**IF202 CRYPTOLOGIE — TD partie 2 — 2020-2021**

## Exercice 1 (Cryptosystème de Merkle et Hellman)

Le but de cet exercice est d'étudier un cryptosystème à clé publique fondé sur la difficulté d'un problème complet de la classe *NP*. Le cryptosystème repose sur le problème SUBSETSUM (somme de sous ensembles) qui est *NP*-complet :

SUBSETSUM

*entrée* : Une suite $\sigma$ de $n$ nombres entiers $a_1, \ldots, a_n$ et un nombre $S$.

*sortie* : OUI s'il existe un sous-ensemble des nombres dont la somme est $S$, NON sinon

On s'intéresse à la version fonctionnelle, qui a les mêmes entrées mais qui demande en sortie un certificat, c'est-à-dire un sous ensemble des $a_i$ dont la somme vaut $S$ s'il existe. Ce sous ensemble peut être encodé par un mot $b_1 \ldots b_n$ binaire de longueur $n$.

1. Soit $\sigma = (1, 5, 6, 11, 14, 20)$. Existe-t-il une solution pour les entrées $\sigma, 22$ et $\sigma, 24$ ?
2. Supposons que pour un certain ensemble d'instances, le problème SUBSETSUM est facile à condition de connaître une information supplémentaire. Plus précisément, on fait l'hypothèse suivante :

   il existe $E$, ensemble de suites d'entiers, $B$ ensemble de « brèches » et une fonction $f : E \to B$ qui associe à chaque suite de $E$ une brèche $b$ tels que le problème

   *entrée* $\sigma \in E, b = f(\sigma)$ et $S$ entier

   *sortie* un sous-ensemble de $\sigma$ dont la somme des éléments vaut $S$ s'il en existe

   est facile.

   En déduire un cryptosystème à clés publiques.
3. Une suite $a_1, \ldots, a_n$ est *super croissante* si pour tout $i, 1 < i \leq n$, $\sum_{j=1}^{i-1} a_j < a_i$. Montrer qu'il existe un algorithme polynomial qui résout le problème SUBSETSUM pour toute entrée $(\sigma, S)$ où $\sigma$ est une suite super croissante.
4. Dans le cryptosystème de Merkle et Hellman, la clé privée est une suite super croissante $\sigma$, et deux entiers $q$ et $N$ tels que $N$ est plus grand que la somme des éléments de $\sigma$ et $q$ est premier avec $N$. La clé publique est la suite obtenue en multipliant chaque élément de $\sigma$ par $q$ modulo $N$.

   Quels sont les algorithmes de chiffrement et déchiffrement ? Soit $\sigma = \{2, 3, 6, 13, 27, 52\}$, $q = 31$ et $N = 105$. Chiffrer le message 011000 110101. Déchiffrer le message 280 333.
5. Si vous deviez implémenter ce cryptosystème, quelle taille (en bits) préconiseriez-vous pour la clé privée ?

Malheureusement, il existe des failles dans la transformation de la clé privée vers la clé publique qui permettent de retrouver la clé privée à partir de la clé publique ou de se passer de celle ci pour retrouver le message en clair. L'étude de cette attaque dépasse le cadre de ce TD. Voir Adi Shamir *A polynomial-time algorithm for breaking the basic Merkle - Hellman cryptosystem*, Information Theory, IEEE Transactions on, vol.30, no.5, pp. 699-704, Sep 1984. À ce jour, la cryptologie à clé publique ne repose pas sur l'hypothèse de la difficulté des problèmes *NP*-complet, mais sur la difficulté (supposée) de certains problèmes qui ont trait à la théorie des nombres. Néanmoins, le cryptosystème fde Merkle et Hellman a un intérêt historique puisqu'il fût l'un des premiers cryptosystèmes à clé publique publiés.

## Exercice 2 (Racine carrée modulaire, factorisation et cryptosystème de Rabin)

On note ($|p|$ est le nombre de bits pour encoder $p$)
— $C = \{N : \exists p, q \text{ premiers }, |p| = |q|, pq = N\}$
— $\mathbb{Z}_N = \{i : 0 \leq i \leq N-1\}$
— $\mathbb{Z}_N^* = \{i : 1 \leq i \leq N-1 : pgcd(i, N) = 1\}$
— $Q_N = \{y \in \mathbb{Z}_N^* : \exists x \in \mathbb{Z}_N^*, x^2 = y\}$

