TP5 - Communications Sécurisées

01 Octobre 2021

L'objectif de ce TP est d'imbriquer les différents éléments cryptographiques que vous avez vu/manipulé afin de construire une petite application de communication sécurisée.
Un rendu est attendu : une archive contenant tout votre code ainsi qu'un rapport. N'oubliez pas de commenter votre code et de détailler votre rapport autant que possible.

Le rendu est à envoyer par mail pour le **10 Décembre, 23h59**, en incluant "[BCS]" dans l'objet, à l'adresse daniel.de-almeida-braga@irisa.fr.

Ce TP est un projet à part entière qui va vous demander un peu de travail pour avoir un rendu propre. N'attendez pas la dernière minute pour le faire. Je reste ouvert aux questions en dehors de la séance de TP, mais je ne répondrai pas à 20 messages la veille du rendu.

> Pour ce TP, vous pourrez vous mettre en binôme, et choisir le langage de programmation de votre choix.

Vous êtes relativement libre dans le design de votre application, puisque l'exercice principale est de choisir les bonnes primitives cryptographiques pour répondre à vos besoins. Voici les contraintes :

- Vous devez implémenter un client et un serveur, qui peuvent tourner en local (similaire au matériel mis à disposition pour le TP3).
- Le client s'authentifie par l'intermédiaire d'un mot de passe
  - On peut supposer que le mot de passe (et uniquement le mot de passe !) a été transmis par un canal sécurisé tiers (pour vous épargner la mise en place d'une PKI et la lourdeur de la cryptographie asymétrique).
  - Vous pouvez choisir d'entrer le mot de passe sur le serveur à chaque fois, ou de le stocker (attention au stockage, je ne veux pas de mot de passe en clair !)
  - Vous pouvez avoir recours à de la crypto asymétrique si vous le souhaitez, mais le fait qu'un mot de passe soit déjà partagé devrait vous épargner cette partie.
  - **Ne transmettez pas la clé (ou autre secret) en clair !**
- Le mot de passe est dérivé en clé de chiffrement pour protéger la suite des échanges (pour les fonctions de dérivation, regardez du côté des fonctions *memory hard* telles que scrypt ou argon2i).
  - Attention, même si le mot de passe est le même la clé de session devrait changer à chaque nouvelle session (pensez à utiliser des paramètres éphémères).
- L'application doit garantir un niveau de sécurité de 128 bits (ou plus).
- Le client et le serveur peuvent, à la fin, échanger des messages de manière sûre (la confidentialité, l'intégrité et l'authenticité des messages est garantie). Pensez à choisir un mode chiffrement adéquat.
- Libre à vous de ne supporter qu'une suite de chiffrement ou plusieurs.

N'oubliez pas que le client et le serveur doivent supporter les mêmes mécanismes pour pouvoir établir un canal de communication. L'objectif ici n'est pas d'implémenter chacune des primitives cryptographiques à utiliser. Vous pouvez vous baser sur des bibliothèques/packages existant (comme PyCryptodome) qui devrait vous fournir ce dont vous avez besoin.

Votre rapport devra contenir :

1. une description de votre protocole, étape par étape, avec un schéma l'illustrant,
2. une justification de vos choix de mécanismes.
3. les faiblesses éventuelles de votre application si vous en avez identifié (je n'attend pas de vous que l'application soit parfaitement sécurisée, mais vous devez être conscient de ses faiblesses).

Seuls l'aspect cryptographique du protocole (justification dans le rapport) et la lisibilité/fonctionnalité du code seront évalués.

Vous pouvez trouvez des conseils sur le mécanismes à utiliser dans des documents tels que le RGS de l'ANSSI ou le SOGIS (plus récent).