EXTERNAL ENCODINGS DO NOT PREVENT TRANSIENT FAULT ANALYSIS

Christophe Clavier

Gemalto, Security Labs

CHES 2007 – Vienna - September 12, 2007

OUTLINE

1 INTRODUCTION

- Motivation
- Externally encoded DES
- Assumptions

2 DESCRIPTION OF THE ATTACK

- The principle
- The basic attack
- An improved version
- Classical counter-measures

3 CONCLUSION

- Lessons
- Open problems

OUTLINE

1. INTRODUCTION
   - Motivation
   - Externally encoded DES
   - Assumptions
2. DESCRIPTION OF THE ATTACK
   - The principle
   - The basic attack
   - An improved version
   - Classical counter-measures
3. CONCLUSION
   - Lessons
   - Open problems

THE QUESTION

Is it possible to reveal the secret key of an unknown algorithm by means of transient fault analysis?

THE QUESTION

Is it possible to reveal the secret key of an unknown algorithm by means of transient fault analysis?

Any known transient fault analysis on a cryptographic algorithm requires the knowledge of either the input or the output:

THE QUESTION

Is it possible to reveal the secret key of an unknown algorithm by means of transient fault analysis?

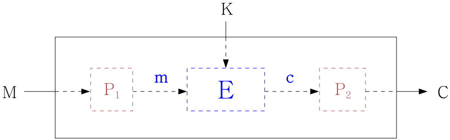
Any known transient fault analysis on a cryptographic algorithm requires the knowledge of either the input or the output:

- *Differential Fault Analysis*, DFA (Biham and Shamir, Crypto ’97)
  - Exploits paires $(c, \overset{\leadsto}{c})$ of normal and faulty ciphertexts
  - Requires the knowledge of the output
- *Collision Fault Analysis*, CFA (Hemme, Ches ’04)
  - Given a faulty cipertext $\overset{\leadsto}{c}$ corresponding to some input m, try to find another input $m^\star$ encrypting to the same output
  - Requires the knowledge (even the control) of the input

MOTIVATION

A POSSIBLE DESIGN

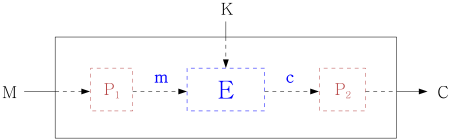
A way to design a secret (proprietary) block cipher could be to enclose a public and well studied ciphering function E (DES, AES, . . .) between two secret external encodings



MOTIVATION

A POSSIBLE DESIGN

A way to design a secret (proprietary) block cipher could be to enclose a public and well studied ciphering function E (DES, AES, ...) between two secret external encodings

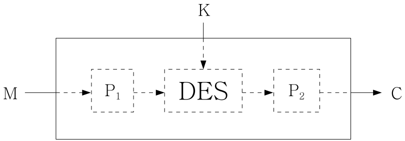


- P_1 and P_2 are two secret and deterministic one-to-one mappings.
- The design inherits its cryptographic strength from the core function E.
- Fault analysis should be prevented by the *obfuscation* layers P_1 and P_2 which conceal inputs m and outputs c of cipher E from the attacker.

THE CASE OF OBFUSCATED DES

THIS PAPER

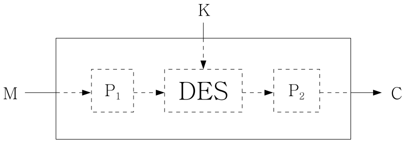
An externally encoded DES is not secure against transient fault analysis.



THE CASE OF OBFUSCATED DES

THIS PAPER

An externally encoded DES is not secure against transient fault analysis.



The hereafter described attack allows to recover the DES key without any knowledge about P_1 and P_2.

- Also applies to obfuscated Triple-DES
- Practically relevant if such constructions actually exist

ASSUMPTIONS FOR THE ATTACK TO WORK

FAULT MODEL

Faulting on a XOR instruction results in a fixed and known output (assumed to be 0 in the sequel) whatever the inputs are.

ASSUMPTIONS FOR THE ATTACK TO WORK

FAULT MODEL

Faulting on a XOR instruction results in a fixed and known output (assumed to be 0 in the sequel) whatever the inputs are.

