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

Introduction à la cryptologie
TD n° 4 : Correction.

Exercice 1 (Isomorphisme de graphes)**.** On suppose $G_1 = \pi(G_0)$. On utilise le protocole suivant.

1. Le prouveur tire un isomorphisme ϕ uniformément aléatoire de G_0 (un tel isomorphisme revient à une permutation des labels des sommets du graphe, il suffit donc de tirer une permutation uniforme), et pose $H = \phi(G_0)$. Le prouveur envoie H au vérifieur.
2. Le vérifieur tire $b \leftarrow \{0, 1\}$ uniformément aléatoirement, et envoie b au prouveur.
3. Le prouveur répond en envoyant $\psi = \phi \circ \pi^{-b}$.
4. Le vérifieur accepte ssi $H = \psi G_b$.

Il s'agit ci-dessus d'une version condensée du protocole. En réalité cette version condensée est itérée λ fois, pour un certain paramètre de sécurité λ. Le vérifieur accepte ssi $H = \psi G_b$ à toutes les itérations.
Completeness. On vérifie aisément que si les deux parties sont honnêtes, le vérifieur accepte.
Soundness. Comme il s'agit d'une preuve de connaissance (et non simplement d'appartenance à un langage), on veut montrer qu'il existe un extracteur. On rappelle que l'extracteur contrôle l'aléa du prouveur. Il peut donc le forcer à répondre à deux challenges b différents pour le même H. Il obtient ψ_0 tel que $H = \psi_0 G_0$ et ψ_1 tel que $H = \psi_1 G$. Puisqu'il a un isomorphisme entre H et G_0, et entre H et G_1, il déduit un isomorphisme $\pi = \psi_1^{-1} \circ \psi_0$ entre G_0 et G_1. (On voit qu'on a même la propriété de *special soundness* vue en cours.)
Zero-knowledge. Le transcript (i.e. la séquence de toutes les communications échangées entre les parties) d'une exécution honnête du protocole peut être facilement simulée. En effet, il suffit de choisir b d'abord, puis de prendre $H = \phi G_b$ pour un ϕ uniformément aléatoire, et enfin de fixer $\psi = \phi$. La distribution ainsi obtenue est identique à la distribution d'une exécution honnête du protocole. En effet, pour ϕ uniforme, la distribution de ϕG_0 et ϕG_1 est identique. (On remarque que l'hypothèse d'honnêteté est nécessaire : si un vérifieur malhonnête choisit b en fonction de H, la preuve ne tient plus.)

Exercice 2 (Non-isomorphisme de graphes)**.**

1. Le *vérifieur* tire $b \leftarrow \{0, 1\}$ uniformément aléatoirement. Il tire ensuite un isomorphisme ϕ uniformément aléatoire de G_b, et pose $H = \phi(G_b)$. Le vérifieur envoie H au prouveur.
2. Le prouveur utilise sa puissance de calcul infinie pour déterminer si H est isomorphe à G_0 ou G_1. Si G_0 et G_1 ne sont pas isomorphes, un seul des deux cas est possible. Soit b' tel que $G_{b'} \sim H$. Le prouveur envoie b' au vérifieur.
3. Le vérifieur accepte ssi $b' = b$.

Il s'agit ci-dessus d'une version condensée du protocole. En réalité cette version condensée est itérée λ fois, pour un certain paramètre de sécurité λ. Le vérifieur accepte ssi $b' = b$ à toutes les itérations.
Completeness. On vérifie aisément que si les deux parties sont honnêtes, le vérifieur accepte.
Soundness. Cette fois il s'agit d'une preuve d'*appartenance* à un langage (le langage des paires de graphes non-isomorphes), non d'un preuve de *connaissance*. Au lieu de construire un extracteur, on va donc montrer que si le vérifieur accepte, alors G_0 et G_1 sont bien non-isomorphes, sauf avec une probabilité $2^{-\lambda}$. Comme dans l'exercice précédent, le point crucial est que si G_0 et G_1 sont isomorphes, la distribution de ϕG_0 et ϕG_1 pour ϕ uniforme est identique, donc le prouveur n'apprend rien sur b en obtenant H. La probabilité que $b' = b$ est donc $1/2$. La probabilité que le prouveur accepte après λ itérations indépendantes est donc $2^{-\lambda}$.
Zero-knowledge. Le transcript d'une interaction honnête est simulable trivialement puisqu'il suffit d'exécuter le protocole du vérifieur et de fixer $b' = b$. (Si le vérifieur est malhonnête par contre, il peut apprendre si un graphe arbitraire est isomorphe à G_0 ou à G_1, ce qui n'est pas a priori un calcul qu'il saurait faire par lui-même.)

