Science.* Springer, 2002.

A Deeper Analysis of the AES Case

In this appendix we analyze why 2 right pairs $(C;C^*)$ and $(D;D^*)$, both releasing information on the same 4 bytes of K^R, do not allow to compute an unique value for these 4 bytes in about 2% of the cases.

Let $K^R_\bullet := \langle K^R_{0,0}, K^R_{1,3}, K^R_{2,2}, K^R_{3,1}\rangle$ denote 4 bytes of the last round key of an AES. Let $C_\bullet := \langle C_{0,0}, C_{1,3}, C_{2,2}, C_{3,1}\rangle$ and $C^*_\bullet := \langle C^*_{0,0}, C^*_{1,3}, C^*_{2,2}, C^*_{3,1}\rangle$ be a right ciphertext and its faulty counterpart, both limited to the same 4 bytes.
It is easy to see that applying our attack to pair $(C;C^*)$ will return, together with $K^R_\bullet$ and other candidates, $K^R_\bullet \oplus C_\bullet \oplus C^*_\bullet$ and the 14 other candidates obtained when only some bytes of $C_\bullet \oplus C^*_\bullet$ are XORed to $K^R_\bullet$.
Consider a second pair $(D;D^*)$ with $D_\bullet := \langle D_{0,0}, D_{1,3}, D_{2,2}, D_{3,1}\rangle$ and $D^*_\bullet := \langle D^*_{0,0}, D^*_{1,3}, D^*_{2,2}, D^*_{3,1}\rangle$; $D^*_\bullet$ is the faulty counterpart of $D_\bullet$. Assume $D_\bullet \oplus D^*_\bullet$ share some bytes with $C_\bullet \oplus C^*_\bullet$; suppose for example $C_{0,0} \oplus C^*_{0,0} = D_{0,0} \oplus D^*_{0,0}$. Then, $K^R \oplus \langle C_{0,0} \oplus C^*_{0,0}, 0, 0, 0\rangle$ will be returned by our attack (applied to both $(C;C^*)$ and $(D;D^*)$) as well as K^R.

As the probability of having the same value at a given position of $C \oplus C^*$ and $D \oplus D^*$ is $1/255$, the probability that we observe the same value at at least one position is $1 - (254/255)^4 \simeq 0,015$. So we have found the main reason why more than one key is returned in 2% of the cases. Note that this phenomenon is not specific to AES; furthermore this explanation could be generalized by referring to the XOR distribution table[4] of the S-boxes. It appears then that paradoxically good S-boxes with respect to differential cryptanalysis are also those making our fault attack the most efficient...