ATTACKER MODEL

The attacker can choose the inputs, and knows the outputs.

(May be somewhat relaxed)

ASSUMPTIONS FOR THE ATTACK TO WORK

FAULT MODEL

Faulting on a XOR instruction results in a fixed and known output (assumed to be 0 in the sequel) whatever the inputs are.

ATTACKER MODEL

The attacker can choose the inputs, and knows the outputs.
(May be somewhat relaxed)

IMPLEMENTATION ASSUMPTIONS

The targeted device contains a

ASSUMPTIONS FOR THE ATTACK TO WORK

FAULT MODEL

Faulting on a XOR instruction results in a fixed and known output (assumed to be 0 in the sequel) whatever the inputs are.

ATTACKER MODEL

The attacker can choose the inputs, and knows the outputs.

(May be somewhat relaxed)

IMPLEMENTATION ASSUMPTIONS

The targeted device contains a

- software implementation of an externally encoded DES,

ASSUMPTIONS FOR THE ATTACK TO WORK

FAULT MODEL

Faulting on a XOR instruction results in a fixed and known output (assumed to be 0 in the sequel) whatever the inputs are.

ATTACKER MODEL

The attacker can choose the inputs, and knows the outputs.

(May be somewhat relaxed)

IMPLEMENTATION ASSUMPTIONS

The targeted device contains a

- software implementation of an externally encoded DES,
- on an 8-bit architecture,

ASSUMPTIONS FOR THE ATTACK TO WORK

FAULT MODEL

Faulting on a XOR instruction results in a fixed and known output (assumed to be 0 in the sequel) whatever the inputs are.

ATTACKER MODEL

The attacker can choose the inputs, and knows the outputs.

(May be somewhat relaxed)

IMPLEMENTATION ASSUMPTIONS

The targeted device contains a

- software implementation of an externally encoded DES,
- on an 8-bit architecture,
- with a natural (straightforward) implementation,

ASSUMPTIONS FOR THE ATTACK TO WORK

FAULT MODEL

Faulting on a XOR instruction results in a fixed and known output (assumed to be 0 in the sequel) whatever the inputs are.

ATTACKER MODEL

The attacker can choose the inputs, and knows the outputs.

(May be somewhat relaxed)

IMPLEMENTATION ASSUMPTIONS

The targeted device contains a

- software implementation of an externally encoded DES,
- on an 8-bit architecture,
- with a natural (straightforward) implementation,
- and without counter-measure (discussed later).

OUTLINE

1 INTRODUCTION
- Motivation
- Externally encoded DES
- Assumptions

2 DESCRIPTION OF THE ATTACK
- The principle
- The basic attack
- An improved version
- Classical counter-measures

3 CONCLUSION
- Lessons
- Open problems

THE ATTACK PRINCIPLE

FAULT INJECTION AS A PROBING TOOL

By comparing the outputs of two executions (one normal, one faulty) with same input, one infers whether the normal output of a faulted XOR is zero.

THE ATTACK PRINCIPLE

FAULT INJECTION AS A PROBING TOOL

By comparing the outputs of two executions (one normal, one faulty) with same input, one infers whether the normal output of a faulted XOR is zero.

Assuming that the fault is certainly injected on the targeted XOR, an identity of ciphertexts implies that the fault was *ineffective*.
This reveals a local intermediate value equal (more precisely, equivalent) to 0.

THE ATTACK PRINCIPLE

FAULT INJECTION AS A PROBING TOOL

By comparing the outputs of two executions (one normal, one faulty) with same input, one infers whether the normal output of a faulted XOR is zero.

Assuming that the fault is certainly injected on the targeted XOR, an identity of ciphertexts implies that the fault was *ineffective*.
This reveals a local intermediate value equal (more precisely, equivalent) to 0.

If, for the same message, faults on two related XOR instructions are both ineffective, then the normal XOR outputs are simultaneously equal to zero.
It is then possible to infer some information about the key.