1. Soit $p$ un nombre premier et $y \in Q_p$. Combien existe-t-il de racines carrés modulo $p$ de $y$, c'est-à-dire quel est le cardinal de $\{x \in \mathbb{Z}_p^* : x^2 = y \mod p\}$ ? On pourra se rappeler que $\mathbb{Z}_p^*$ est un groupe cyclique et utiliser le théorème d'Euler.
2. Donnez un algorithme efficace (de complexité polynomiale) pour le problème suivant :

   RAC : entrée : $p$ premier tel que $p = 3 \mod 4$, $y \in Q_p$
   sortie : $\{x \in \mathbb{Z}_p^* : x^2 = y \mod p\}$

Soit $N = pq$ avec $p, q$ premiers entre eux. On rappelle que $\varphi : x \to ([x \mod p], [x \mod q])$ est un isomorphisme de $\mathbb{Z}_N^*$ vers $\mathbb{Z}_p^* \times \mathbb{Z}_q^*$, et de $\mathbb{Z}_N$ vers $\mathbb{Z}_p \times \mathbb{Z}_q$.

3. Que valent $\varphi^{-1}(0, 1)$ et $\varphi^{-1}(1, 0)$ ? Plus généralement, comment calculer $\varphi^{-1}(x)$, pour $x \in \mathbb{Z}_p \times \mathbb{Z}_q$ ?
4. Démontrer que si la factorisation $(p, q)$ de l'entrée $N$ est connue, et $p, q = 3 \mod 4$, le problème suivant admet un algorithme de complexité polynomiale :

   4RAC : entrée : $N \in C$, $y \in Q_N$
   sortie : $\{x \in \mathbb{Z}_N^* : x^2 = y \mod N\}$

5. Réciproquement, démontrer que le problème FACTORISATION défini ci-dessous se réduit polynomialement à 4RAC

   FACTORISATION : entrée : $N \in C$
   sortie : $p, q > 1$ tels que $N = pq$

6. Définir une fonction à sens unique et à brèche secrète issue du problème 4RAC ; en déduire un cryptosystème (dit de Rabin) à clés publiques.
7. Montrer qu'une attaque à texte chiffré est polynomialement équivalente à FACTORISATION

## Exercice 3 (RSA)

1. Rappeler la fonction à sens unique à brèche secrète issue du problème RSA puis en déduire un cryptosystème (appelé le cryptosystème RSA).
   Vous utiliserez le Théorème d'Euler ou l'un de ses corollaires.
2. Démontrer que RACINEIEMEMODULAIRE (appelé aussi problème RSA) est aussi facile que FACTORISATION.
   Vous utiliserez un algorithme en complexité en temps polynomial.

Remarque : Si RACINEIEMEMODULAIRE est aussi facile que FACTORISATION, le contraire est supposé (espéré ?) mais non prouvé !

## Exercice 4 (LogarithmeDiscret)

Soient $p$ un nombre premier et $\alpha$ un générateur de $\mathbb{Z}_p^*$ (c'est-à-dire un entier tel que $\{[\alpha^i \mod p] : 0 \leq i\} = \mathbb{Z}_p^*$). Soit la fonction :

$$\begin{array}{rcl} f : [1, p-2] \times [1, p-2] \times [0, p-1] & \longrightarrow & [0, p-1] \times [0, p-1] \times [0, p-1] \\ (a, k, m) & \longrightarrow & ([\alpha^a \mod p], [\alpha^k \mod p], [m \cdot (\alpha^a)^k \mod p]) \end{array}$$

1. Rappeler la définition des problèmes LogarithmeDiscret et DiffieHellman.
2. En utilisant la fonction $f$, écrire un cryptosystème à clé publique $\Gamma$.
3. Étudier la sécurité de $\Gamma$. Pour cela, démontrer qu'une attaque est aussi difficile que l'un des problèmes mentionnés ci-dessus.

## Exercice 5 (Échanges de clés)

L'objet de cet exercice est d'étudier différents protocoles d'échanges de clés reposant sur la difficulté des problèmes de DiffieHellman, de comparer leurs sécurités et d'étudier notamment s'ils vérifient les propriétés suivantes :

**Fonctionnalité** A la fin du protocole, les deux participants partagent une même clé.

**Rafraîchissement de la clé** Chaque participant est assuré que la clé est nouvelle.

**Autocertification des clés** Chaque participant est assuré que seul le destinataire désiré pourra obtenir la clé de session échangée.

**Authentification des clés** Chaque participant est assuré de l'identité de l'autre participant.

**Confirmation de la clé** Chaque participant confirme à l'autre la possession de la clé.

Pour chacun des protocoles suivants,

1. Écrire en détail les algorithmes ;
2. Étudier leurs sécurités ;
3. Vérifier s'ils respectent ou non les propriétés mentionnées ci-dessus.

