# Final exam – April 27., 2026

*No document is allowed. Unless indicated otherwise, answers must be carefully justified to receive full credit. The exercises are independent, but questions within the same exercise may depend on each other. You may use a result from a previous question by clearly stating it. You may answer in English or French. The duration is 2 hours.*

**There are three exercises, do not forget to turn the page!**

## Exercise 1 (5 points). *Short questions*

1. Attribute each one of the three security notions IND-CPA, PRP and EUF-CMA to one of the three primitives block cipher, public-key encryption scheme, MAC.
2. Let $N = p \times q$ where $p$ and $q$ are distinct prime numbers. Express the number of invertible elements in $\mathbb{Z}/N\mathbb{Z}$ in terms of $p$ and $q$. *Justify.*
3. In a public-key encryption scheme, two keys are used: the *encryption* and the *decryption* keys. Which one is supposed to be *public* and which one must remain *private*? *Justify.*
4. We recall the three problems related to a cyclic group $G = \langle g \rangle$:
   - CDH: given $g^a$ and $g^b$, compute $g^{ab}$;
   - DDH: given $g^a$, $g^b$ and $g^c$, determine if $g^c = g^{ab}$;
   - DLP: given $g^a$, compute $a$.

   Sort the three problems by increasing order of difficulty. *Justify.*
5. Define what a Key Encapsulation Mechanism is, in which situation it can be used, and what benefits it brings.

## Exercise 2 (6 points). *Lamport signatures*

Let $H : \{0,1\}^{<N} \to \{0,1\}^m$ be some hash function and $\mathcal{M} = \{0,1\}^k$ be the message space. The **Lamport signature scheme** is defined by

- $\mathsf{Gen}_n()$:
  1 for $1 \leq i \leq k$ and $b \in \{0,1\}$ :
  2 $\quad x_i^{(b)} \twoheadleftarrow \{0,1\}^n$ (uniformly)
  3 $\quad y_i^{(b)} \leftarrow H(x_i^{(b)})$
  4 return $\mathsf{sk} = \begin{pmatrix} x_1^{(0)} & x_2^{(0)} & \cdots & x_k^{(0)} \\ x_1^{(1)} & x_2^{(1)} & \cdots & x_k^{(1)} \end{pmatrix}$ and $\mathsf{vk} = \begin{pmatrix} y_1^{(0)} & y_2^{(0)} & \cdots & y_k^{(0)} \\ y_1^{(1)} & y_2^{(1)} & \cdots & y_k^{(1)} \end{pmatrix}$
- $\mathsf{Sign}_{\mathsf{sk}}(m)$:
  0. write $m = m_1 \| \ldots \| m_k$ where $m_i \in \{0,1\}$ for $1 \leq i \leq k$
  1. return $\sigma = (x_1^{(m_1)}, x_2^{(m_2)}, \ldots, x_k^{(m_k)})$

For instance with $k = n = 3$, if $\mathsf{sk} = \left(\begin{smallmatrix} 010 & 011 & 101 \\ 110 & 100 & 001 \end{smallmatrix}\right)$, $\mathsf{Sign}_{\mathsf{sk}}(001) = (010, 100, 001)$.

1. **i.** Describe the verification algorithm. *The algorithm must be completely specified, including its inputs.*
   **ii.** Give a necessary and sufficient condition on the keys for the verification algorithm to only accept valid signatures.
2. Provide a two-chosen-message key recovery attack on the Lamport signature scheme. *That is, an adversary that can make two signature queries of its choice can compute the signing key.*
3. In the original signature scheme, the space of messages is fixed to $\{0,1\}^k$. We want to use the signature scheme with message of length *at most* $k$. The simple solution is to define the signature

of a message $m$ of length $\ell \leq k$ as $(x_1^{(m_1)}, \ldots, x_\ell^{(m_\ell)})$.

**i.** Give a one-chosen-message universal forgery attack on this variant. *That is, an adversary is able to compute a valid signature $\sigma$ for a given message $m$, after getting the signature $\sigma'$ of another message $m'$ of its choice.*

**ii.** Propose a variant of the signature scheme that allows to sign any message of length $\leq k$, with keys of size $2(k + \lceil \log k \rceil)$. Argue that it prevents the previous attack.

**Exercise 3 (9 points).** *Pedersen hash function*

Let $G = \langle g \rangle$ be a cyclic group of prime order $p$ and $h = g^a \in G$. Define the compression function $f : \{0, \ldots, p-1\} \times \{0, \ldots, p-1\} \to G$ by $f(x, y) = g^x h^y$.

**1.** Prove that if an adversary is able to compute discrete logarithms in $G$, it can solve the following problems. *Provide an explicit attack for each problem.*

**i.** First preimage: Given $h$ and $t$, compute $(x, y)$ such that $f(x, y) = t$.

**ii.** Second preimage: Given $h$ and $(x_0, y_0)$, compute $(x_1, y_1) \neq (x_0, y_0)$ such that $f(x_0, y_0) = f(x_1, y_1)$.

**iii.** Collision: Given $h$, compute $(x_0, y_0)$ and $(x_1, y_1) \neq (x_0, y_0)$ such that $f(x_0, y_0) = f(x_1, y_1)$.

**2.** We prove now the converse: Assume that an adversary is able to find a collision $f(x_0, y_0) = f(x_1, y_1)$ with $(x_0, y_0) \neq (x_1, y_1)$. Prove that it can compute the discrete logarithm $a$ of $h$.

**3.** **i.** Recall the name and the cost of the best generic algorithm for the DLP in a group of order $p$.

**ii.** Recall the name and the cost of the best generic algorithm for finding collisions for a hash function that takes values in a set of size $p$.

**iii.** Recall the best generic algorithm and its cost to compute a first (resp. second) preimage for a hash function that takes values in a set of size $p$.

**iv.** Would you say that $f$ has a *good* first (resp. second) preimage resistance? a *good* collision resistance? *Justify.*

To build a hash function from the compression function, we are given an efficiently computable one-to-one correspondence $\pi : G \to \{0, \ldots, p-1\}$. We view an element of $\{0, \ldots, p-1\}$ as a *block* and define hash functions that take as input an $n$-block message, $n \leq N$, and output a one-block digest. We propose two constructions :

- Fix $h \in G$, define $\bar{f} : \{0, \ldots, p-1\} \times \{0, \ldots, p-1\} \to \{0, \ldots, p-1\}$ by $\bar{f}(x, y) = \pi(f(x, y)) = \pi(g^x h^y)$, and build $H$ from $\bar{f}$ using Merkle-Damgård construction;
- Fix $h_1, \ldots, h_N \in G$ and for any $n \leq N$, define $F(x_1 \| \cdots \| x_n) = \pi\left(\prod_{i=1}^n h_i^{x_i}\right)$ where each $x_i \in \{0, \ldots, p-1\}$.

**4.** **i.** Explain the Merkle-Damgård construction to build a hash function $H$ from $\bar{f}$.

**ii.** Which construction provides the most efficient hash function for an $n$-block message? *The relevant measure of complexity for the comparison is the number of exponentiations needed to hash the n-block message.*