Projet : chiffrement des messages échangés

Cryptage symétrique et asymétrique

Il existe deux sortes de cryptages, symétrique et asymétrique.

- Dans le **cryptage symétrique**, une même clé est utilisée pour encrypter et décrypter l'information.
    - Dans le projet, cela peut être utilisé par exemple par le serveur de ballons qui transmets un ballon à un robot, le robot remettra plus tard ce « ballon » au validateur de but, qui demandera au générateur de ballons de confirmer que c'est bien un bon ballon en utilisant la clef pour décrypter le ballon (ou bien le générateur de ballons partage sa clef avec le validateur de but qui peut lui même vérifier qu'un ballon est ok).
    - Note implémentation : *OpenSSL* est un outil gratuit qui contient une bibliothèque nommée *libcrypto* qui permet de coder et décoder un message suivant une clef.
- Dans le **cryptage asymétrique**, deux clés (publique et privée) sont utilisées pour crypter et décrypter les messages. L'utilisateur distribue sa clé publique et tient secrète sa clé privée avec laquelle il décode les messages codés qui lui sont adressés (cryptés par ceux qui lui envoient des messages avec sa clé publique). Ici, tout le monde peut utiliser une clé publique pour écrire au destinataire, mais seul le destinataire (détenteur de la clé privée correspondante) peut décrypter les messages qui lui sont adressés. Dans ce cas, le cryptage assure la confidentialité de données portées à la seule connaissance des personnes autorisées.
    - Dans le projet, cela peut être utilisé dans les situations où l'on veut que le contenu d'un message envoyé par A à B sur un réseau commun à C reste confidentiel. Même si C est à l'écoute, tout ce qu'il verra passer est un message encodé avec la clef publique de B, clef qu'il ne connait pas…donc C ne peut connaître le contenu du message échangé.
    - Si les messages échangés par A et B doivent être confidentiels dans les deux sens (c'est-à-dire quand A parle à B, et quand B parle à A) on agit ainsi : chaque interlocuteur chiffre alors les messages qu'il envoie avec la clé publique de l'autre et déchiffre les messages qu'il reçoit avec sa propre clé secrète.
    - Note implémentation : *OpenSSL* offre des bibliothèques qui permettent de communiquer par sockets sur la base d'un cryptage asymétrique (voir post sur forum Moodle pour un exemple). Dans ce cas, il s'occupe de façon transparente pour le programmeur de générer / communiquer / utiliser les clefs publiques et privées des deux interlocuteurs :)

Note sur l'encodage : pour transmettre *manuellement* un message crypté (sur une clef USB, sur une socket non sécurisée), on utilisera avantageusement le codage base64 qui n'utilise que des caractères basiques et donc qui ont plus de chances d'être compris par les deux interlocuteurs, quel que soit leur système d'exploitation et langage de programmation respectifs.