Exercice 3 (Non-résiduosité quadratique)**.** On utilise le protocole suivant. Soit $N = pq$ la factorisation non-triviale de N.

1. Le *vérifieur* tire $b \leftarrow \{0, 1\}$ uniformément aléatoirement. Il tire ensuite un entier aléatoire y modulo N et envoie au prouveur $z = x^b y^2 \bmod N$.

2. Le prouveur utilise sa connaissance de la factorisation de N pour calculer si z est un carré modulo N (en utilisant le symbole de Jacobi). Si c'est le cas il renvoie $b' = 0$, sinon il renvoie $b' = 1$.
3. Le vérifieur accepte ssi $b' = b$.

Il s'agit ci-dessus d'une version condensée du protocole. En réalité cette version condensée est itérée λ fois, pour un certain paramètre de sécurité λ. Le vérifieur accepte ssi $b' = b$ à toutes les itérations.
Par ailleurs, on va supposer que le nombre $0 < y < N$ tiré par le vérifieur est toujours premier avec N, sans quoi la vérifieur a réussi à factoriser N (et peut notamment vérifier tout seul si x est un carré ; plus généralement il a autant d'information que le prouveur). On suppose également que x est premier avec N.
Completeness. On vérifie aisément que si les deux parties sont honnêtes, le vérifieur accepte. En effet, le groupe multiplicatif des éléments de $\mathbb{Z}_N$ premiers avec N est isomorphe par le théorème des restes chinois à $(\mathbb{Z}_p, \times) \times (\mathbb{Z}_q, \times) \sim (\mathbb{Z}_{p-1}, +) \times (\mathbb{Z}_{q-1}, +)$. On déduit que le produit de x^b avec y^2 est un carré ssi x^b est un carré ssi $b = 0$.
Soundness. Si x est un carré, soit r tel que $x = r^2$. Comme y est tiré uniformément, et que r est inversible (on est dans le groupe multiplicatif de $\mathbb{Z}_N$) ; ry est également uniforme, donc y^2 et $xy^2 = (ry)^2$ ont la même distribution, donc la donnée de $x^b y^2$ n'apprend rien au prouveur sur b. Il s'ensuite que si x est un carré, la probabilité que $b' = b$ est $1/2$. Après λ itérations la probabilité que le vérifieur accepte est donc $2^{-\lambda}$ (quels que soient par ailleurs les choix du prouveur).
Zero-knowledge. Le transcript d'une interaction honnête est simulable trivialement puisqu'il suffit d'exécuter le protocole du vérifieur et de fixer $b' = b$. (Si le vérifieur est malhonnête par contre, il peut apprendre si un élément arbitraire de $\mathbb{Z}_N$ est un carré, ce qui n'est pas a priori un calcul qu'il saurait faire par lui-même.)

