École normale supérieure **2018-2019**
Département d'informatique

Introduction à la cryptologie
TD n° 10 : Correction.

Exercice 1 (Calcul de produit à n joueurs)**.**

1. Soit x la clef secrète d'un chiffrement ElGamal, sur un groupe $\mathbb{G}$ généré par g. Soient (g^r, mg^{xr}) et $(g^{r'}, m'g^{xr'})$ deux chiffrements ElGamal. Alors le produit coordonnée par coordonnée des deux chiffrements est égal à $(g^{r+r'}, mm'g^{x(r+r')})$. C'est un chiffrement valide du message mm'.
2. Les n parties peuvent utiliser le protocole suivant.
   — Les parties génèrent de manière distribuée une paire clef publique/clef secrète ElGamal (voir slides du cours sur la cryptographie distribuée). La clef secrète est donc partagée entre les n parties.
   — Chaque participant i chiffre son message m_i en utilisant la clef publique du chiffrement (qui, comme son nom l'indique, est publique), et publie ce chiffré.
   — Les n participants calculent le produit des n messages chiffrés, puis déchiffrent ce produit de manière distribuée. Du fait que ElGamal est multiplicativement homomorphe, ils obtiennent le produit des m_i.

Exercice 2 (Transfert inconscient)**.**

1. Par le théorème de Lagrange, $g^p = 1$. Il s'ensuit que g^{p-1} est un inverse de g. En utilisant l'algorithme d'exponentiation rapide, g^{p-1} peut être calculé en temps polynomial en $\log p$. Par ailleurs, on a $g^{p+1} = g$. Sans perte de généralité, on peut supposer $p > 2$, et p est premier, donc $p+1$ est pair. On déduit que $g^{(p+1)/2}$ est la racine carrée de g. Noter que cette racine carrée est unique (une manière de le voir est que $\mathbb{G} \sim (\mathbb{Z}_p, +)$, et la racine carrée correspond à la division par 2, or $p > 2$ est premier donc 2 est premier avec p donc inversible modulo p).
2. Si on a un algorithme efficace pour le problème 1, trivialement on a un algorithme efficace pour le problème 2, en prenant $b = a$. Réciproquement, supposons qu'on a un algorithme $\mathcal{A}$ efficace pour le problème 2. Soit $A = g^a$, $B = g^b$ une instance du problème 1. Alors la réponse au problème 1 est la racine carrée de
$$\mathcal{A}(AB)\mathcal{A}(A)^{-1}\mathcal{A}(B)^{-1} = g^{(a+b)^2}g^{-a^2}g^{-b^2} = g^{2ab}.$$
Comme on sait calculer efficacement $\mathcal{A}$, les racines carrées et les inverses, on a gagné.
3. Soit $\mathcal{A}$ un algorithme qui résout le problème de l'énoncé, *i.e.* $\mathcal{A}(b, g^a, mg^{a^2+ab}) = m$. Alors $A \mapsto \mathcal{A}(0, A, 1)^{-1}$ résoud le problème 2 de l'énoncé, dont on a déjà vu qu'il est équivalent au problème 1, c'est-à-dire le problème de Diffie-Hellman calculatoire (pour g fixé). Réciproquement, si on sait résoudre le problème 1, il est trivial de résoudre le problème de l'énoncé.
4. Voir cours.
5. On suppose qu'Alice a deux messsages m_0 et m_1, et Bob a un bit x. On veut réaliser un transfert inconscient, c'est-à-dire : Bob apprend m_x, mais rien sur m_{1-x}, et Alice n'apprend rien sur x. On peut utiliser le protocole suivant.
   — Alice tire $a \in \mathbb{Z}_p$ uniformément et envoie $A = g^a$.
   — Bob tire $b \in \mathbb{Z}_p$ uniformément et envoie $B = g^b$ si $x = 0$, $B = Ag^b$ si $x = 1$.
   — Alice envoie m_0B^a et $m_1B^aA^{-a}$.

Si $x = 0$, Bob sait calculer $B^a = g^{ab} = A^b$, donc il sait retrouver m_0. Par contre, retrouver m_1 revient à résoudre une instance du problème de la question précédente, dont nous avons vu qu'il est équivalent au problème de Diffie-Hellman calculatoire. Un raisonnement similaire s'applique au cas $x = 1$. Par ailleurs Alice n'apprend rien sur x, puisque g^b et Ag^b sont distribués identiquement (tous deux uniformes dans $\mathbb{G}$).

