Final exam – April 25., 2025

No document is allowed. Except indicated otherwise, answers must be carefully justified to get maximum credit. The exercises are independent, but not all questions. You may admit a result from a previous question by clearly stating it. You may answer in English or French.

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

Exercise 1 (4 points). *Authentication*

1. What is the difference between a *passive* and an *active* adversary? What type of adversary can authentication protect against?
2. Describe the similarities and differences between a MAC and a signature scheme.
3. Bob wants to send an authenticated message to Alice, using a signature scheme. Which key(s) should Bob use to authenticate the message? Which key(s) Alice must use to verify the authenticity of the message? *In both cases, precise the kind of key and the owner.*
4. Describe the *Hash-and-sign* paradigm for designing a signature scheme. What is its purpose? What security assumption is required on the hash function to avoid an existential forgery?

Exercise 2 (9 points). *Even-Mansour construction*

Let $\pi : \{0,1\}^n \to \{0,1\}^n$ be a public (random) permutation: Everyone is able to apply π *or its inverse* to any $x \in \{0,1\}^n$. Let $E : \{0,1\}^{2n} \times \{0,1\}^n \to \{0,1\}^n$ be the block cipher defined by $E_k(m) = \pi(m \oplus k_1) \oplus k_2$ where $k = k_1 \| k_2 \in \{0,1\}^{2n}$ is the key, split into two n-bit blocks.

1. Define the decryption algorithm for E.
2. Let m_1, m_2 such that $m_1 \oplus m_2 = k_1$. Prove that $E_k(m_1) \oplus \pi(m_1) = E_k(m_2) \oplus \pi(m_2)$.
3. We want to describe an attack along the following lines: The adversary samples q messages $m_1, \dots, m_q$ and queries $E_k(m_i)$ for all i, and compute $\pi(m_i)$ for all i; For each *collision* $E_k(m_i) \oplus \pi(m_i) = E_k(m_j) \oplus \pi(m_j)$, the adversary computes a tentative key k' and checks whether it is correct. The goal is to understand how to compute the tentative key, and how to use it afterwards.
   - **i.** What is the probability that there exist $i \neq j$ such that $m_i = m_j \oplus k_1$?
   - **ii.** Let m_i and m_j provoking a collision: how can the adversary compute a tentative (full) key k'?
   - **iii.** How can the adversary check that the tentative key is (probably) the correct one?
4. We now turn this *key-recovery attack* into a significant advantage in the PRP game. Recall that the adversary has access to an oracle $\mathcal{O}$ and must distinguish between the cases where $\mathcal{O} \twoheadleftarrow \mathrm{Perm}_n$ is a random permutation and $\mathcal{O} = E_k$ where $k \twoheadleftarrow \{0,1\}^{2n}$.
   - **i.** Turn the key-recovery attack into an adversary that makes q queries to the oracle, and tries to distinguish between the cases. *You must describe the queries the adversary makes, the additional computations and the answer given depending on the results.*
   - **ii.** Assume that $\mathcal{O} \twoheadleftarrow \mathrm{Perm}_n$. Explain what must happen for the adversary to be fooled (that is for the adversary to answer that $\mathcal{O} = E_k$), and justify that this happens with very low probability.
   - **iii.** Provide a value for q such that the adversary has a constant nonzero advantage in the PRP game.

Exercise 3 (10 points). *Merkle-Hellman encryption scheme*

In the *knapsack problem*, the inputs are a tuple of positive integers $a = (a_0, \ldots, a_{n-1})$ and a integer *target* T. The output is a tuple $(x_0, \ldots, x_{n-1}) \in \{0,1\}^n$ such that $\sum_{i=0}^{n-1} a_i x_i = T$, if it exists. An algorithm for the problem is to test all 2^n tuples $(x_0, \ldots, x_{n-1}) \in \{0,1\}^n$ and check whether $\sum_{i=0}^{n-1} a_i x_i = T$.

An input tuple $(a_0, \ldots, a_{n-1})$ is said *super-increasing* if for $0 < i < n$, $\sum_{j=0}^{i-1} a_j < a_i$.

1. Let $a = (1, 2, 5, 10, 20, 50, 100)$ a super-increasing tuple, and $T = 73$. Give a solution to the knapsack problem.
2. Let a be a super-increasing tuple, T be the target, and let x be a solution.
   i. Let i be the largest index such that $a_i \leq T$. Prove that $x_i = 1$.
   ii. Describe an efficient and simple algorithm for the knapsack problem when a is super-increasing.
   iii. Prove that when a is super-increasing, there exists at most one solution.

Merkle-Hellman encryption scheme is based on the knapsack problem. The key generation algorithm is as follows:

$\textsc{Gen}_n()$:

1 Choose a super-increasing tuple $(b_0, \ldots, b_{n-1})$ and $u > \sum_{i=0}^{n-1} b_i$
2 $v \twoheadleftarrow \mathbb{Z}/u\mathbb{Z}$ such that $\textsc{gcd}(u, v) = 1$
3 $w \leftarrow v^{-1} \bmod u$
4 for $i = 0$ to $n-1$: $a_i \leftarrow w \cdot b_i \bmod u$
5 return $pk = a$ and $sk = (u, v, b)$.

The encryption a message $m \in \{0,1\}^n$ is $\mathsf{Enc}_{pk}(m) = \sum_{i=0}^{n-1} m_i a_i$.

3. Why is it legitimate to compute $v^{-1} \bmod u$ in the key generation algorithm? Which algorithm is used for this computation?
4. Describe the decryption algorithm $\mathsf{Dec}_{sk}(c)$ and prove the correction: if $c \leftarrow \mathsf{Enc}_{pk}(m)$, then $\mathsf{Dec}_{sk}(c)$ outputs m.
5. Explain why Merkle-Hellman encryption cannot be IND-CPA secure.[1] Exhibit an adversary that breaks the IND-CPA security of this encryption scheme.
6. We prove that the algorithm sketched in the introduction to solve the knapsack problem can be significantly improved using the technique *meet-in-the-middle*. Let $t = \lfloor n/2 \rfloor$ and let $\mathcal{L} = \left\{ \sum_{i=0}^{t-1} a_i x_i : (x_0, \ldots, x_{t-1}) \in \{0,1\}^t \right\}$ and $\mathcal{R} = \left\{ \sum_{i=t}^{n-1} a_i x_i : (x_t, \ldots, x_{n-1}) \in \{0,1\}^{n-t} \right\}$.
   i. Explain how to find a solution to the problem, using $\mathcal{L}$ and $\mathcal{R}$.
   ii. Describe an algorithm for the knapsack problem that has complexity $O(n \cdot 2^{n/2})$. *Note. The algorithm must be fully specified and the complexity proved. In particular, explain how the idea of the previous question can be implemented within the time bound.*

[1] This cryptosystem is actually known to be fully broken (one can compute m from c using only the public key). The question asks for an explanation that does not use this fact: Prove that it would not be IND-CPA secure even if computing m from c using only the public key was really difficult.