Case Study of a Fault Attack on Asynchronous DES Crypto-Processors

Y. Monnet,
M. Renaudin, R. Leveugle

TIMA Laboratory
46 Av. Félix Viallet
38031 Grenoble Cedex

C. Clavier, P. Moitrel

Gemalto
La Vigie, Avenue du Jujubier
ZI athelia IV
13705 La Ciotat Cedex, France

FDTC
Oct 10th 2006

Outline

- Introduction :
  - Objectives
  - Quasi Delay Insensitive (QDI) asynchronous circuits
  - Previous work: practical results
- Asynchronous DES crypto-processors
  - The counter modules (reference and hardened)
- Experimental Setup
  - Laser characteristics and fault injection campaigns
- Fault exploitation
  - How can we retrieve the key ?
- Practical Results
  - Attacking the reference and hardened version
- Failure analysis and counter-measure
- Conclusion

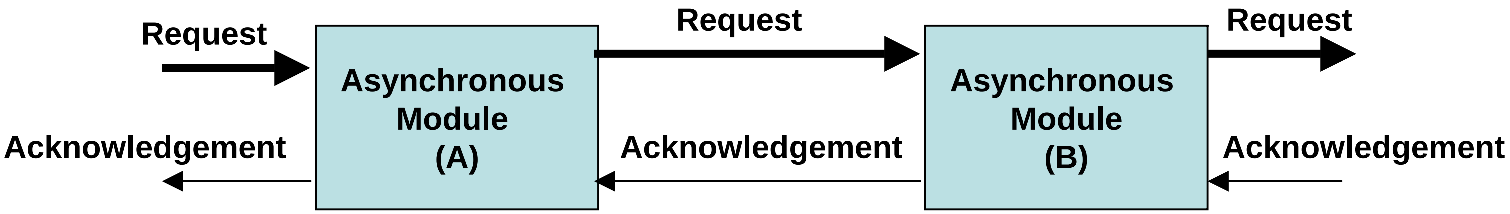
Introduction: objectives

- The DES algorithm :
  - Symmetric key algorithm
  - 16 iterations of a non linear transformation function

- Objective:
  - Reducing/Corrupting the number of rounds in a DES circuit to retrieve the key

Quasi Delay Insensitive Logic

- Quasi Delay Insensitive (QDI) Logic :



Handshake based communication between modules.
A module can actually be of any complexity.

- Distributed activity
- Multi-rail encoding
- Interesting properties:
  - Easy fault detection
  - Delay insensitivity
  - Interesting properties against DPA