THE ATTACK PRINCIPLE

FAULT INJECTION AS A PROBING TOOL

By comparing the outputs of two executions (one normal, one faulty) with same input, one infers whether the normal output of a faulted XOR is zero.

Assuming that the fault is certainly injected on the targeted XOR, an identity of ciphertexts implies that the fault was *ineffective*.
This reveals a local intermediate value equal (more precisely, equivalent) to 0.

If, for the same message, faults on two related XOR instructions are both ineffective, then the normal XOR outputs are simultaneously equal to zero.
It is then possible to infer some information about the key.

Remark: ‘simultaneously’ means *for the same input*, not that faults are injected *on the same execution*.

THE ATTACK PRINCIPLE

FAULT INJECTION AS A PROBING TOOL

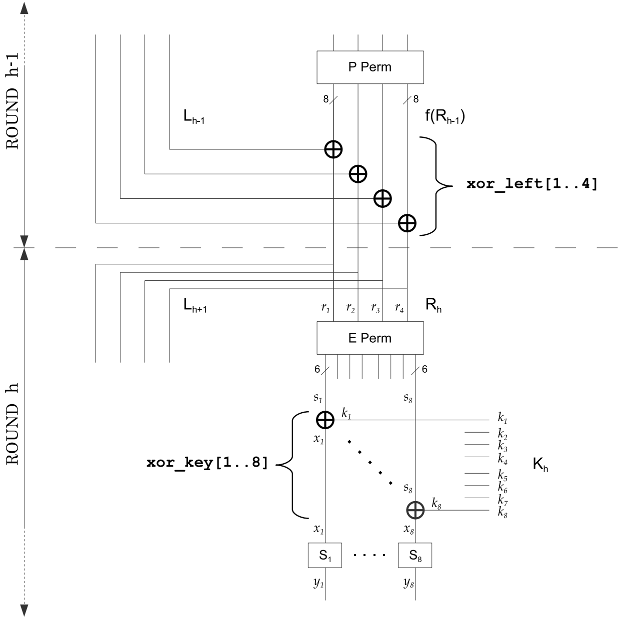
By comparing the outputs of two executions (one normal, one faulty) with same input, one infers whether the normal output of a faulted XOR is zero.

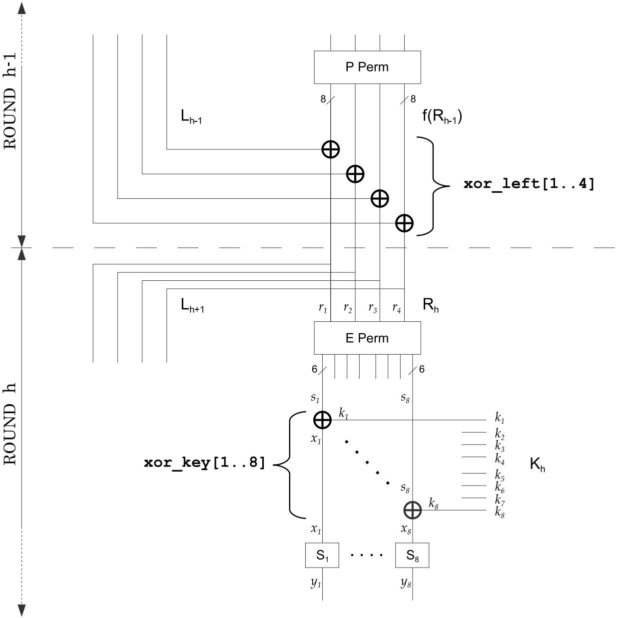
Assuming that the fault is certainly injected on the targeted XOR, an identity of ciphertexts implies that the fault was *ineffective*.
This reveals a local intermediate value equal (more precisely, equivalent) to 0.

If, for the same message, faults on two related XOR instructions are both ineffective, then the normal XOR outputs are simultaneously equal to zero.
It is then possible to infer some information about the key.