Exercice 4 (Preuve de connaissance d'un logarithme discret)**.**
Completeness. On vérifie aisément que si les deux parties sont honnêtes, le vérifieur accepte.
Soundness. Il s'agit d'un preuve de *connaissance*. On veut donc prouver l'existence d'un extracteur. Comme souvent, il suffit à l'extracteur d'interagir avec le prouveur (Alice) en fixant son aléa (donc en fixant r) et en lui demandant de répondre à deux challenges $c \neq c'$. Alice fournit alors s et s' tels que $r = g^s y^c = g^{s'} y^{c'}$. On déduit $g^{s-s'} = y^{c'-c}$, donc $y = g^{(s-s')(c'-c)^{-1}}$. Ici l'inversion est dans $\mathbb{Z}_{q-1}$ (on rappelle que le groupe multiplicatif de $\mathbb{Z}_q$ est cyclique d'ordre $q-1$). On voit au passage qu'on a la propriété de *special soundness* vue en cours.
Zero-knowledge. Pour simuler le transcript d'une interaction honnête (sans connaître x), il suffit de choisir c et s d'abord (uniformément aléatoirement), puis de fixer $r = g^s y^c$. On voit que r est uniformément aléatoire, et on déduit que la distribution de (r, c, s) ainsi obtenue est identique à celle qui provient d'une interaction honnête.
Remarque : cette preuve de connaissance peut être directement transformée en une signature : on obtient le schéma de signature de Schnorr.

Exercice 5 (Preuve de connaissance d'une représentation)**.** On peut s'inspirer du protocole de Schnorr vu dans l'exercice 4 comme suit.

1. Le prouveur tire k, ℓ dans $\mathbb{Z}_q^*$ uniformément aléatoirement. Il envoie au vérifieur $r = g^k h^\ell$.
2. Le vérifieur tire c uniformément aléatoirement dans $\mathbb{Z}_q^*$. Il envoie c au prouveur.
3. Le prouveur calcule $a = k - sc$, $b = \ell - tc$. Il envoie (a, b) au vérifieur.
4. Le vérifieur accepte ssi $r = y^c g^a h^b$.

Completeness. On vérifie aisément que si les deux parties sont honnêtes, le vérifieur accepte. En effet, $y^c g^a h^b = g^{sc+a} h^{tc+b} = g^k h^\ell = r$.
Soundness. On construit un extracteur sembable à celui de l'exercice 4. L'extracteur fixe l'aléa du prouveur, donc fixe r, et demande la réponse à deux challenges $c \neq c'$. Il obtient a, b, a', b' tels que $r = y^c g^a h^b = y^{c'} g^{a'} h^{b'}$. On déduit $y = g^{(a'-a)(c-c')^{-1}} h^{(b'-b)(c-c')^{-1}}$. On a à nouveau la propriété de *special soundness*.
Zero-knowledge. Le transcript d'une interaction honnête peut être simulé (sans connaître s, t) en tirant a, b, c uniformément aléatoirement. On choisit ensuite $r = y^c g^a h^b$. Cet élément est distribué uniformément aléatoirement, et on voit que la distribution du transcript (r, c, a, b) est la même que lors d'une exécution honnête du protocole.

Exercice 6 (Un vote électronique simple)**.**

1. Un chiffrement à clef publique déterministe est généralement problématique, parce que la clef de chiffrement est publique : dans le cas où il y a peu d'entropie dans les messages clairs (ici il n'y a que deux messages possibles !) il suffit à un attaquant de chiffrer lui-même les messages clair les plus probables et de comparer ce qu'il obtient avec les messages chiffrés qui l'intéressent. C'est une des raisons pour lesquelles RSA n'est jamais utilisé dans sa version naïve ; un chiffrement à clef publique est en réalité toujours probabiliste, de manière à ce qu'à un message clair donné correspondent de nombreux messages chiffrés. (Tout cela est d'ailleurs impliqué par les définitions de sécurité dite IND-CCA et IND-CPA classiques des chiffrements à clef publique, qui évitent proprement ce genre de problèmes, et bien d'autres.)
2. Considérons d'abord comment l'autorité pourrait prouver qu'Alice a voté pour le candidat untel, sans révéler aucune autre information. La méthode naïve serait de publier la clef secrète sk, qui permettrait à tout le monde de déchiffrer le vote d'Alice, mais cela révèle bien plus d'information que nécessaire (cette information permettrait par exemple de déchiffrer aussi le message de Bob) : c'est ce qu'on veut éviter. Au lieu de cela, l'idée clef est la suivante : il suffit de montrer que la propriété qu'on souhaite prouver est exprimable par une formule SAT. En effet, vous avez vu en cours que le problème du 3-coloriage admet une preuve à divulgation nulle de connaissance, donc tout problème dans NP en admet une, y compris SAT. (Le problème du 3-coloriage est NP-complet, donc il existe une transformation polynomiale d'une instance SAT en une instance de 3-coloriage équivalente.)