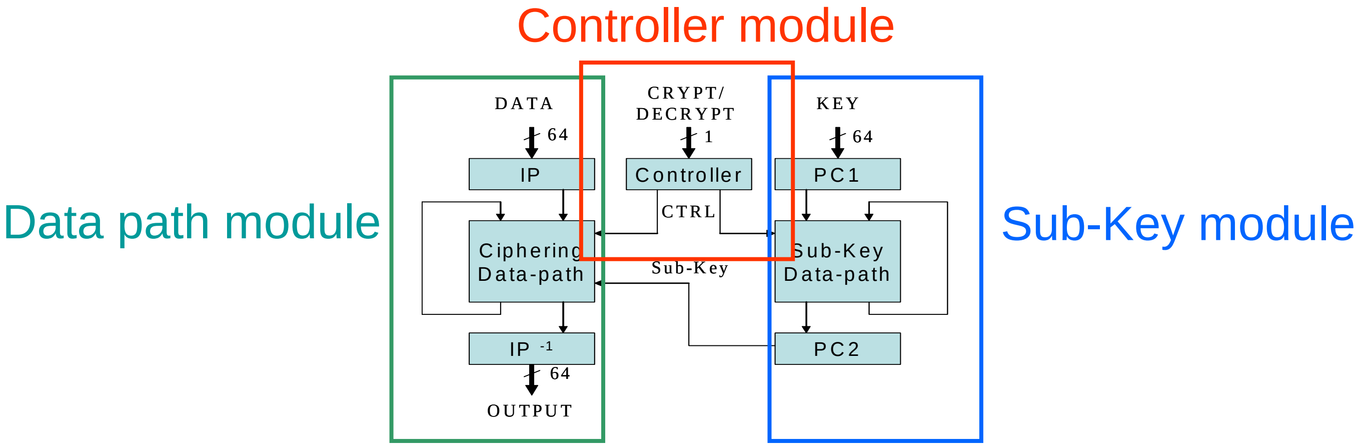
Introduction: Previous Work

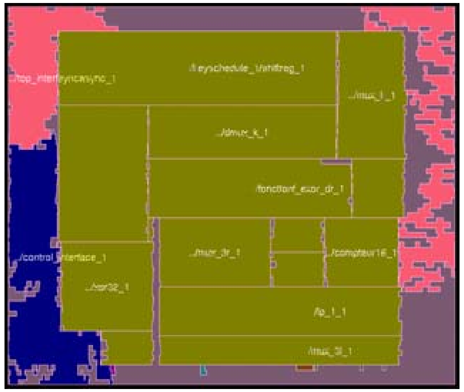
- Implementation of two asynchronous DES circuits:
  - **Reference version**: basic implementation, no optimizations
  - **Hardened version**: hardening techniques implemented in different parts of the circuit at the design level
- Previous work: attack on S-Boxes ***[TC'06]***
  - Using a laser fault injection
  - Practical evaluation/validation of the counter-measures
  - Exploiting faulty results to perform a DFA
    - ➢ Reference DES: so far, unsuccessful cryptanalysis on the S-Boxes
    - ➢ Hardened DES: no faulty result to exploit !

- Objectives of this work:
  - To attack the circuits by injecting faults in the counter module
  - Round number corruption => information to retrieve the key

Asynchronous DES crypto-processors

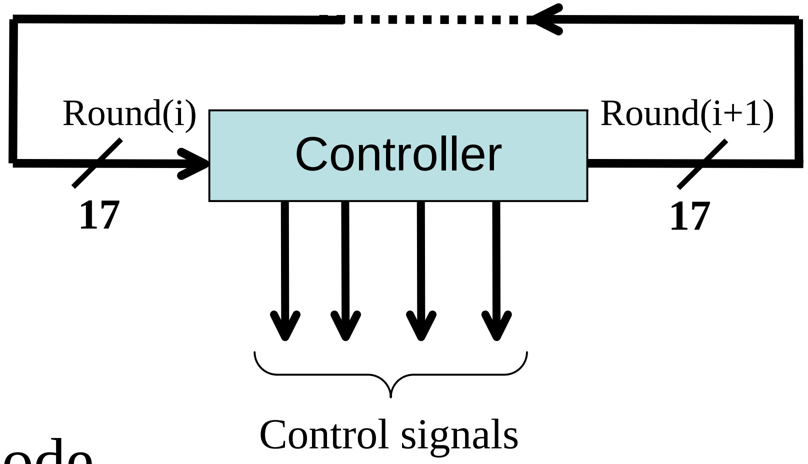
- Global circuits architecture:





- 130 nm STmicroelectronics CMOS process
- Constrained floor plan to provide a better localization and to avoid side effects in order to characterize the circuits

The counter module



- 1-out-of-17 code

00000000000000001 ← Round 1
00000000000000010 ← Round 2
....
01000000000000000 ← Round 16
10000000000000000 ← Exit !

- Data path control signal: Loop/Exit
- Key scheduling control signal: Left/Right shift, 1/2 bits

The counter module

- Control signals are computed from the 1-out-of-17 signal
  ⇒ What happens if the 1-out-of-17 is corrupted ?

00000000**1**000000**1**0
Round 2 is corrupted: wrong control signal generation

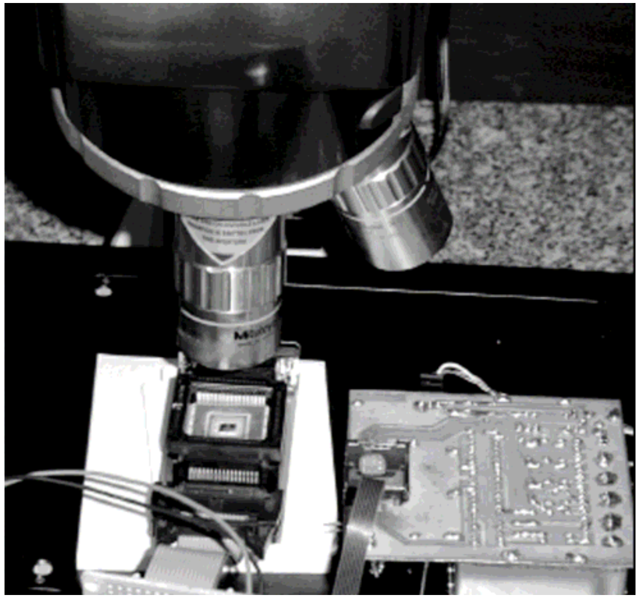
100000000000000**1**0
Round 2 is corrupted: wrong control signal generation and EXIT!
Notation: **[2 → 17]**

