TP3 : accréditation anonyme et coloriage de graphe

Durée du TP : 4h.
Date limite de remise du TP : vendredi 12 février 2015 à 23h59.
Le tp se fait par groupe de 2 étudiants (au maximum) et les fichiers à remettre doivent être envoyées au plus tard vendredi 12 février à minuit par courriel à l'adresse électronique "sgambs@irisa.fr".
Rappel : tout plagiat est formellement interdit et bien qu'il soit naturel que vous puissiez parfois discuter avec vos camarades oralement sur comment attaquer ou résoudre un problème, il est formellement interdit d'échanger des fichiers de code.

Description :
Le but de ce troisième TP est de vous faire implémenter une accréditation anonyme par une preuve de connaissance à divulgation nulle de type 3-coloriage de graphe. Plus précisément, on est dans la situation où un utilisateur souhaite convaincre un vérificateur qu'il connaît une manière de colorier entièrement un graphe avec 3 couleurs de telle façon qu'aucun nœud du graphe n'ait la même couleur que l'un des nœuds voisins mais sans révéler aucune autre information à propos de ce coloriage. Voir l'entrée correspondante sous wikipedia pour plus de détails à propos de ce problème: http://fr.wikipedia.org/wiki/Coloration_de_graphe Le TP lui-même se décompose en quatre parties.

Matériel à rendre :

Il vous est aussi demandé d'écrire un court programme qui sert à tester les différentes fonctions que vous aurez écrit durant ce TP. Vous devez remettre ce programme ainsi que la/les classe(s) qui contiendront les fonctions que vous avez réalisées (avec un minimum de commentaire dans le code) ainsi qu'un court rapport qui répond aux questions posées dans la partie 4.

Partie 1 : génération du graphe et du coloriage

On suppose que le graphe qu'on souhaite colorier se compose de 20 noeuds et on souhaite le représenter sous forme de matrice d'adjacence:

```
  1 2 3 4 5
1   x x   x
2 x   x x
3 x x   x x
4   x x
5 x   x
```

où un "x" représente une arête entre les noeuds de ligne et la colonne correspondante.
Pour cette première partie, vous devrez générer un graphe aléatoire qui est 3-coloriable. Pour cela vous utiliserez l'algorithme suivant :
1. Vous commencez par tirer une couleur au hasard pour chaque noeud entre "rouge", "vert" et "bleu" (chaque couleur doit être équiprobable). Libre à vous de décider dans quelle type de structure vous souhaitez ensuite stocker les couleurs, cela pourrait être par exemple un tableau où chaque case servirait à stocker sous forme de String ou d'entier la couleur du noeud se trouvant sous l'index

correspondant.
2. Vous devez ensuite générer la matrice d'adjacence d'un graphe qui contient seulement des arêtes reliant des noeuds comportant des couleurs différentes. Pour cela, vous *pouvez* pour chaque arête possible ET qui ne relierait pas deux noeuds ayant la même couleur, choisir aléatoirement de relier ou non ces deux noeuds dans la matrice d'adjacence (par exemple avec probabilité 1/2). Pour éviter de faire le travail en double, n'oubliez pas que la matrice que vous allez obtenir est diagonale symétrique.

Votre fonction s'intitulant `genererGraphe3Coloriable` doit produire en sortie un objet de la classe Graphe (que vous aurez écrite vous même) qui contient la matrice d'adjacence du graphe ainsi que son coloriage sous forme de tableau.

Partie 2 : mise en gage des couleurs

On suppose que la structure du graphe est connue et publique mais que le coloriage constitue le secret. Lorsqu'un utilisateur souhaite prouver à un vérificateur distant qu'il connaît ce secret, c'est à dire une manière de 3-colorer le graphe, il va mettre en gage (*commitment* en anglais) les différentes couleurs des noeuds puis les envoyer au vérificateur. Comme étape initiale avant la mise en gage, l'utilisateur commence par faire une permutation aléatoire sur les couleurs. Par exemple en transformant "rouge" en "vert", "bleu" en "rouge" et "vert" en "bleu".

Pour mettre en gage une couleur l'utilisateur tire une valeur aléatoire de 128 bits qu'il concatène à la couleur avant de passer le tout à travers une fonction de hachage. Soit c_i la couleur du ième noeud et r_i le nombre aléatoire de 128 bits choisit aléatoirement pour cette couleur et $y\_i = h(r\_i \| c\_i)$ la valeur mise en gage pour h une fonction de hachage (vous utiliserez par exemple la fonction de hachage SHA-1). Pour la mise en gage, l'utilisateur envoie les différents y_i (pour i allant de 1 à 20) dans un tableau au vérificateur.

Pour cela, il vous est demandé de réaliser une fonction qui s'appelle `miseEnGageColoriage` et qui prend deux arguments en entrée 1) un tableau contenant les couleurs de taille 20 et 2) un tableau contenant des valeurs aléatoires de 128 bits. Cette fonction doit produire en sortie un tableau de 20 cases contenant les valeurs mises en gage.

Partie 3 : preuve de connaissance à divulgation nulle d'un 3-coloriage

Soit le protocole suivant de preuve de connaissance à divulgation nulle pour convaincre un vérificateur de la connaissance d'un 3-coloriage d'un graphe particulier:
1. L'utilisateur envoie une mise en gage d'un coloriage à un vérificateur en n'oubliant pas au préalable de faire une permutation aléatoire sur les couleurs des noeuds (comme précisé dans la partie précédente).
2. Le vérificateur demande à voir les couleurs de deux noeuds i et j qui sont reliés par une arête (telle que défini par la matrice d'adjacence).
3. L'utilisateur envoie les (r_i,c_i) et (r_j,c_j) qui correspondent à ces deux noeuds.
4. Le vérificateur calcule ensuite si $h(r\_i \| c\_i) = y\_i$ et $h(r\_j \| c\_j) = y\_j$ et vérifie aussi que les deux couleurs soient différentes (soit $c\_i \neq c\_j$). Si ces trois conditions

sont réunies le vérificateur accepte et sinon il refuse.
Ecrivez une fonction `preuveColoriage` qui prend en entrée la matrice d'adjacence d'un graphe, une mise en gage d'un coloriage et l'information nécessaire pour ouvrir les mises en gages de noeuds *i* et *j* et retourne vraie si l'étape 4 dans le protocole ci-dessus est réussie et faux sinon.
Cette procédure est ensuite répétée 400 fois et l'utilisateur est considéré comme authentifié seulement s'il passe tous les tests.

Il vous est aussi demandé d'écrire un court programme qui implémente un scénario servant à tester les différentes fonctions que vous aurez écrit durant ce tp.

Partie 4 : analyse du protocole

En plus des différentes classes, remettez un rapport qui répond aux questions suivantes montrant votre compréhension du protocole :
1. Pourquoi est ce que si l'utilisateur et le vérificateur sont honnêtes et suivent les directives du protocole, un utilisateur possédant la preuve d'un 3-coloriage pourra toujours convaincre le vérificateur (propriété de *completeness*)?
2. Pourquoi est ce qu'un utilisateur qui ne possède pas de preuve d'un 3-coloriage ne pourra pas réussir à convaincre un vérificateur, sauf avec une probabilité négligeable (propriété de *soundness*)?
3. Expliquer en quoi ce protocole est à divulgation nulle (*zero-knowledge* en anglais), c'est à dire qu'il n'apporte aucune autre information au vérificateur que la véracité de l'énoncé.