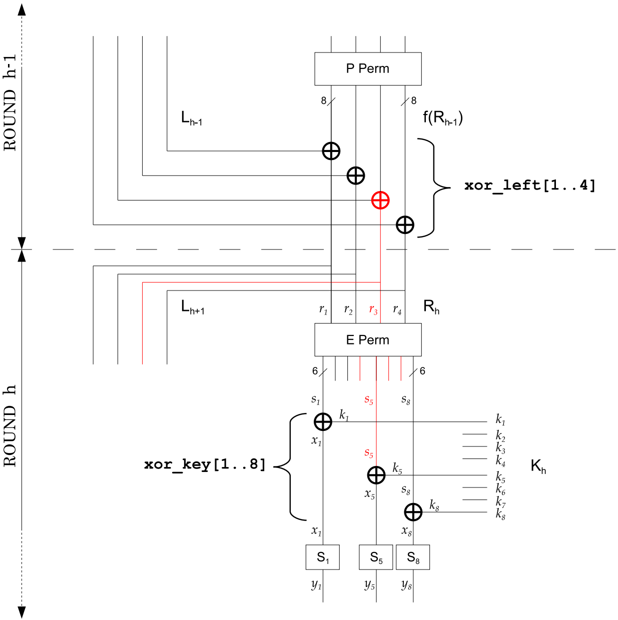
Remark: 'simultaneously' means *for the same input*, not that faults are injected *on the same execution*.

Indeed, the attacker does not need to inject 'multi-faults'.


ROUND h-1
P Perm
8
8
L_{h-1}
f(R_{h-1})
xor_left[1..4]
ROUND h
L_{h+1}
r1 r2 r3 r4
R_h
E Perm
6
6
s1
s8
k1
x1
xor_key[1..8]
k1 k2 k3 k4 k5 k6 k7 k8
K_h
s8
k8
x1
x8
S1
S8
y1
y8

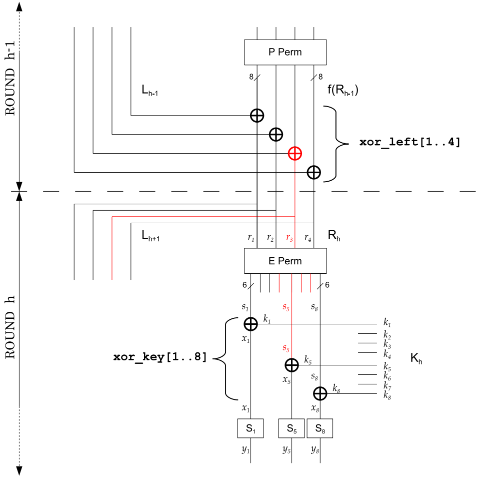


$$M \xrightarrow{normal} C$$



$$M \xrightarrow{\textit{normal}} C$$

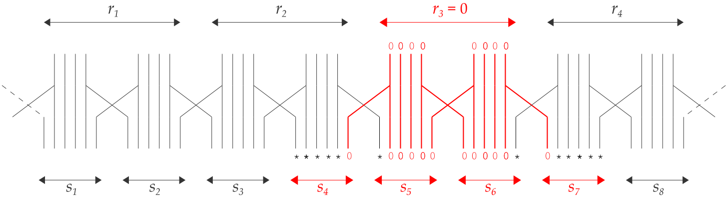
$$M \xrightarrow{\textit{fault}} \overset{\rightsquigarrow}{C}$$



$$M \xrightarrow{normal} C$$

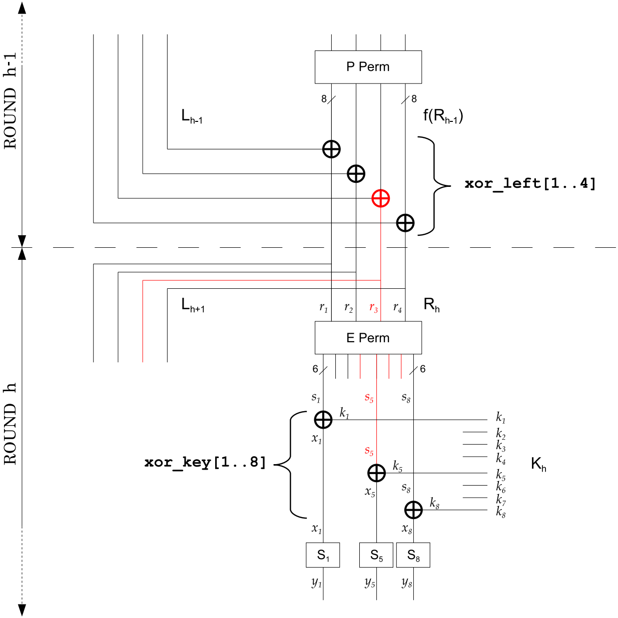
$$M \xrightarrow{fault} \overset{\leadsto}{C}$$

