Cryptage RC4

1. Présentation

L'algorithme RC4 a été pensé par Ron Rivest en 1987 et développé pour la RSA Security. Il est basé sur Les permutations aléatoires, avec des opérations sur des octets.

L'algorithme a une longueur de clé variable (de 1 à 256 octets). Cependant à cause des lois d'exportation, la clé a souvent une longueur de 40 bits. La clé est utilisée pour initialiser une "table d'états" de 256 octets. La table d'état est employée pour la génération d'octets pseudo-aléatoires et ensuite pour produire le flux pseudo-aléatoire avec lequel le texte clair sera transformé avec l'opération du OU-Exclusif.

Le RC4 est employé dans plusieurs applications commerciales, par exemple dans le protocole SSL et dans Oracle Secure SQL. Il s'exécute très rapidement dans les logiciels.

Le RC4 au format allégé est appelé ARCFOUR (ARC4).

2. Algorithme

Deux étapes sont nécessaires pour le chiffrement :

- Mélange de la "table d'états" à l'aide de la clé (réalisé une seule fois avant le début du cryptage),
- Chiffrement du texte clair.

Remarque :

Les différentes additions qui seront réalisées à partie du tableau d'état seront effectuées modulo 256 (correspondant à la longueur du tableau). Cela évite un débordement dans le parcours du tableau (après l'indice 255 on repasse à l'indice 0).

2.1. Mélange de la "table d'états"

Les différentes étapes :

- Initialisation du "tableau d'état" de 256 octets. Le premier octet est à 0, le deuxième octet à 1,…le 256ème octet à 255.
- Mélange du "tableau d'état", en effectuant 256 permutations à partir de la clé.

i,j et indiceCle étant 3 indices (i,j relatifs au "tableau d'état" et indiceCle au tableau clé) initialisés à 0.

. Ajouter à la variable j l'élément d'indice i du "tableau d'état" et l'élément d'indice indiceCle du tableau cle (indiceCle étant égal à i modulo la longueur de la clé),

. Permuter l'élément d'indice i du "tableau d'état" avec l'élément d'indice j du même tableau.

Après le mélange les variables i et j sont remises à 0.

2.2. Chiffrage

En plus des variables i et j (déjà définies), nous utiliserons une variable n permettant de définir l'indice d'élément du "tableau d'état" qui servira à crypter l'octet de donnée à l'aide d'un ou exclusif .

Les différentes étapes :

- Incrémenter la variable i de 1.
- Ajouter à la variable j l'élément d'indice i du "tableau d'état".
- Permuter l'élément d'indice i du "tableau d'état" avec l'élément d'indice j du même tableau.
- Déterminer n en additionnant l'élément d'indice i du "tableau d'état" avec l'élément d'indice j du même tableau.
- On réalise un ou exclusif entre l'octet de la donnée à crypter et l'élément d'indice n du "tableau d'état".

Remarque :

Le cryptage étant réalisé à l'aide d'un ou exclusif, le décryptage sera réalisé en utilisant la même technique.

Etape 1 :

On suppose que la clé est "ABCD" et est dans un tableau appelé "maCle" (de taille 4).

Constantes

```
TAILLE_TABLEAU_ETAT c'est entier 256    // taille du tableau d'état
TAILLE_CLE c'est entier 4               // taille de la clé de cryptage
```

Variables

Nom	**Type**	**Initialisation**	**Commentaire**
tableauEtat	tableau entier [TAILLE_TABLEAU_ETAT]	ind	tableau d'état
i	entier	ind	indice du tableau d'état
j	entier	ind	indice du tableau d'état
maCle	tableau entier [TAILLE_CLE]	ind	contient la clé de cryptage
indiceCle	entier	ind	indice du tableau clé
n	entier	ind	donnée de cryptage
octet	entier	ind	donnée à cripter

```
// Initialisation du tableau d'état
Pour i VariantDe 0 à TAILLE_TABLEAU_ETAT - 1
        tableauEtat[i] ← i
FinPour

// Mélange du "tableau d'état", en effectuant 256 permutations à partir de la clé
j ← 0
Pour i VariantDe 0 à TAILLE_TABLEAU_ETAT - 1
        indiceCle ← i  MOD TAILLE_CLE
        j ←( j + tableauEtat[i] + maCle[indiceCle] ) MOD TAILLE_TABLEAU_ETAT
        swap ( tableauEtat[i], tableauEtat[j] )
FinPour
i ← 0
j ← 0
```

Action swap

Fonctionnalité

Inversion de deux entiers

Paramètres

Nom	**Type**	**Entrée/Sortie**	**Commentaire**
val1	entier	E/S	
val2	entier	E/S	

Variables

Nom	**Type**	**Initialisation**	**Commentaire**
inter	entier	ind	

Début
inter ← val1
val1 ← val2
val2 ← inter
Fin

Etape 2 :

// Chiffrage

i ← (i + 1) MOD TAILLE_TABLEAU_ETAT
j ← (j + tableauEtat[i]) MOD TAILLE_TABLEAU_ETAT

swap (tableauEtat[i], tableauEtat[j])
n ← (tableauEtat[i] + tableauEtat[j]) MOD TAILLE_TABLEAU_ETAT

// Cryptage
octet ← octet xor tableauEtat[n]