   Construisons un circuit C_1 (par exemple avec des portes ET, OU, NON) qui prend en entrée l'aléa utilisé dans la procédure de génération de clef du chiffrement à clef publique utilisé, et qui donne en sortie les clefs publiques et privées pk et sk. Soit C_2 le circuit de déchiffrement, qui prend en entrée la clef secrète et déchiffre un message. On peut fixer le message à déchiffrer dans C_2 pour obtenir un circuit C_2' qui déchiffre spécifiquement le message chiffré publié par Alice. Ensuite, on peut combiner les circuits C_1 et C_2' de manière à ce que la clef secrète utilisée par C_2 soit celle qui est sortie par C_1. On obtient ainsi un circuit C_A qui prend en entrée de l'aléa (celui utilisé par C_1 et éventuellement par C_2') et qui donne en sortie la clef publique générée avec cet aléa par C_1, et le déchiffré du message d'Alice avec la clef secrète correspondante. On peut transformer ce circuit en une formule SAT exprimant qu'il existe une assignation de l'entrée du circuit (l'aléa de génération de clef) dont la sortie est la clef publique pk, et le message clair exprimant un vote pour le candidat untel. L'autorité exécute un protocole *zero-knowledge* (à divulgation nulle de connaissance) pour cette formule SAT avec tout parti qui souhaite vérifier le vote. Les garanties de *soundness* impliquent que si les vérifieurs acceptent, la formule est vraie (avec forte probabilité), ce qui signifie par construction que le déchiffré du message d'Alice est bien ce que prétend l'autorité. La propriété de *zero-knowledge* implique que les vérifieurs n'apprennent rien de plus que cette information, par exemple ils n'apprennent rien de plus sur la clef secrète que ce qu'ils auraient pu trouver par eux-mêmes (puisqu'il peuvent simuler tous seuls l'interaction qu'ils ont avec l'autorité).

   Il ne reste plus qu'à créer un circuit C_B correspondant à Bob de la même manière, et le combiner avec C_A de manière à ce que la sortie des deux circuits puissent être échangées (donc on ne sait pas qui a voté pour qui). Il suffit de voir que cela peut être exprimé par une formule SAT. Pour cela, il suffit par exemple de combiner les circuits en ajoutant un circuit qui prend en entrée un bit b et échange les sorties de C_A et C_B ssi $b = 1$; la formule SAT qu'on en déduit finalement exprime qu'il existe un bit b (et un choix des autres aléas) tel que la sortie globale est ce que prétend l'autorité.

   D'une manière générale, l'idée de cet exercice est que savoir faire des preuves à divulgation nulle de connaissance pour tous les problèmes dans NP est quelque chose de très puissant ! Notamment on sait toujours prouver des affirmations du type : il existe une assignation des entrées d'un certain circuit telle que la sortie est ceci—pour n'importe quel circuit (puisque c'est équivalent à SAT). On sait en particulier toujours prouver qu'une certaine quantité a été calculée d'une manière spécifique, sans avoir à rien révéler sur les valeurs secrètes apparaissant dans ce calcul.

   Mieux, en réalité on sait réaliser ce type de preuve sans interaction, c'est-à-dire qu'il suffit à l'autorité de publier une unique preuve statique, sans avoir à interagir avec chaque personne qui souhaite vérifier.
3. Il n'est pas difficile de voir qu'on peut réaliser la même fonctionnalité pour n votants. Au lieu de tirer un bit b, on tire une permutation aléatoire sur les déchiffrements des n votes. On peut se demander comment la taille de la preuve croît avec n, mais c'est une autre question.