00000**1**000000000**0**0
Counter is corrupted from Round 2 to Round 12. **[2 → 12]**

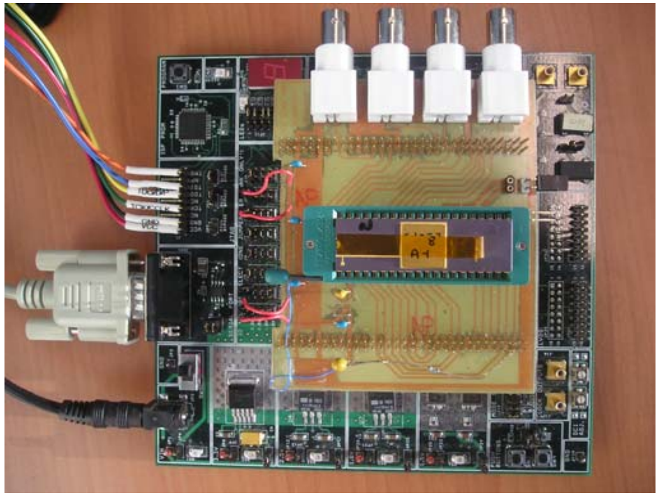
- Hardened version of the counter:
  - Alarms cells that detect any wrong p-out-of-17 code with 1 < p <= 17
  - The environment is informed with an alarm signal

Experimental Setup

- Laser Characteristics
  - Green Laser
  - 6 ns pulse
  - Tunable spot size (220 µm²)
  - Tunable energy (0.8 pJ/µm²)



Gemalto laser platform



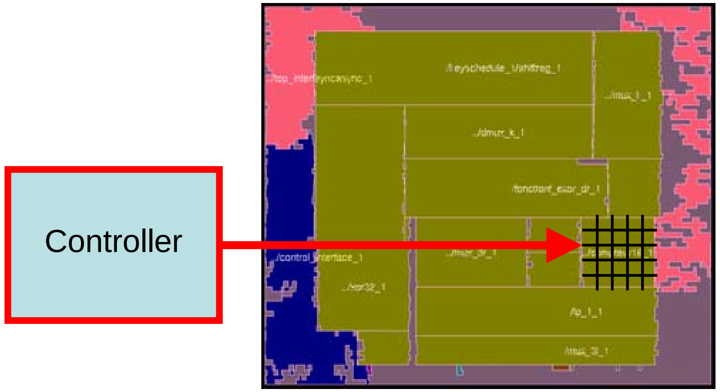
Laser Board for the DES

Experimental Setup

- Fault injection campaign

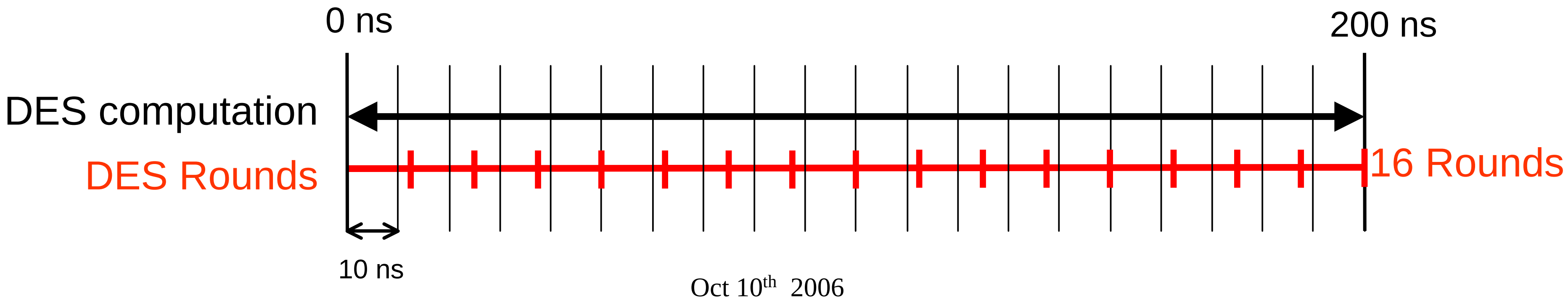
  - Spatial scan :

    White Box approach
    => Circuit under test and coordinates are known



Controller

  - Time scan :