On nommera Alice et Bob les deux participants au protocole. On suppose que Bob (resp. Alice) a pour clé privée $a_{Bob}$ (resp. $a_{Alice}$) et pour clé publique $(p, \alpha, y_{Bob})$ (resp. $(p, \alpha, y_{Alice})$). Les couples $(p, \alpha)$ considérés sont formés d'un entier premier $p$ et d'un générateur $\alpha$ de $\mathbb{Z}_p^*$ et sont supposés tels que le problème DiffieHellman est difficile.

**Version 1** Alice envoie à Bob $\mathsf{EncElGamal}\big((p, \alpha, y_{Bob}), K\big)$. La clé échangée est $K$.

**Version 2** Alice envoie à Bob $\alpha^{K_1} \mod p$. Bob envoie à Alice $\alpha^{K_2} \mod p$. La clé échangée est $K := \alpha^{K_1 \cdot K_2} \mod p$.

**Version 3** Alice envoie à Bob $\alpha^{K_1} \mod p$. La clé échangée est $K := y_{Bob}^{K_1} \mod p$.

**Version 4** Alice envoie à Bob $\alpha^{K_1} \mod p$. Bob envoie à Alice $\alpha^{K_2} \mod p$. La clé échangée est $K := \alpha^{K_1 \cdot a_{Bob} + K_2 \cdot a_{Alice}} \mod p$.

**Version 5** Alice envoie à Bob $\alpha^{K_1} \mod p$. Bob envoie à Alice $\alpha^{K_2} \mod p, \mathsf{Enc}_K(S_{Bob}(\alpha^{K_2} \mod p, \alpha^{K_1} \mod p))$ avec $K = \alpha^{K_1 \cdot K_2} \mod p$. Alice envoie à Bob $\mathsf{Enc}_K(S_{Alice}(\alpha^{K_1} \mod p, \alpha^{K_2} \mod p))$. La clé échangée est $K$. $\mathsf{Enc}$ désigne ici la fonction de chiffrement d'un cryptosystème symétrique (par exemple A.E.S) et $S$ est un procédé de signature (par exemple RSA).

## Exercice 6 (Signature d'El Gamal (DSS))

Considérons le procédé de signature d'El Gamal. Alice signe des messages qui sont vérifiés par Bob.

1. Démontrer que Alice doit changer à chaque fois la valeur de $k$.
2. Écrire un problème de contrefaçon associé au procédé de signature, que l'on notera CONTREFAÇON.
3. Démontrer que le problème LOGDISCRET est équivalent au problème :
   *entrées* : $g, \beta, p, H$
   *sorties* : $r, s, m, k$ tels que $g^{H(m)} = \beta^r \cdot r^s \mod p$, $r = g^k \mod p$ et $pgcd(r, p-1) = 1$.
4. Démontrer que CONTREFAÇON se réduit au problème de l'inversion de la fonction de hashage $H$.

**Rappel : Cryptosystème d'El Gamal**

Algorithme de génération de clé GenElGamal

*entrées* : entier $n$
*sorties* : clé publique $(p, g, y)$ et clé secrète $(p, g, x)$ telles que
— $p$ est premier [1] et $||p|| = n$
— $g$ générateur de $\mathbb{Z}_p^*$
— $x \in [1, p-2]$ entier tiré aléatoirement
— $y = [g^x \mod p]$

Algorithme de chiffrement EncElGamal

*entrées* : clé publique $(p, g, y)$, message $m \in \mathbb{Z}_p^*$
*sortie* : message chiffré $(c, c')$
EncElGamal$((p, g, y), m)$
  $k \leftarrow$ entier $\in [1, p-2]$ tiré aléatoirement ;
  $c \leftarrow [g^k \mod p]$ ;
  $c' \leftarrow [m \cdot y^k \mod p]$ ;
  *retourner*$(c, c')$

Algorithme de déchiffrement DecElGamal

*entrées* : message chiffré $(c, c')$, clé secrète $(p, g, x)$
*sortie* : message en clair $m'$
DecElGamal$((p, g, x), (c, c'))$
  $m' \leftarrow [c' \cdot c^{-x} \mod p]$
  *retourner*$(m')$

**Signature d'El Gamal/DSS** Soit $H : \{0,1\}^* \to \mathbb{Z}_{p-1}$ fonction de hashage. Les clés publiques et secrètes $(p, g, y)$ et $(p, g, x)$ sont générées par GenElGamal.