Exercice 3 (Dîner de cryptologues)**.**

1. Il suffit de calculer la somme modulo 2 des valeurs publiées par les dîneurs. L'un d'entre eux a payé ssi cette somme vaut 1. En effet cette somme est égale à la somme des p_i, puisque tous les autres termes apparaissent deux fois.
2. Soit $P = [\![1, n]\!] \setminus K$ l'ensemble des cryptologues qui ne sont pas cooptés par le KGB. Le but est de montrer que le KGB ne peut rien déduire sur les p_i pour $i \in P$, sauf ce qui est impliqué par l'entrée du protocole connue du KGB, à savoir les p_i pour $i \in K$, et sa sortie, à savoir $s = \sum p_i$. Autrement dit, le protocole ne révèle rien de plus au KGB que ce qui est impliqué par son bon fonctionnement.

   Si P est de taille 0 ou 1, le KGB sait déjà tout avec s et $(p_i)_{i\in K}$, donc la question est triviale. De même, si aucun membre de P n'a payé le repas, le KGB le sait à partir de s et $(p_i)_{i\in K}$, et n'a rien non plus à apprendre. Le cas intéressant est celui où $|P| \geq 2$ et un des membres de P a payé le repas. Dans ce cas le KGB voudrait savoir lequel. Soit $i \neq j \in P$, et supposons que le cryptologue i a payé.

   Alors supposons qu'on effectue la chose suivante : on inverse p_i, p_j, et $x_{i,j}$. Par cette transformation, toutes les informations observées par les membres de K sont inchangées. Cependant, c'est j qui a payé et non plus i. On obtient ainsi une bijection entre les choix des $x_{i,j}$ et $(p_i)_{i\in P}$ compatibles avec ce qu'a observé K, et où i a payé, et ceux où j a payé. Comme la distribution des $x_{i,j}$ est par ailleurs uniforme, on déduit que la probabilité que i a payé, ou que j a payé, conditionnée aux observations des membres de K, est identique.
3. Non : si un membre non-KGB est assis entre deux membres du KGB, ceux-ci apprennent son p_i. Si $n > 3$ ça ne devrait pas être le cas.
4. Non, le problème de la question précédente est inévitable : si le graphe n'est pas complet, soit s un sommet d'arité inférieure à $n - 1$, alors si les sommets adjacents à s sont contrôlés par le KGB, celui-ci apprend le p_i du sommet s. Or si le KGB ne contrôle pas d'autre sommet, on est dans le cas $|P| \geq 2$, donc le KGB ne devrait pas pouvoir déduire la valeur de ce p_i.

Exercice 4 (Un peu de complexité pour conclure)**.**
Soit $\mathcal{C}$ un schéma de chiffrement à clef publique sûr. On considère le problème de décision $P(x)$: est-ce que le dernier bit de la clef secrète correspondant à la clef publique x est 1 ? Ce problème est dans NP : en effet, si x est effectivement une clef publique valide de $\mathcal{C}$, et que la clef secrète s correspondante termine par 1, il existe un algorithme polynomial $\mathcal{A}$ qui vérifie ce fait en utilisant un témoin $t(x)$. Pour $t(x)$, on peut prendre les bits aléatoires utilisés dans l'algorithme de génération de clef de $\mathcal{C}$ et qui aboutissent à la paire clef publique/clef secrète (x, s). Pour $\mathcal{A}(t(x))$, il suffit de faire tourner l'algorithme de génération de clef avec l'aléa $t(x)$ pour obtenir une paire (x', s'), vérifier $x = x'$, et vérifier que le dernier bit de s est 1. Si P = NP, on peut décider le problème P en temps polynomial, donc on sait retrouver en temps polynomial le dernier bit de la clef secrète à partir de la clef publique. On peut répéter le même raisonnement pour tous les autres bits ; comme il y a un nombre polynomial de bits dans la clef secrète, tous les retrouver continue à ne coûter qu'un temps polynomial. Conclusion : comme on ne sait pas prouver P $\neq$ NP, on ne sait pas non plus s'il existe un chiffrement à clef publique sûr. Plus généralement si P = NP la cryptographie s'effondre, en théorie, et aussi en pratique si l'algorithme polynomial qui provient de l'égalité est efficace (à peu près toutes les communications deviennent non sûres, chaos international à suivre).