0 ns

200 ns

DES computation

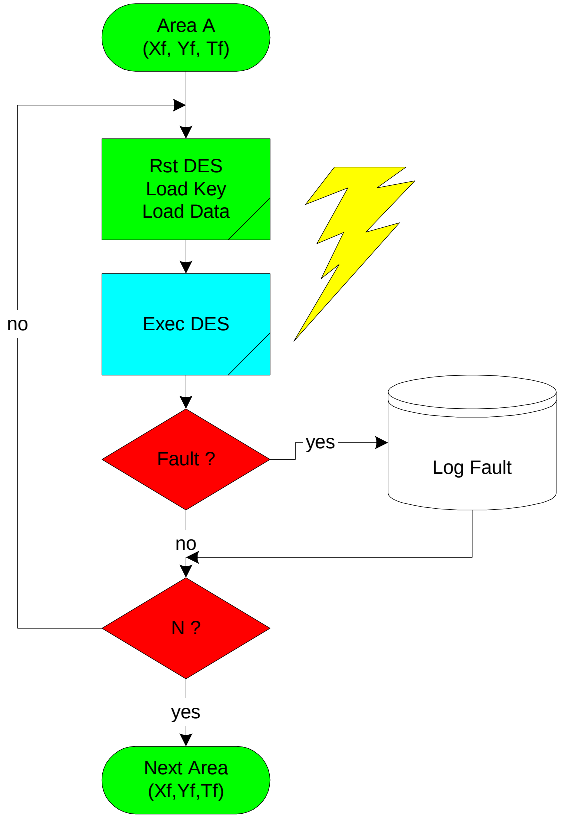
DES Rounds

16 Rounds

10 ns

Experimental Setup

- Fault injection campaign



For each coordinate **[X,Y]**

For each time position **T**

For each iteration **N**

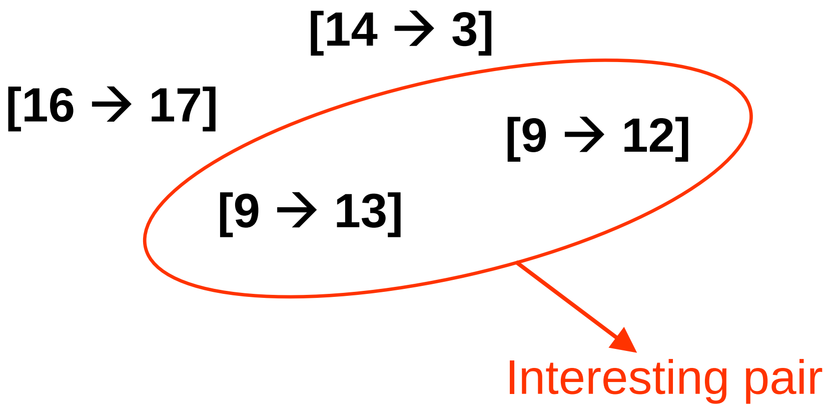
- Reset the circuit
- Load key and data
- Start the computation
- Shoot !

- 2 shots/second
- Over 5000 shots per campaign

Oct 10th 2006
Yannick Monnet

Fault Exploitation

- How can we retrieve the key ?
- Context:
  - We suppose a constant plain text during the campaign
  - The correct cipher text is known
  - We obtained a pool of round corrupted results:



- Results are analyzed by pairs whose rounds execution are close

Fault Exploitation

Round execution of **[9 → 12]**:

(1, 2, 3, 4, 5, 6, 7, 8, 12, 13, 14, 15, 16, 17)

Round execution of **[9 → 13]**:

(1, 2, 3, 4, 5, 6, 7, 8, 13, 14, 15, 16, 17)

Sub-key sequence of **[9 → 12]**: (**1,2,4,6,8,10,12,14,16,18,20**,22,23)

Sub-key sequence of **[9 → 13]**: (**1,2,4,6,8,10,12,14,16,18,20**,21)