Exercice 7 (Sécurité du protocole de signature de Groth)**.**

1. On demande les chiffrées des messages $m = 0$ et $m = 1$. On obtient r_1, r_2, r_1', r_2' tels que :

$$y_1^{r_1} y_2^{r_2} = y_3 \qquad g y_1^{r_1'} y_2^{r_2'} = y_3$$

On peut alors forger une signature pour un message quelconque m, en effet on a :

$$y_1^{(m-1)r_1} y_2^{(m-1)r_2} = y_3^{m-1}$$
$$g^m y_1^{mr'_1} y_2^{mr'_2} = y_3^m$$

donc en combinant :

$$g^m y_1^{mr'_1-(m-1)r_1} y_2^{mr'_2-(m-1)r_2} = y_3.$$

La signature $(mr'_1 - (m-1)r_1, mr'_2 - (m-1)r_2)$ est donc valide pour le message m.

2. L'équation (1) de l'énoncé implique qu'une signature (r_1, r_2) ne peut être valide que pour un unique message m (le log discret de $y_3 y_1^{-r_1} y_2^{-r_2}$ en base g).

3. (a) Les variables y_1, y_2, y_3 sont indépendantes (chacune est uniformément aléatoire), seule y_3 dépend de c_s donc nous pouvons limiter notre attention à y_3. Il suffit de remarquer que la distribution de y_3 conditionnée à $c_s = c$, pour tout c fixé, reste uniformément aléatoire à cause du choix uniforme de b_s, en particulier elle ne dépend pas de c.

   (b) On vérifie que la signature $(b_s - m - a_s, c_s)$ est valide. D'autre part on a vu que c_s est uniformément aléatoire même conditionné à la clef publique, et $r_1 = b_s - m - a_s$ est l'unique exposant donnant une signature correcte pour $r_2 = c_s$. C'est donc la même distribution qu'une signature véritable : en effet le même relation existe entre r_1 et r_2 dans une signature générée en suivant le protocole (le protocole tel qu'il est écrit tire r_1 uniformément et déduit r_2, mais on voit que les rôles de r_1 et r_2 sont complètement symétriques et qu'on peut faire l'inverse).

   (c) Supposons que $\mathcal{A}$ produit une contrefaçon sur un message $m^* \neq m$. Il produit donc r_1^*, r_2^* tels que :

$$g^{m^*} y_1^{r_1^*} y_2^{r_2^*} = y_3$$
$$g^{m^*} g^{a_s r_1^*} h^{r_2^*} = g^{b_s} h^{c_s}$$
$$g^{(m^* + a_s r_1^* - b_s)(c_s - r_2^*)^{-1}} = h.$$

   L'inversion $(c_s - r_2^*)^{-1}$ est valide parce qu'on a supposé $r_2^* \neq r_2 = c_s$.

4. Comme déjà remarqué plus haut, les rôles de r_1 et r_2 sont complètement symétriques dans ce protocole, donc on peut réécrire le même raisonnement en échangeant les rôles de y_1 (et ses exposants) et y_2 (et ses exposants).

5. Soit $\mathcal{B}$ l'algorithme qui tire $b \leftarrow \{0, 1\}$ uniformément aléatoirement et qui exécute $\mathcal{B}_b$ (qui fait lui-même appel à $\mathcal{A}$). On a vu que la distribution de la clef publique, et de la signature du message demandé par $\mathcal{A}$ sont identiques à celle des clefs publiques et signatures légitimes (en particulier elles sont identiques pour les deux choix de b). L'algorithme $\mathcal{A}$ produit donc une signature forgée (r_1^*, r_2^*) pour un message m^* avec probabilité ϵ, après avoir éventuellement demandé la signature (r_1, r_2) d'un message choisi m. Nous avons vu que $(r_1^*, r_2^*) \neq (r_1, r_2)$, donc $r_1 \neq r_1^*$ ou $r_2 \neq r_2^*$, donc avec probabilité au moins 50% l'hypothèse de l'algorithme $\mathcal{B}_b$ est satisfaite. Avec probabilité au moins $\epsilon/2$ on retrouve donc le logarithme discret de h en base g.