Examen final – 25 avril 2025

Aucun document n'est autorisé. Sauf indication contraire, les réponses doivent être soigneusement justifiées pour obtenir tous les points. Les exercices sont indépendants, mais pas toutes les questions. Vous pouvez admettre un résultat d'une question précédente en l'indiquant clairement. Vous pouvez répondre en anglais ou en français.

Il y a trois exercices, ne pas oublier de tourner la page !

Exercice 1 (sur 4 pts). *Authentification*

1. Quelle est la différence entre un adversaire *passif* et un adversaire *actif* ? De quel type d'adversaire l'authentification permet-elle de se prémunir ?
2. Décrire les similarités et les différences entre un MAC et un schéma de signature.
3. Bob veut envoyer un message authentifié à Alice, en utilisant un schéma de signature. Quelle(s) clef(s) Bob doit-il utilisé pour authentifier le message ? Quelle(s) clef(s) Alice doit-elle utiliser pour vérifier l'authenticité du message ? *Dans les deux cas, préciser le type de clef et son propriétaire.*
4. Décrire le paradigme *Hash-and-sign* (« hacher et signer ») pour concevoir un schéma de signature. Quel est son but ? Quelle hypothèse de sécurité sur la fonction de hachage est nécessaire pour éviter une forgerie existentielle ?

Exercice 2 (sur 9 pts). *Construction d'Even-Mansour*

Soit $\pi : \{0,1\}^n \to \{0,1\}^n$ une permutation publique (aléatoire) : tout le monde est capable d'appliquer π *ou son inverse* à tout $x \in \{0,1\}^n$. Soit $E : \{0,1\}^{2n} \times \{0,1\}^n \to \{0,1\}^n$ un chiffre par bloc défini par $E_k(m) = \pi(m \oplus k_1) \oplus k_2$ où $k = k_1 \| k_2 \in \{0,1\}^{2n}$ est la clef, coupée en deux blocs de n bits.

1. Décrire l'algorithme de déchiffrement pour E.
2. Soit m_1 et m_2 tels que $m_1 \oplus m_2 = k_1$. Montrer que $E_k(m_1) \oplus \pi(m_1) = E_k(m_2) \oplus \pi(m_2)$.
3. On veut décrire une attaque selon les grandes lignes suivantes : l'adversaire tire aléatoirement q messages $m_1, \ldots, m_q$ et fait une requête $E_k(m_i)$ pour tout i ; pour chaque *collision* $E_k(m_i) \oplus \pi(m_i) = E_k(m_j) \oplus \pi(m_j)$, l'adversaire calcile une clef potentielle k' et vérifie si elle est correcte. Le but est de comprendre comment calculer la clef potentielle, et comment l'utiliser ensuite.
   - **i.** Quelle est la probabilité qu'il existe $i \neq j$ tels que $m_i = m_j \oplus k_1$?
   - **ii.** Soit m_i et m_j provoquant une collision : comment l'adversaire peut-il calculer une clef potentielle (complète) k' ?
   - **iii.** Comment l'adversaire peut-il vérifier que la clef potentielle est (probablement) correcte ?
4. On transforme maintenant cette *attaque par recouvrement de clef* en un avantage significatif dans le jeu PRP. Rappelons que l'adversaire a accès à un oracle $\mathcal{O}$ et doit distinguer entre les deux cas où $\mathcal{O} \twoheadleftarrow \mathrm{Perm}_n$ est une permutation aléatoire et $\mathcal{O} = E_k$ où $k \twoheadleftarrow \{0,1\}^{2n}$.
   - **i.** Transofomer l'attaqe par recouvrement de clef en un adversaire qui effectue q requêtes à l'oracle, et tente de distinguer entre les deux cas. *Vous devez décrire les requêtes que l'adversaire effectue, les calculs additionnels et la réponse en fonction des résultats.*
   - **ii.** Supposons que $\mathcal{O} \twoheadleftarrow \mathrm{Perm}_n$. Expliquer ce qui doit arriver pour que l'adversaire soit trompé (c'est-à-dire pour qu'il réponde que $\mathcal{O} = E_k$), et justifier que cela n'arrive qu'avec une très faible probabilité.
   - **iii.** Fournir une valeur de q telle que l'adversaire a un avantage constant non nul dans le jeu PRP.