Algorithme de signature Sign

1. On admettra qu'il existe des algorithmes de complexité en temps polynomiale qui retourne un entier premier $p$ de taille $n$ et un générateur $g$ de $\mathbb{Z}_p^*$ où $n$ est un entier donné en entrée. Voir par exemple l'ouvrage de Menezes p 164.

*entrées* : clé secrète $(p, g, x)$, message $m \in \{0,1\}^*$
*sorties* : signature $\sigma$ de $m$
$\mathsf{Sign}((p, g, x), m)$
$k \leftarrow$ entier $\in \mathbb{Z}_{p-1}^*$ tiré aléatoirement ;
$r \leftarrow [[g^k \mod p] \mod (p-1)]$ ;
$s \leftarrow [(H(m) + xr)k^{-1} \mod (p-1)]$ ;
*retourner*$(r, s)$

Algorithme de vérification $\mathsf{Vrfy}$
*entrées* : clé publique $(p, g, y)$, message $m \in \{0,1\}^*$, signature $\sigma = (r, s)$
*sorties* : 1 ou 0
$\mathsf{Vrfy}((p, g, y), m, (r, s))$
$u_1 \leftarrow [H(m)s^{-1} \mod (p-1)]$ ;
$u_2 \leftarrow [rs^{-1} \mod (p-1)]$ ;
**si** $r = [[g^{u_1} y^{u_2} \mod p] \mod (p-1)]$ **alors** *retourner* 1
**sinon** *retourner* 0

## Exercice 7 (Pile ou face)

Alice et Bob veulent jouer à pile ou face mais n'ont pas de pièce à lancer.

1. Alice propose le protocole suivant : Bob choisit aléatoirement un bit, ensuite Alice choisit aléatoirement un bit. Enfin Alice et Bob calculent le ou exclusif des deux bits qui est le résultat du tirage.

   On admettra que le résultat du ou exclusif de deux bits dont seul l'un est aléatoire est un bit aléatoire. Alice et Bob prennent un café. Alice propose d'utiliser ce protocole pour déterminer qui paie les cafés. Bob doit-il accepter ?

2. Après avoir payé les cafés, Bob propose le protocole suivant. Soit $f : \{0,1\}^* \to \{0,1\}^*$.
   (a) Alice choisit un nombre aléatoire $x$ et calcul $y = f(x)$ ;
   (b) Alice envoie $y$ à Bob ;
   (c) Après avoir reçu $y$, Bob choisit un bit $b$ et l'envoie à Alice ;
   (d)
   (e)

   Compléter les deux dernières étapes de ce protocole. Quelles hypothèses doit satisfaire la fonction $f$ pour que ni Alice, ni Bob ne puissent influer sur le résultat du tirage ?

3. Voici un autre protocole qui repose sur l'utilisation d'un cryptosystème « commutatif » :

   $$\forall k1, k2 \in \mathcal{K}, \forall m \in \mathcal{M} : \mathsf{Dec}_{k1}(\mathsf{Enc}_{k2}(\mathsf{Enc}_{k1}(m))) = \mathsf{Enc}_{k2}(m)$$

   Cette propriété n'est en général pas vérifiée par les cryptosystèmes à clé secrète. Elle est par contre vérifiée par le systèmes à clé publique RSA.

   (a) Alice et Bob génèrent une paire (clé publique, clé privé) que l'on notera respectivement $(pa, sa)$ et $(pb, sb)$.
   (b) Alice choisit aléatoirement deux messages $m_0$ et $m_1$, l'un pour pile et l'autre pour face.

(c) Alice chiffre $m_0$ et $m_1$ avec sa clé publique et envoie à Bob le résultat $\mathsf{Enc}_{pa}(m_0), \mathsf{Enc}_{pa}(m_1)$.

(d) Après avoir reçu ces deux messages, Bob choisit l'un d'entre eux noté $c$, le chiffre avec sa clé publique et envoie à Alice le résultat $\mathsf{Enc}_{pb}(c)$.

(e) Alice déchiffre ce message avec sa clé privée et envoie à Bob le résultat $\mathsf{Dec}_{sa}(\mathsf{Enc}_{pb}(c)) = \mathsf{Dec}_{sa}(\mathsf{Enc}_{pb}(\mathsf{Enc}_{pa}(m_i))) = \mathsf{Enc}_{pb}(m_i)$ pour un certain $i \in \{0, 1\}$.

(f) Bob déchiffre le message et envoie le résultat $m_i$ à Alice.

(g) Alice vérifie que $m_i \in \{m_0, m_1\}$ et lit le résultat du lancer.

