Advance Communication Architecture

Labo 2

Luca Srdjenovic & Gaetan Kolly

15 mars 2022

Implémentation du procédé

Algorithmes

Les algorithmes ont été implémenté dans le fichier AlgoGSM.py

Selon le protocol GSM, les variables doivent être de la taille suivante:
SRAND= 128bits
SRES= 32bits
Ki = 128bits
Kc = 64bits

A3:
Il a été décidé d'utiliser comme algorithme Blowfish. Blowfish accepte une clé de chiffrement de 32 à 448bits et une taille de bloc de 64bits.
Vu que notre taille de bloc (SRAND) est de 128bits, dans notre implémentation le bloc est tout d 'abord concaténé en faisant un xor des 64 premiers bits avec les 64 derniers.
Ensuite ces 64bits résultants sont chiffrés avec la clé Ki de 128bits.
Finalement, le texte chiffré (64bits) est à nouveau concaténé avec le même procédé, soit un xor des 32 premiers bit avec les 32 derniers bits ce qui donne le SRES de 32 bits voulu.

A8:
Pour la dérivation des clés, il a été décidé d'utiliser AES. Cet algorithme chiffre des blocs de 128bits avec une clé de 128bits. Notre bloc de texte clair correspond à RAND (128bits) et la clé de chiffrement est le secret partagé Ki (128bits). Le texte chiffré sortie est concaténé en faisant un xor des 64 premiers bits avec les 64 derniers. Le résultat donne la clé Kc (64bits).

A5:
Pour chiffrer la communication, il a été d'utiliser Blowfish avec un mode opératoire CTR afin de réaliser un algorithme de chiffrement par flux. Le mode opératoire CTR utilise un nonce qui doit impérativement être utilisé qu'une seule fois. Ce nonce est alors généré aléatoirement puis est envoyé au récepteur des textes. Ensuite, le texte entier est chiffré à l'aide de la clé Kc puis est envoyé.

Code

Les fichiers ci-dessous ont été implémenté pour ce travail:

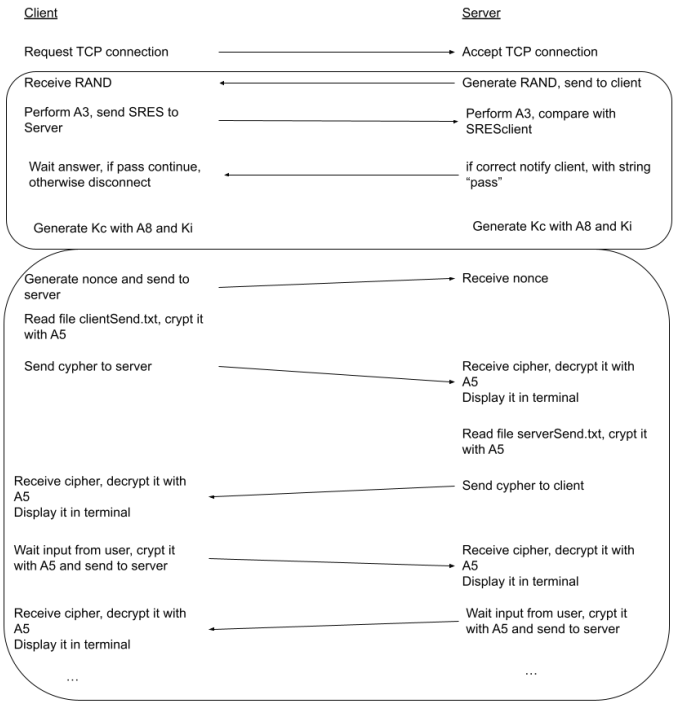
- Server.py: lance le programme server
- Client.py: lance le programme client
- serverModule.py:
    - fonction startServer: manage communication
    - class: ServerGSM: contient les fonctions d'authentification et d'envoi/réception des donnés
- clientModules.py:

- fonction startClient: manage communication
- class: ClientGSM: contient les fonctions d'authentification et d'envoi/réception des donnés

- AlgoGSM.py: contient les fonctions de chifrement et de déchiffrement

Séquence

Notre protocol a été implémenté de la manière suivante:



De plus, toutes les 20 minutes un message est envoyé depuis le server au client afin de réaliser à nouveau l'authentification dans le but de régénérer une clé Kc.

Test de l'application

Afin de tester l'application, il suffit de lancer le programme python server.py dans un invité de commande puis le programme client.py dans un autre invité de commande.

Les paquets reçus ainsi que le cypher sont affichés à l'utilisateur. Tout d'abord les fichiers clientSend.txt et serverSend.txt sont envoyés puis les programmes demandent des entrées utilisateurs en mode chat qui seront envoyé avec chiffrement puis affiché de l'autre côté.

Après 20 minute l'authentification est refaite.

Comment peut-on faire une authentification mutuelle ?

Il est possible de faire une authentification mutuelle en ajoutant une étape de challenge du côté client.