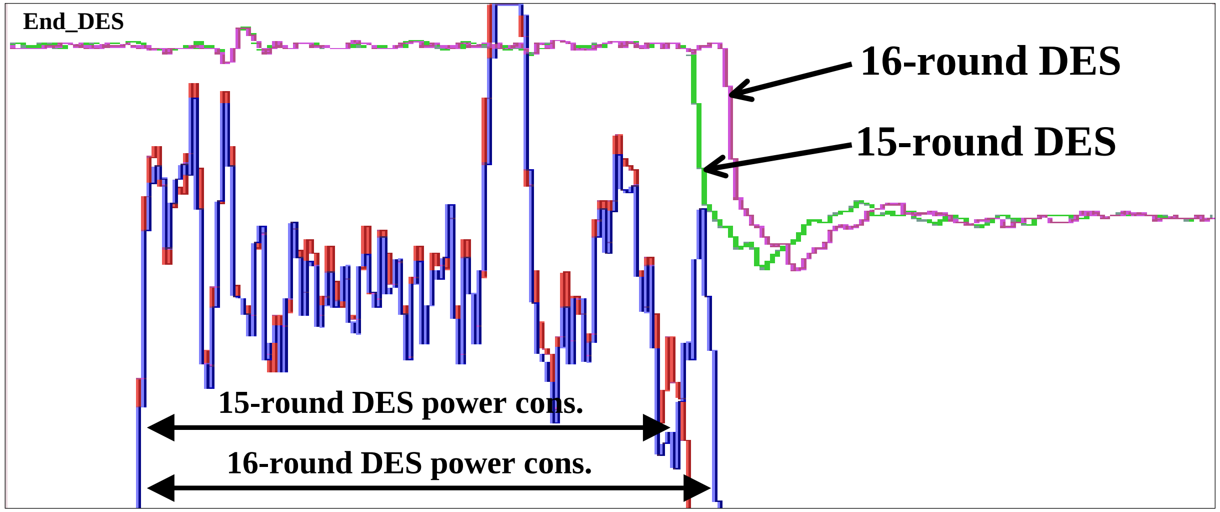
⇒ The prefix part gives us enough information on the input/outputs of the remaining part to perform an easy cryptanalysis on the last round of [9 → 12]

Practical Results

- Reference DES:
  - Over 50 faulty results were identified as round corruptions
  - Large pool of pairs that provide an easy cryptanalysis to retrieve the key
  - Reproducible results !

- Hardened DES:
  - Same behavior as the reference version, but alarms are raised
    - ⇒ Most of the faults are detected
  - However, some of the faults remain undetected!
    - ⇒ The counter was corrupted, no alarm raised
  - Some of the results are reproducible

Practical Results



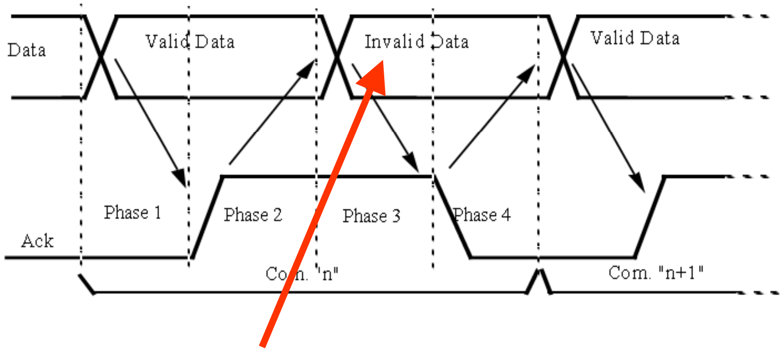
Undetected and reproducible [16 → 17] sequence

- The DES execution is shortened of 1 round (12 ns)
- No alarm raised, the 1-out-of-17 code was corrupted into a valid code

Failure Analysis

- Two hypotheses of failure can explain this result:
  - Multiple fault injection: possible but unlikely
  - Single fault injection: exploitation of a weakness in the communication protocol

    ⇒ Formally verified



Return to zero phase

Counter-Measure

- The failure was identified in a formal way and verified in simulation: 1 bit-flip with a determined timing constraint is enough to corrupt the code.
- Counter-measure: control of the timing constraints between the modules at the design time. This can be achieved by implementing a synchronization control circuit to handshake with the controller *[TC'06]*

  ⇒ The efficiency was measured in simulation

Conclusion

- Successful attack on both the reference and the hardened version
- Hardened version showed a much better security level
- But there are some weaknesses left !
- Weaknesses were analyzed and characterized both in a simulation environment and a formal environment
- Efficient counter-measures can be applied to increase the security level for a low cost
- Asynchronous technology is an attractive alternative to design robust systems

Conclusion

- The attack was performed in the best conditions
  - White box approach
  - Constrained floor plan
  - No security strategy provided

- The work showed:
  - The feasibility
  - The potential of fault attacks: even multi-rail schemes protected by alarms cells may not be completely safe