Hardware Security exam (2 hours)

Renaud Pacalet - 2025-06-24

You can use any document but communicating devices are strictly forbidden. Please number the different pages of your paper and indicate on each page your first and last names. You can write your answers in French or in English, as you wish. Precede your answers with the question's number. If some information or hypotheses are missing to answer a question, add them. If you consider a question as absurd and thus decide to not answer, explain why. If you do not have time to answer a question but know how to, briefly explain your ideas. Note: copying verbatim the slides of the lectures or any other provided material is not considered as a valid answer. Advice: quickly go through the document and answer the easy parts first.

To solve parts of this exam you will need a global understanding of the DES encryption/decryption standard. The provided appendix should be sufficient.

The 5 questions are worth 2 points each. The problem is worth 10 points.

1. Questions

1.1. Timing attacks

The course enumerates 5 different hypotheses for a timing attack to be practical against the modular exponentiation $z = y^x \bmod n$, where n is the public modulus and x is the secret exponent:

- The cryptosystem takes different amounts of time to process different inputs.
- Timing depends on secret exponent x and input data y.
- The attacker knows the input data y.
- For several input data y the victim computes $y^x \bmod n$ and the attacker records the timing.
- The attacker knows the implementation and uses this knowledge to exploit the timing measurements.

For each of them propose a countermeasure based on canceling it and discuss the efficiency, pros and cons of such a solution.

1.2. On-board probing attacks

To prevent on-board probing attacks one can encipher / decipher the bus between a microprocessor and its external memories. In case the microprocessor has on-chip data and instruction caches, as shown on Figure 1, where would you put the encryption / decryption engines? Between the CPU and the caches (position A) or between the caches and the memory controller (position B)? Why? What are the consequences of this choice? Is it different for data and for instructions?

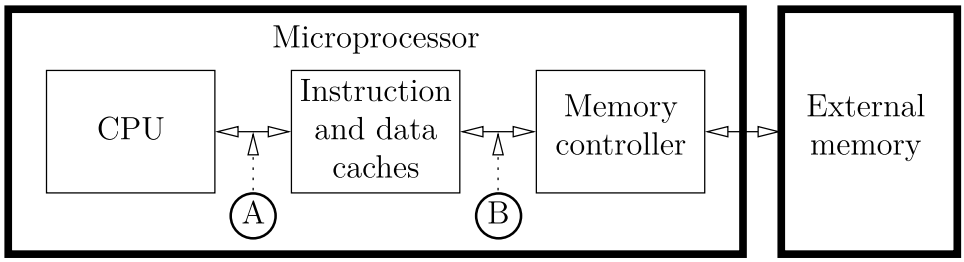

Figure 1: A microprocessor and its external memory

1.3. Fault attacks

As we saw during the lectures, fault attacks against DES are possible and can significantly reduce the cost of brute-forcing the secret key. Consider a DES hardware implementation similar to the one we attacked during the second lab about power attacks, and represented on Figure 2. Assume you are asked to protect it against fault attacks. Propose two different countermeasures, discuss their efficiency, their advantages and their drawbacks. Which one would you chose? Why?