(h) Alice et Bob révèlent leurs clés secrètes.

Ce protocole est à sécurité intrinsèque : aucune entité extérieure n'est nécessaire pour effectuer le protocole ou arbitrer les litiges. Montrer que chaque participant détecte immédiatement tout tentative de tricherie.

4. Plus tard, Alice et Bob rencontrent Carole qui leur propose de jouer au poker. Adapter le protocole précédent pour leur permettre de jouer par courrier électronique.

## Exercice 8 (Preuve à divulgation nulle [2])

Paul prétend avoir résolu la grille de sudoku #2349 du journal l'Univers. Un problème de sudoku se présente sous la forme d'une grille de dimension $9 \times 9$, dont certains cases sont pré-remplies avec des entiers entre 1 et 9. Une solution au problème est l'attribution d'un entier $\in \{1, \ldots, 9\}$ à chaque case non pré-remplie de telle sorte que chaque ligne, colonne ou sous grille $3 \times 3$ ne comporte pas deux fois le même entier. Victoria a néanmoins quelques doutes et souhaiterait que Paul lui montre sa solution afin de pouvoir vérifier ses dires. Au lieu de cela, Paul propose d'effectuer le protocole suivant :

— **Paul** dessine au sol une grille de taille $9 \times 9$. Sur chaque case, il dépose 3 cartes. Si la case correspond à une case pré-remplie du problème, les trois cartes sont déposées faces ouvertes ; leur valeur est celle indiquée dans la case correspondantes. Sinon, les trois cartes sont déposées faces cachées.

— **Victoria** Pour chaque ligne/colonne/sous-grille, Victoria choisit (aléatoirement) une des trois cartes de chaque case dans la ligne/colonne/sous-grille correspondante.

— **Paul** rassemble les cartes choisies par Victoria en tas : un tas par ligne/colonne/sous-grille. Il mélange ensuite chacun des tas séparément.

— **Victoria** récupère les tas et vérifie qu'aucun d'entre eux ne contient deux cartes de même hauteur. Si c'est le cas, Victoria **accepte** l'affirmation de Paul et **rejette** sinon.

1. Montrer que si Paul a effectivement résolu la grille, et que Paul et Victoria suivent le protocole, Victoria **accepte** toujours. Le système de preuve est dit *consistant*.
2. Supposer que Paul ne connaît pas la solution du problème. Montrer que Victoria accepte avec probabilité au plus 1/3, même si Paul ne respecte pas le protocole. Bonus : montrer que cette probabilité est en fait au plus 1/9. Le protocole est dit *robuste* : Victoria rejette avec une probabilité non nul les assertions fausses.
3. En déduire un protocole dans lequel Victoria accepte avec probabilité $< 1/3^{10}$ lorsque Paul ne connaît pas la solution.

2. L'exercice est tirée de R. Gradwohl, M. Naor, B. Pinkas and, G. Rothblum, *Cryptographic and Physical Zero-knowledge Proof Systems for Solutions of Sudoku Puzzles.* Theory Comput. Syst. 44(2) : 245-268 (2009) `http://www.wisdom.weizmann.ac.il/~naor/PAPERS/sudoku_abs.html`

4. Démontrer que le système de preuve est *à divulgation nulle* : le protocole ne révèle rien sur la solution du problème, même si Victoria ne respecte pas sa partie du protocole.

Un procédé de mise en gage est la donnée d'un algorithme $\mathsf{Commit}$ qui prend en paramètre des couples $\in \mathcal{M} \times \mathcal{K}$ et vérifie les propriétés

— *Engagement* Il n'existe pas $m \neq m' \in \mathcal{M}, k, k' \in \mathcal{K}$ tels que $\mathsf{Commit}(m,k) = \mathsf{Commit}(m',k')$.
— *Dissimulation* Étant donné $\mathsf{Commit}(m,k)$, le problème de calculer $m$ (ou toute fonction non triviale $f(m)$) est difficile.

5. Supposer l'existence de procédés d'engagement. Donner un protocole de preuve à divulgation nulle pour le problème du sudoku qui repose sur de tels procédés plutôt que sur la manipulation de paquets de cartes.
6. Le problème sudoku est NP-complet [3]. Paul prétend qu'un graphe $G$ est 3-coloriable. Peut-il le démonter à Victoria sans indiquer comment le 3-colorier ?

3. Takayuki Yato, Complexity and Completeness of Finding Another Solution and its Application to Puzzles, Master thesis, U. of Tokyo, Jan 2003 `http://www-imai.is.s.u-tokyo.ac.jp/~yato/data2/MasterThesis.ps`