Hardware Security exam (2 hours)

Renaud Pacalet - 2026-04-08

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.

The 3 questions are worth 3 points each. The problem is worth 11 point.

1. Questions

1.1. On-Board probing attacks

We consider a computer system with CPU and external memory as shown on Figure 1.

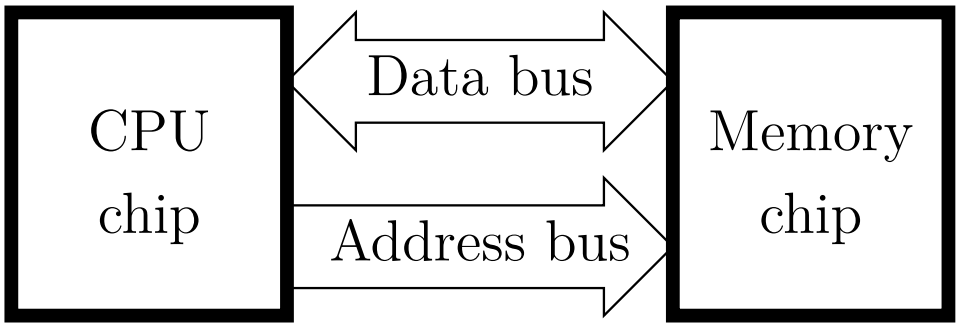

Figure 1: A microprocessor and its external memory

To protect it against on-board probing attacks, one can use code and data encryption: a Hardware Security Module inside the CPU encrypts on-the-fly every instruction or data that the CPU writes in the external memory. The HSM also decrypts on-the-fly every instruction or data that the CPU reads from the external memory, before it is used by the CPU.

Explain why the code and data encryption is not sufficient to protect this computer system against on-board probing attacks. Do you know an attack that this countermeasure cannot prevent?

1.2. Side channel attacks

Among the different leak sources usually designated by the term "*side channel*" we find the computation time, the power consumption and the electromagnetic emissions. Assume you are in charge of designing countermeasures. In your opinion what leak will be the easiest to fix?

Why?

And what about the 2 others?

1.3. Fault attacks

Explain why the RSA implementations based on the Chinese Remainder Theorem are considered as sensitive to fault attacks.

2. Problem: timing attacks against RSA signature

The notations used in this problem are the following:

- $|x|$: bit-width of x
- x_0: Least Significant Bit (LSB) of x
- x_k: k^{th} bit of x
- $x_{|x|-1}$: Most Significant Bit (MSB) of x

Let $n = p \cdot q$ be the public modulus of a RSA key pair and d the corresponding private exponent, where p and q are the two secret large prime numbers. Reminder: the RSA signature of a message m with public modulus n and private exponent d is $s = m^d \bmod n$.

In this problem we consider three different implementations of RSA signature. They all use the multiply-and-square exponentiation algorithm, one exponent bit per iteration from LSB to MSB, which pseudo-code is shown in Algorithm 1.

```
1:  Inputs: m, d, n
2:  Output: s = m^d mod n
3:  s ← 1 // initial value of s
4:  u ← m // initial value of u (temporary variable)
5:  for k ← 0 to |d| − 1 // loop from LSB to MSB of d
6:      if d_k = 1
7:          s ← (s · u) mod n // modular multiplication
8:      end if
9:      u ← u^2 mod n // modular square
10: end for
11: return s
```

Algorithm 1: Multiply-and-square (MS1) exponentiation algorithm

An attacker tries to recover as much information as possible about the secret exponent d. They have access to the public modulus n and they can send as many messages to sign as they want. For each message m they can record the output signature $s = m^d \bmod n$ and the **total** exponentiation time.

2.1. Timing attack of a software implementation

A first implementation of RSA signature, `RSA-sig`, is 100% software. The modular multiplication (Line 7) and modular square (Line 9) are optimized for speed using various techniques. A consequence of these optimizations is that their computation time depends on their operands. As seen in class this indicates that the implementation is maybe prone to timing attacks.

What amount of information could the attacker obtain?

Provide a description of the attack. Carefully and clearly define your notations, express your attack algorithm in a semi-formal (pseudo-language), complete and non ambiguous way.

2.2. Timing attack of a mixed hardware-software implementation

Another implementation, `RSA-sig-hw`, is a mixture of hardware and software. The modular square (Line 9) is the same as in `RSA-sig`, its computation time depends on its operands, but

the modular multiplication (Line 7) is accelerated with a hardware modular multiplier that computes in constant time.

What amount of information could the attacker obtain?

Provide a description of the attack. Carefully and clearly define your notations, express your attack algorithm in a semi-formal (pseudo-language), complete and non ambiguous way.

2.3. Timing attack of a Chinese Remainder Theorem (CRT) software implementation

A third implementation, `RSA-sig-crt`, uses the Chinese Remainder Theorem (CRT) optimization shown in Algorithm 2 were we denote:

- $dp = d \bmod (p-1)$
- $dq = d \bmod (q-1)$
- $r = q^{-1} \bmod p$

1: **Inputs**: n, m, p, q, r, dp, dq
2: **Output**: $s = m^d \bmod n$
3: $sp \leftarrow (m \bmod p)^{dp} \bmod p$
4: $sq \leftarrow (m \bmod q)^{dq} \bmod q$
5: $s \leftarrow ((sp - sq) \cdot q \cdot r + sq) \bmod n$

Algorithm 2: CRT optimization of RSA signature

Like `RSA-sig` it is 100% software and its modular multiplications and modular squares are optimized for speed with computation times that depend on the operands.

What amount of information could the attacker obtain?

Provide a description of the attack. Carefully and clearly define your notations, express your attack algorithm in a semi-formal (pseudo-language), complete and non ambiguous way.