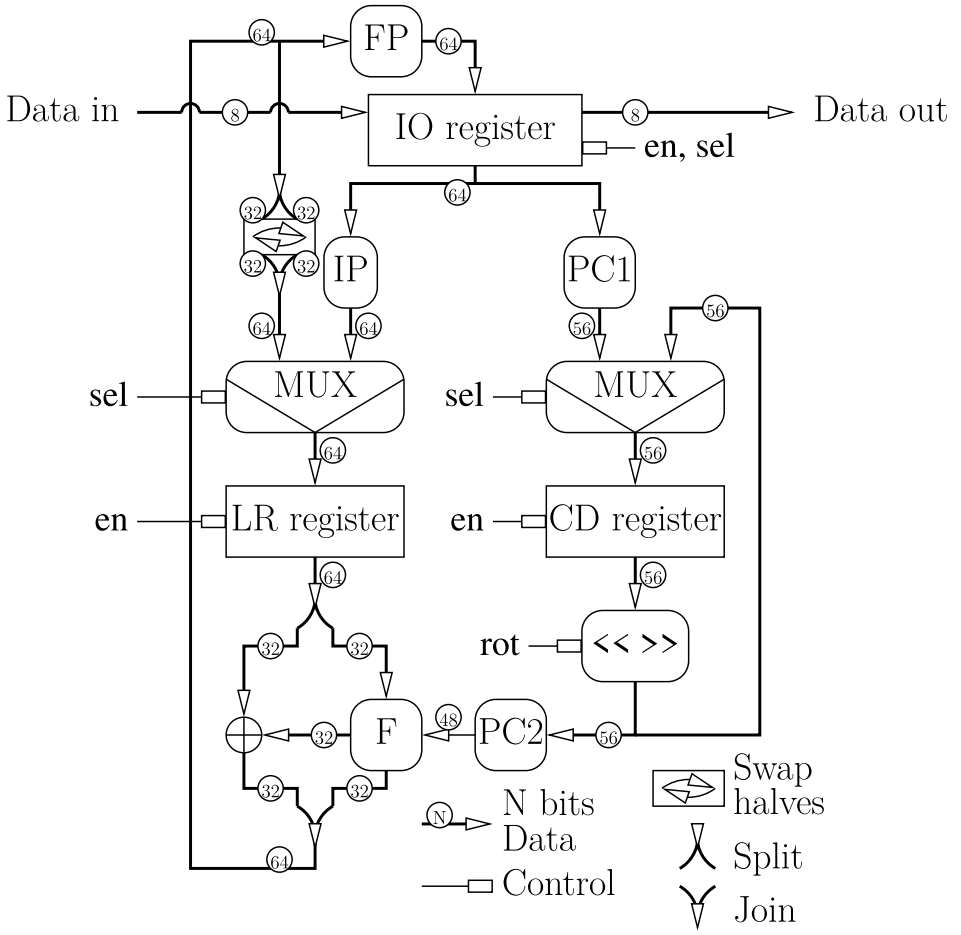

Figure 2: DES hardware implementation

1.4. Countermeasures against on-chip probing

AES-256 (Advanced Encryption Standard with 256 bits secret keys) is a symmetric block cipher, just like DES. The main differences are the block length (128 bits for AES-256 instead of 64 for DES) and the key length (256 bits for AES-256 instead of 56 for DES). A company decides to design and manufacture an Integrated Circuit (IC) containing a hardware AES-256 engine. In order to protect it against on-chip probing attacks an engineer decides to slightly modify the AES-256 algorithm by replacing the standard SBoxes by his own SBoxes and to keep these custom SBoxes secret. This way, even if an attacker was probing the IC during operations, understanding what he observes would be much more difficult. Discuss the efficiency and the cost of this countermeasure. What are its main advantages and drawbacks?

1.5. Power attacks

Triple DES encryption (TDEA) is defined as:

$$\mathrm{TDEA}_{k_1,k_2,k_3}(P) = \mathrm{DES}_{k_3}\Big(\mathrm{DES}^{-1}_{k_2}\Big(\mathrm{DES}_{k_1}(P)\Big)\Big)$$

where k_1, k_2, k_3 are 3 DES secret keys, $\mathrm{DES}_{k(P)}$ is the DES encryption of plaintext P with key k and $\mathrm{DES}^{-1}_k(C)$ is the DES decryption of ciphertext C with key k. Is TDEA significantly more robust than simple DES against power attacks? If no, why? If yes, why and how much more robust?