Exercice 3 (sur 10 pts). *Schéma de chiffrement de Merkle-Hellman*

Dans le *problème du sac-à-dos*, les entrées sont un n-uplet d'entiers strictement positifs $a = (a_0, \ldots, a_{n-1})$ et un entier *cible* T. La sortie est un n-uplet $(x_0, \ldots, x_{n-1}) \in \{0,1\}^n$ tel que $\sum_{i=0}^{n-1} a_i x_i = T$, s'il en existe un. Un algorithm pour le problème consiste à tester tous les 2^n n-uplets $(x_0, \ldots, x_{n-1}) \in \{0,1\}^n$ et de vérifier si $\sum_{i=0}^{n-1} a_i x_i = T$.

Un n-uplet d'entrée $(a_0, \ldots, a_{n-1})$ est di *super-croissant* si pour $0 < i < n$, $\sum_{j=0}^{i-1} a_j < a_i$.

1. Soit $a = (1, 2, 5, 10, 20, 50, 100)$ un n-uplet super-croissant, et $T = 73$. Donner une solution de ce problème de sac-à-dos.
2. Soit a un n-uplet super-croissant, T la cible, et x une solution.
   - **i.** Soit i le plus grand indice tel que $a_i \leq T$. Montrer que $x_i = 1$.
   - **ii.** Décrire un algorithme efficace et simple pour le problème du sac-à-dos lorsque a est super-croissant.
   - **iii.** Montrer que lorsque a est super-croissant, il existe au plus une solution.

Le schéma de chiffrement de Merkle-Hellman est basé sur le problème du sac-à-dos. L'algorithme de génération de clefs est le suivant :

$\text{GEN}_n()$:

1 Choisir un n-uplet super-croissant $(b_0, \ldots, b_{n-1})$ et $u > \sum_{i=0}^{n-1} b_i$
2 $v \twoheadleftarrow \mathbb{Z}/u\mathbb{Z}$ tel que $\text{GCD}(u, v) = 1$
3 $w \leftarrow v^{-1} \bmod u$
4 pour $i = 0$ à $n-1$: $a_i \leftarrow w \cdot b_i \bmod u$
5 renvoyer $pk = a$ et $sk = (u, v, b)$

Le chiffrement d'un message $m \in \{0,1\}^n$ est $\mathsf{Enc}_{pk}(m) = \sum_{i=0}^{n-1} m_i a_i$.

3. Pourquoi est-il légitime de calculer $v^{-1} \bmod u$ dans l'algorithme de génération de clefs ? Quel algorithme utilise-t-on pour ce calcul ?
4. Décrire l'algorithme de déchiffrement $\mathsf{Dec}_{sk}(c)$ et prouver la correction : si $c \leftarrow \mathsf{Enc}_{pk}(m)$, alors $\mathsf{Dec}_{sk}(c)$ renvoie m.
5. Expliquer pourquoi le chiffrement de Merkle-Hellman ne peut pas avoir la sécurité IND-CPA [1]. Exhiber un adversaire qui casse la sécurité IND-CPA de ce schéma de chiffrement.
6. On prouve que l'algorithme esquissé en introduction pour résoudre le problème du sac-à-dos peut être significativement amélioré à l'aide de la technique *couper la poire en deux* [2]. Soit $t = \lfloor n/2 \rfloor$ et soit $\mathcal{L} = \left\{ \sum_{i=0}^{t-1} a_i x_i : (x_0, \ldots, x_{t-1}) \in \{0,1\}^t \right\}$ et $\mathcal{R} = \left\{ \sum_{i=t}^{n-1} a_i x_i : (x_t, \ldots, x_{n-1}) \in \{0,1\}^n \right\}$.
   - **i.** Expliquer comment trouver une solution au problème, en utilisant $\mathcal{L}$ et $\mathcal{R}$.
   - **ii.** Décrire un algorithme pour le problème du sac-à-dos, de complexité $O(n \cdot 2^{n/2})$. *Note. L'algorithme doit être entièrement spécifié et la complexité prouvée. En particulier, expliquer comment l'idée de la question précédente peut être implantée avec la complexité voulue.*

1. Ce cryptosystème est en réalité totalement cassé (il est possible de calculer m à partir de c en utilisant uniquement la clef publique. La question demande une explication qui n'utilise pas ce fait : prouver qu'il serait impossible d'avoir la sécurité IND-CPA même si calculer m depuis c avec uniquement la clef publique était vraiment difficile.
2. Tentative de traduction de *meet-in-the-middle*.