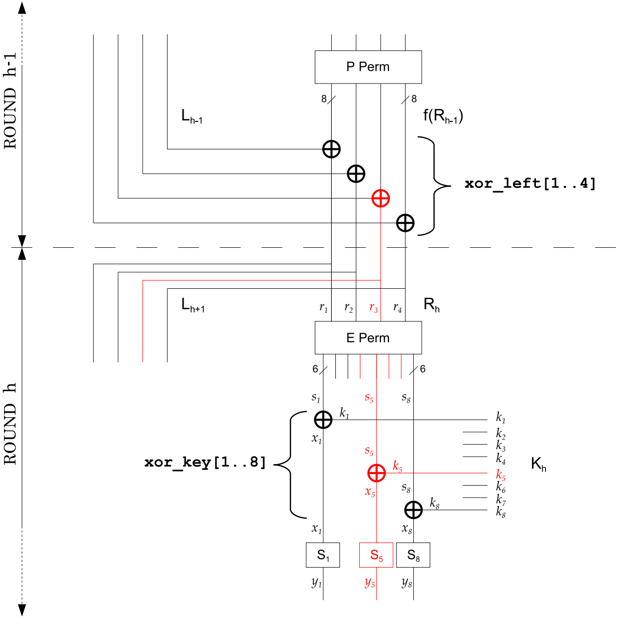
For some input M, observation that $C = \overset{\leadsto}{C}_{\texttt{xor\_left[3]}}$ implies that $r_3 = 0$

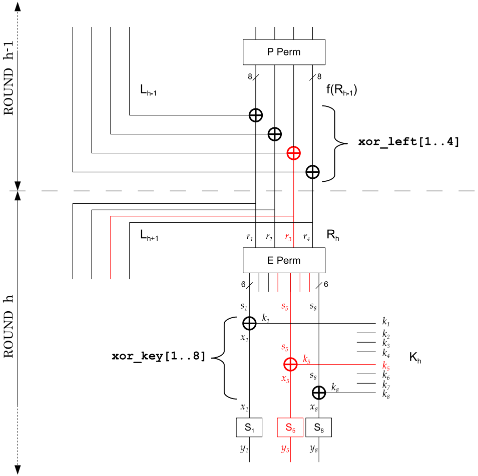


$r_3 = 0$ implies that s_5 and s_6 are almost zero after the expansive permutation.

Knowing that $s_5 \approx 0$, it may be interesting to know what happens when next XOR is also faulted: $\overset{\rightsquigarrow}{x_5} = s_5 \overset{\rightsquigarrow}{\oplus} k_5$.


ROUND h-1
ROUND h
P Perm
8
8
L_{h-1}
f(R_{h-1})
xor_left[1..4]
L_{h+1}
r_1 r_2 r_3 r_4
R_h
E Perm
6
6
s_1 s_5 s_8
k_1
x_1
xor_key[1..8]
s_5
k_5
x_5
s_8
k_8
x_1
x_8
k_1 k_2 k_3 k_4 k_5 k_6 k_7 k_8
K_h
S_1 S_5 S_8
y_1 y_5 y_8


ROUND h-1
P Perm
8
8
L_{h-1}
$f(R_{h-1})$
xor_left[1..4]
ROUND h
L_{h+1}
r_1 r_2 r_3 r_4
R_h
E Perm
6
6
s_1 s_5 s_8
k_1
x_1
xor_key[1..8]
s_5
k_5
x_5
s_8
k_8
x_1 x_8
k_1 k_2 k_3 k_4 k_5 k_6 k_7 k_8
K_h
S_1 S_5 S_8
y_1 y_5 y_8



If for the same input M, one also observes that $C = \overset{\leadsto}{C}_{\texttt{xor\_key[5]}}$, then:

$$x_5 \oplus s_5 \;=\; k_5 \;\in\; \mathcal{A}_5 \,\cup\, \big(\mathcal{A}_5 \oplus (1,0,0,0,0,0)\big) \qquad (\text{with } \mathcal{A}_5 = S_5^{-1}[S_5(0)])$$

THE BASIC ATTACK

Each such *double ineffective fault* reveals 3 bits of information about a round subkey. (8 remaining candidates out of 64.)