2. Problem: Timing attacks against DES

In the following we consider a software DES implementation running on a 8 bits micro-controller with 16 bits address bus. The $2^{16} = 65536$ bytes address space is split in M_{fast}, a small 200 bytes fast memory starting at address 0 and accessed in only one clock clycle, and M_{slow}, a $65536 - 200 = 65336$ bytes slower memory starting at address 200 and accessed in two clock cycles.

To speed-up the processing the eight DES S-boxes are implemented as a 256 bytes table in memory starting at address 0, as shown on Figure 3

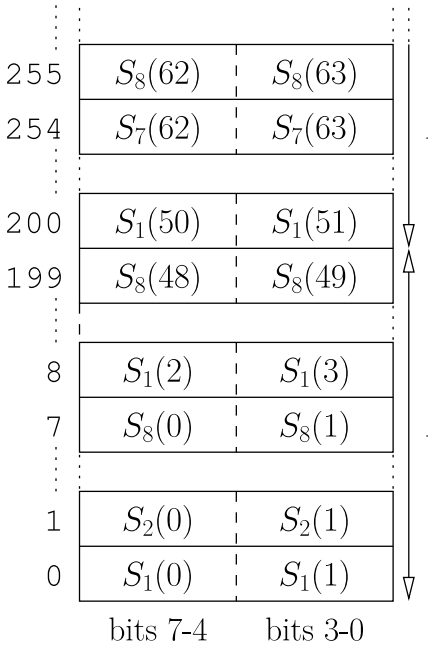

Figure 3: The memory layout of the S-boxes table

To compute the 4 bits output of S-box number s ($1 \leq s \leq 8$) corresponding to the 6 bits input x ($0 \leq x \leq 63$), the program simply reads the byte in memory at address $8 \times \lfloor \frac{x}{2} \rfloor + s - 1$ (where $\lfloor v \rfloor$ is the largest integer not greater v), and keeps the 4 left bits is x is even, else the 4 right bits.

Unfortunately M_{fast} is only 200 bytes long and cannot hold the full table. The S-box computation time thus depends on the input data (for instance, computing $S_8(17)$ is faster than computing $S_8(53)$), which you should know as a potential vulnerability to timing attacks. We assume that this access time difference between M_{fast} and M_{slow} during S-boxes computation is the only part of the whole algorithm for which the timing depends on the input data.

Assume you are in charge of attacking this implementation. You can observe the encryption of as many 64 bits data blocks as you want with the same secret key K. For each encryption you can measure the **total** time it takes, and see the output 64 bits block (you don't have access to the input blocks). Your goal is to retrieve the secret key K.

What would be your attack?

Represent your attack algorithm in the form of some pseudo-code with clearly defined hypotheses and notations.

What amount of information can you extract?

What is the cost of your attack (amount of computation, amount of storage)?

Is it practical?

If yes propose a countermeasure and discuss its efficiency and its drawbacks.

Appendix: the DES encryption/decryption standard

IP and FP are 64 to 64 bits permutations, inverse one of the other. $\oplus$ is the bitwise exclusive OR of two bit strings. E is a 32 to 48 bits expansion-permutation. P is a 32 to 32 bits permutation. $\mathrm{S}_1, \mathrm{S}_2, ..., \mathrm{S}_8$ are 8 different 6 to 4 bits non linear substitution functions (SBoxes). PC_1 is a 64 to 56 bits selection-permutation. PC_2 is a 56 to 48 bits selection-permutation. LS is a 28 to 28 bits rotation by one or two positions to the left, depending on the round index; it is used in the encryption key schedule (as show in Figure 4). RS is a 28 to 28 bits rotation by one or two positions to the right, depending on the round index; it is used in the decryption key schedule. All these primitive functions are perfectly defined in the DES standard. DES decryption is the same as encryption with the round keys used in reverse order: K_{16} in the first round, K_{15} in the second, and so on, with K_1 used in the 16^{th} round. This reverse order is obtained by using RS instead of LS in the key schedule.

