1 Problématique

Les paquets IP transitent sur le réseau internet en circulant de routeurs en routeurs. En théorie, rien n'interdit à un routeur d'inspecter un paquet et donc d'en connaître son contenu. Cette situation est problématique dans de nombreuses situations (transactions bancaires, documents personnels…).

Comment chiffrer le contenu des communications ?

2 Chiffrement symétrique

2.1 Principe : le code de César

La sécurisation d'une communication se déroule en deux étapes :

— La source utilise une *fonction de chiffrement* pour coder un message m avec une clé de chiffrement k. La fonction produit en sortie un message chiffré s.

`chiffrement(m, k)` $\rightarrow$ `s`

— Le destinataire utilise une *fonction de déchiffrement* pour décoder le message s avec la clé de chiffrement k. La fonction produit en sortie le message clair m.

`déchiffrement(s, k)` $\rightarrow$ `m`

À retenir

Dans un chiffrement symétrique on utilise la même clé pour chiffrer et déchiffrer le message.

Activité 1 : Le chiffrement de César utilise un décalage alphabétique comme clé de chiffrement.

1. Écrire la fonction `chiffrement(message: str, cle: int)` $\rightarrow$ `str` qui code le *message*. On n'utilisera que des caractères majuscules ASCII dans le message et on supprimera les espaces.
2. Écrire la fonction `dechiffrement(message: str, cle: int)` $\rightarrow$ `str` qui déchiffre le *message*.
3. Tester la méthode avec une clé $k = +3$ sur le message : *LANSIESTFANTASTIQUE*
4. Quelles sont les faiblesses de cette méthode ?

2.2 Chiffrement polyalphabétique

2.2.1 Principe

Plutôt que d'opérer un simple décalage, on recopie la clé de chiffrement de façon à obtenir une chaîne de la longueur du message. Utilisons la clé *NSI*.

B	R	A	V	O
N	S	I	N	S

Une même lettre ne sera plus forcément codée par le même symbole.

Activité 2 :

Christophe Viroulaud

1. Remplacer chaque lettre en son équivalent ASCII.
2. Écrire la fonction `int_en_bin(nb: int)` → `str` qui renvoie la représentation binaire de l'entier *nb*.
3. Convertir chaque entier en binaire.

2.2.2 Chiffrement par *ou exclusif*

On applique la porte logique *xor* entre chaque bit du message et de la clé. Une propriété intéressante de cette porte est qu'elle est réversible :

$$\text{Si } A \oplus B = C \text{ alors } A \oplus C = B \text{ et } B \oplus C = A$$

Activité 3 :

1. Appliquer le ou exclusif pour chaque bit du message.
2. Écrire la fonction `bin_en_int(paquet: str)` → `int` qui renvoie l'entier correspondant au paquet de bits.
3. Utiliser la fonction pour trouver l'entier correspondant à chaque paquet de sept bits.
4. Donner alors le message chiffré.

3 Avantages du chiffrement symétrique

Les algorithmes de chiffrement symétriques utilisés dans les communications appliquent des principes similaires au chiffrement par *ou exclusif*. Ils sont très sûrs : la taille minimale des clés est actuellement de 80 bits soit 2^{80} possibilités. Ils sont également très rapides et permettent ainsi de chiffrer, en temps réel, les données stockées sur un disque dur.

On peut citer :

— *AES pour Advanced Encryption Standard :* choisi par l'institut de standardisation américain NIST (National Institute of Standards and Technology) en décembre 2001.

— *Chacha20 :* date de 2008 et améliore les performances d'un autre algorithme (Salsa20)