The attack consists in gathering this information about as much subkeys as possible. (This is key dependant)

EXPERIMENTAL RESULTS

Based on 27 000 simulations with random DES keys, the median residual entropy of the key is reduced from 56 bits to:

- 26.49 bits after 50 000 faults
- 22.32 bits after 100 000 faults

AN IMPROVED VERSION

More information about the key may be obtained by analysing ineffectiveness vectors.

DEFINITION

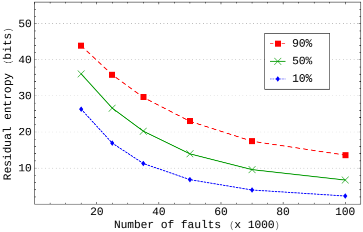
For any message M and any round $r \leqslant 2$, the *ineffectiveness vector* is the joint boolean observation of whether each of the `xor_left[i]` (at round $r-1$) and `xor_key[j]` (at round r) instructions is ineffective or not.

It is possible to compute for each key its *a posteriori* probability given the values of all ineffectiveness vectors observed so far. (See the paper for details.)

EXPERIMENTAL RESULTS

Based on 10 000 simulations, the median residual entropy is reduced to:

- 13.95 bits after 50 000 faults (instead of 26.49)
- 6.68 bits after 100 000 faults (instead of 22.32)



Problem: Final exhaustive search of the key is not possible, except if the attacker has access to an open device implementing the unknown function.

CLASSICAL COUNTER-MEASURES

What is the influence of classical counter-measures?

CLASSICAL COUNTER-MEASURES

What is the influence of classical counter-measures?

- Data masking should thwart the attack.
  (Except possibly in the 'multi-faults' model.)

CLASSICAL COUNTER-MEASURES

What is the influence of classical counter-measures?

- Data masking should thwart the attack.
  (Except possibly in the 'multi-faults' model.)
- Random delays and random order should make it very difficult.
  (*Random order only* should be breakable, see paper.)

CLASSICAL COUNTER-MEASURES

What is the influence of classical counter-measures?

- Data masking should thwart the attack.
  (Except possibly in the 'multi-faults' model.)
- Random delays and random order should make it very difficult.
  (*Random order only* should be breakable, see paper.)
- Double execution and verification has no effect on the attack.

OUTLINE

1. INTRODUCTION
   - Motivation
   - Externally encoded DES
   - Assumptions
2. DESCRIPTION OF THE ATTACK
   - The principle
   - The basic attack
   - An improved version
   - Classical counter-measures
3. CONCLUSION
   - Lessons
   - Open problems

LESSONS

Our result: it is possible to retrieve a DES key by transient fault analysis, even if its inputs/ouputs are not known by the attacker.

LESSONS

Our result: it is possible to retrieve a DES key by transient fault analysis, even if its inputs/ouputs are not known by the attacker.

LESSON 1

Just because access to inputs and outputs of an algorithm is not possible, don't assume it is fault analysis immune.

LESSONS

Our result: it is possible to retrieve a DES key by transient fault analysis, even if its inputs/ouputs are not known by the attacker.

LESSON 1

Just because access to inputs and outputs of an algorithm is not possible, don't assume it is fault analysis immune.

LESSON 2

Secret specifications does not always prevent key recovery.

OPEN PROBLEMS

Is it possible to devise similar fault attacks:

- based on other fault models?
- applicable to other externally encoded algorithms? (*e.g.* AES)

THANK YOU FOR YOUR ATTENTION !

QUESTIONS ?