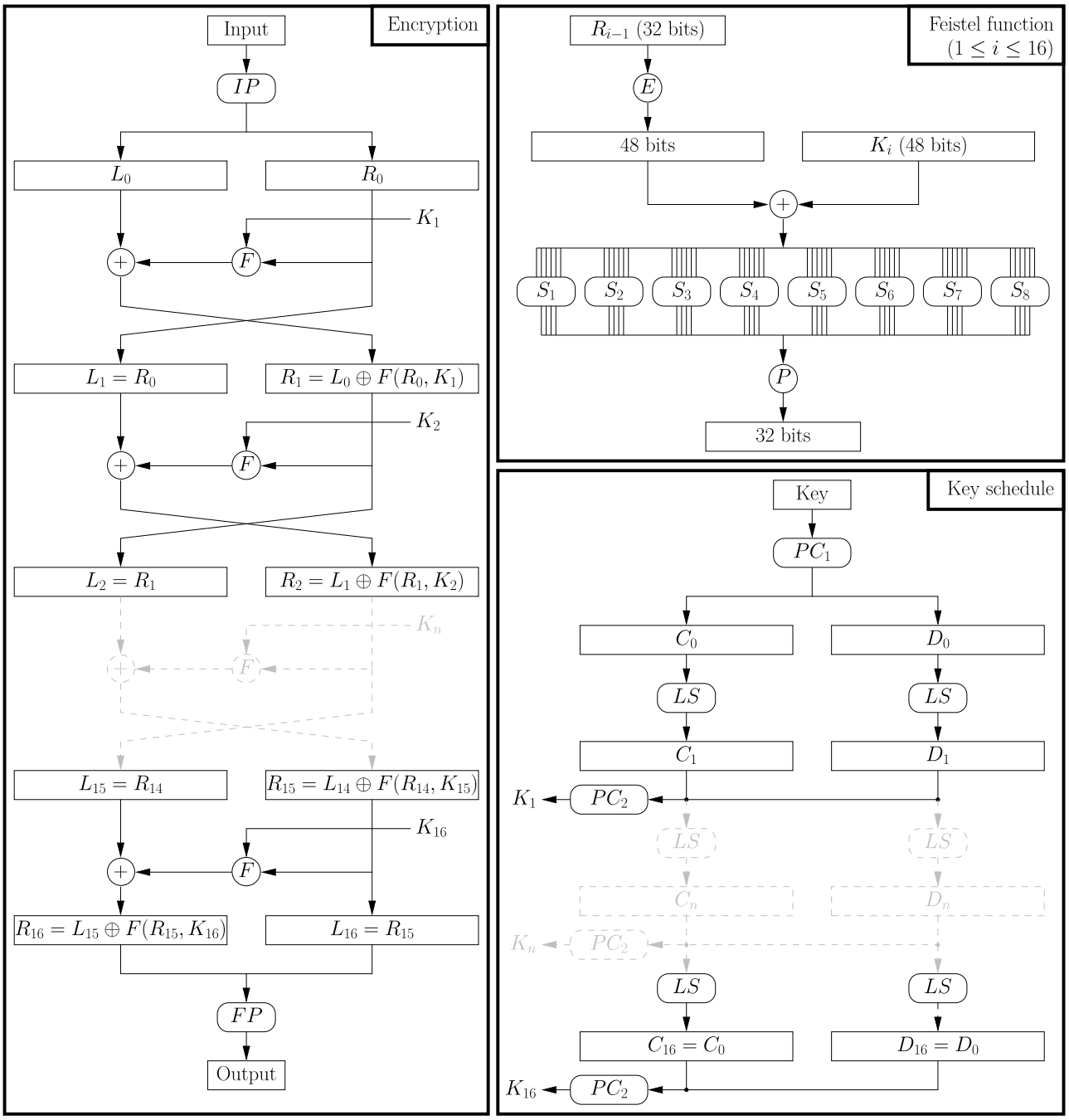

Figure 4: DES encryption