TP4 : canal de diffusion anonyme et génération de secret

Durée du TP : 4h.
Date limite de remise du TP : vendredi 26 février 2016 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 26 février à 23h59 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 quatrième TP est de vous faire implémenter un mécanisme de génération de secret via l'implémentation d'un canal de diffusion anonyme. Un canal de diffusion anonyme permet d'envoyer un message à tous les participants d'un réseau mais sans que ceux-ci ne puissent connaître l'identité de l'envoyeur du message (à moins bien sûr que celui-ci ne fasse figurer en clair son identité dans le message envoyé). Un forum sur Internet qui permet de poster des messages sans avoir à s'identifier est un bon exemple d'implémentation pratique de canal de diffusion anonyme. À noter que dans ce cas, le canal possède la propriété supplémentaire d'être asynchrone, c'est à dire qu'il ne requiert pas que les utilisateurs soient en ligne au moment où le message est envoyé tout en permettant à n'importe qui de pouvoir consulter les messages plus tard.

Pour ce TP, on se place dans un modèle où l'adversaire est passif, c'est à dire qu'il se contente d'écouter les communications comme le canal de diffusion anonyme mais ne cherche pas à tricher activement en postant de faux messages par exemple. Le TP lui-même se décompose en trois 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 ainsi qu'un rapport qui répond aux questions posées dans la partie 3.

Partie 1 : canal de diffusion anonyme

Pour cette première partie, vous devez écrire une classe qui simule un canal de diffusion anonyme (de type forum) et permet de réaliser les opérations suivantes:
1. Poster un message de manière anonyme. Votre fonction s'intitulant `posterMessageAnonyme` doit prendre en entrée une chaîne de caractères correspondant au message et le poster dans la structure qui servira le canal de diffusion anonyme. Libre à vous de choisir la structure de données qui vous semble la plus appropriée pour implémenter le canal de diffusion anonyme. Le message posté doit aussi comporter une étiquette temporelle indiquant à quel moment le message a été diffusé (par exemple *16h39m25s*). La structure de données que vous utilisez doit ordonner les messages chronologiquement en se basant sur cette étiquette temporelle.

2. Lire une série de messages postées de manière anonyme. Votre fonction s'intitulant `recupererMessagesAnonymes` prendra comme arguments d'entrée deux étiquettes temporelles (une de début et une de fin de période) et retournera la liste de tous les messages anonymes postés durant cette période.

Partie 2 : génération de secret via canal de diffusion anonyme

À partir simplement de la primitive du canal de diffusion anonyme, il est possible de réaliser une génération de secret (à partir de rien) entre deux entités tel que même si un adversaire passif prend lui aussi connaissance de tous les messages il n'apprend aucune information sur le secret.

Pour fins d'illustration, on va supposer que les deux entités se nomment Alice et Bob. Ils vont réaliser le protocole de génération de secret suivant :
1. Alice tire aléatoirement à pile ou face un bit *b*.
2. Si le bit *b* vaut 0, elle construit un message comprenant le nom "Alice" et sinon pour *b=1* elle construit un message comprenant le nom "Bob".
3. Alice poste ce message de manière anonyme après un temps aléatoire pris au hasard dans une certaine fourchette (par exemple entre 1 et 10ms).
4. Alice recommence à partir de l'étape 1.
Du côté de Bob il réalise en parallèle le même protocole à la différence près que ses réactions sont inversées, si *b=0* il construit un message comprenant le nom "Bob" et sinon si *b=1* son message sera "Alice".
On suppose qu'Alice et Bob gardent chacun une liste de messages qu'ils ont envoyées (avec les étiquettes temporelles correspondantes). Le protocole se déroule pendant une certaine période (par exemple 1 minute) dont le moment de début et la durée ont pu être négociées par Alice et Bob au préalable par exemple en utilisant le canal de diffusion anonyme.

Après que cette période se soit écoulée, Alice et Bob extraient le secret (qui correspondra à une chaîne de bits) de la manière suivante. Ils regardent les messages dans l'ordre chronologique (on suppose par exemple que le nombre de messages est *n*). Pour chaque message d'index *i* si le message contient le texte "Alice" et qu'il a vraiment été envoyé par Alice alors *s_i* le bit secret généré vaudra 0 (même chose si le message contient le texte "Bob" et qu'il a vraiment été envoyé par Bob). À l'opposé si le message contient le texte "Alice" mais qu'il a été envoyé par Bob alors *s_i* vaudra 1 (même chose si chose si le message contient le texte "Bob" mais qu'il a été envoyé par Alice). Une fois être passé à travers les *n* messages Alice et Bob se retrouvent en possession d'un secret *s* sous la forme d'une chaîne aléatoire de *n* bits.

Pour implémenter cette deuxième partie, vous devez écrire deux fonctions qui réalisent les opérations suivantes:
1. Génération de secret en utilisant le canal de diffusion anonyme. Votre fonction s'intitulant `genererSecret` doit simuler le protocole de génération de secret tel que décrit plus haut en utilisant le canal de diffusion anonyme que vous avez implémenté dans la partie 1. Cette fonction prendre en entrée au moins trois paramètres, deux chaînes de caractère représentant le nom des interlocuteurs (par exemple "Alice" et "Bob" dans l'exemple ci-dessus) ainsi que la durée pendant laquelle ils doivent réaliser ce protocole.
2. Extraction de secret. Votre fonction s'intitulant `extraireSecret` doit permettre à

partir d'un transcript de messages obtenu suite à l'application de `genererSecret` combiné avec la vue partielle des deux entités d'extraire un secret commun. Le secret sera de taille *n*, le nombre de messages du transcript.

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 3 : analyse du protocole

En plus des classes réalisées pour les parties 1 et 2, remettez un rapport qui répond aux questions suivantes :
1. Un canal de diffusion anonyme est une primitive qui peut être utilisée comme brique de base pour réaliser de nombreuses tâches. Imaginer deux autres applications possibles du canal de diffusion anonyme et décrivez les chacune en quelques phrases.
2. Argumenter sur le fait que suite au protocole décrit dans la deuxième partie, Alice et Bob se retrouvent en possession d'un secret sur lequel l'adversaire n'a aucune information. Pourquoi est ce le cas?
3. Expliquer en quelques mots pourquoi il est important d'avoir des primitives permettant à deux entités de générer un secret commun à travers un canal de communication qui est potentiellement surveillé par un espion.