Rapport Module de Crypto

UVSQ université PARIS-SACLAY
IUT de Vélizy-Rambouillet
CAMPUS DE VÉLIZY-VILLACOUBLAY
CAMPUS DE RAMBOUILLET

Alexis Araujo
Samir Subra
Nino Pires
Antoine Bazire
SAE groupe 7

Introduction :

Le travail que nous devions effectuer est la création d'un module dans nôtre projet permettant de chiffrer et déchiffrer avec l'algorithme RC4. L'utilisateur utilisant ce module pourra donc rentrer un mot (en clair) et une clé servant à chiffrer le mot qu'il a choisi, et cela renverra à l'aide de l'algorithme RC4 le mot chiffrer.

Par la suite, sur un autre page (web), l'utilisateur pourra voir ses mots en clair, clés et mot chiffrés dans un tableau qu'il pourra télécharger s'il le souhaite.

Création du module de chiffrement / déchiffrement

Nous avons choisi d'implémenter l'algorithme RC4 en python sur notre page web car nous sommes très familiarisé avec ce langage.

```
import sys
```

Tout d'abord, nous importons le module sys qui nous permettra d'utiliser des fonctions propres à des systèmes qu'on verra par la suite.

```
def permutation(key):
    """Initialise s pour l'algorithme RC4 en utilisant la clé donnée en ASCII"""
    s = list(range(256))
    j = 0
    for i in range(256):
        j = (j + s[i] + ord(key[i % len(key)])) % 256
        s[i], s[j] = s[j], s[i]
    return s
```

Nous commençons à récupérer l'algorithme 2 du sujet KSA puis retranscrit en langage python. La fonction a pour but d'additionner modulairement et de permuter les éléments de la suite "s" en fonction de la clé choisie par l'utilisateur. Cela renvoie la nouvelle suite d'éléments compris entre 0 et 255 pseudo aléatoire.

A noter : la fonction “ord(c)” permet de d’avoir le code ASCII d’un caractère “c” en paramètre.

```python
def rc4(s, length):
    """Génère une clé de la longueur donnée en utilisant s"""
    i = 0
    j = 0
    keystream = []
    for _ in range(length):
        i = (i + 1) % 256
        j = (j + s[i]) % 256
        s[i], s[j] = s[j], s[i]
        keystream.append(s[(s[i] + s[j]) % 256])
    return keystream
```

Cette fonction est l'algorithme 1 du sujet traduit en python permettant de générer une clé de chiffrement à partir de la nouvelle suite “s” obtenue lors de la permutation de longueur paramètre “length”. Elle renvoie “keystream” la clé de chiffrement suite à la permutation de l’addition modulaire comme vu précédemment.

```python
def cryptage(plaintext, key):
    """Chiffre le texte en clair donné à l'aide de la clé donnée en ASCII et renvoyez le texte chiffré en hexadécimal
    """
    s = permutation(key)
    keystream = rc4(s, len(plaintext))
    ciphertext = [chr(ord(plaintext[i]) ^ keystream[i]) for i in range(len(plaintext))]
    return "".join("{:02x}".format(ord(c)) for c in ciphertext)
```

Voici la fonction “cryptage” permettant de chiffrer le message “plaintext” à l’aide d’une clé qu’il rentre aussi en paramètre.

On peut voir qu’elle appelle les fonctions vu par le passé comme “permutation” qui prend la clé de l’utilisateur pour faire une suite “s” pseudo aléatoire.

Puis à l’aide de la fonction “rc4”, on crée une clé de chiffrement avec cette suite “s” de la même longueur que le message “plaintext”.

Après cela, on met dans la variable “ciphertext”, le résultat de chaque caractère du message en claire “plaintext” traduit en amont en son code ASCII (avec la fonction ord()), puis le “^” indique qu’il fait un “ou exclusif” avec l’élément de la clé de chiffrement “keystream” à la même position que celui-ci. Ensuite on remet en caractère le code ASCII des caractères à l’aide la fonction “chr()”, donnant le message chiffré.

Pour finir, on passe chaque caractères “c” du message chiffré en hexadécimal sans espace entre les caractères comme nous le prouve les guillemets juste après le “return”. Elle renvoie donc le message chiffré en hexadécimal.

```python
texte = str(sys.argv[1])
cle = str(sys.argv[2])
print(cryptage(texte,cle))
```

C’est la partie “main” du code, grâce au module “sys”, lors de l’appelle au script python dans la partie php, on met en paramètre le texte et la clé dans la commande et la fonction “sys.argv” récupère les paramètres qu’on a rentré puis il affiche le résultat.

```python
def decryptage(ciphertext, key):
    """Déchiffre le texte chiffré donné (en hexadécimal) en utilisant la clé donnée dans ASCII et renvoie le texte en clair
    """
    ciphertext = bytes.fromhex(ciphertext)
    state = permutation(key)
    keystream = rc4(state, len(ciphertext))
    plaintext = [chr(ciphertext[i] ^ keystream[i]) for i in range(len(ciphertext))]
    return "".join(plaintext)
```

Pour le déchiffrage, nous créons un objet bytes contenant les valeurs hexadécimal (2 par octet) du texte chiffré “ciphertext”.
A savoir que key est la même clé que l’utilisateur à choisi pour chiffrer son message, elle permet aussi de déchiffrer celui-ci. On fait la permutation de s avec cette même clé.
Puis, on refait la fonction RC4 pour avoir la clé de chiffrement.
Enfin, on fait la même chose comme pour le chiffrement le “ou exclusif” avec la clé de chiffrement sur le texte chiffré, pour nous redonner le message en clair.

```python
chiffrage = str(sys.argv[1])
cle = str(sys.argv[2])
print(decryptage(chiffrage,cle))
```

Voici la partie main du fichier python déchiffrement, c’est la similaire au chiffrement sauf que le message en clair est le message chiffrer.

```python
def enregistrer_fichier(ciphertext, fichier):
    """Enregistre le texte chiffré donné (en hexadécimal) dans le fichier spécifié"""
    with open(fichier, "w") as f:
        f.write(ciphertext)

def lire_fichier(fichier):
    """Lire le texte chiffré (en hexadécimal) à partir du fichier spécifié et le renvoyer"""
    with open(fichier, "r") as f:
        return f.read()
```

Ces 2 fonctions nous permettent d’enregistrer un texte “ciphertext”, dans un fichier “fichier”, et de lire un fichier “fichier”.

Vers la clé WEP

La clé WEP utilise l’algorithme de chiffrement RC4, une clé WEP 64 bits dont les 24 premiers bits sont pour le vecteur d'initialisation, qui n’est pas chiffré et 40 bits pour le chiffrement. Et cette clé 64 bits forme la clé de chiffrement RC4.

Le vecteur d’initialisation est aussi la clé qui était choisie par l’utilisateur, il permet donc d’être au début du message chiffré et de faire la permutation de la suite s.
Il doit donc être communiqué à chaque échange pour le déchiffrement.

Pour le changement du code il suffit que la clé choisie par l'utilisateur soit celle aussi qui sera ajoutée devant la clé de chiffrement lors de sa création, il faut donc réduire la longueur de la clé de chiffrement demandée.
Si on veut une clé de chiffrement de 64 bits, le client peut choisir le vecteur d'initialisation par exemple 20 bits, il faudra demander la clé de chiffrement sur la taille de 64 bits moins la taille du vecteur d'initialisation, lors de l'appel à la fonction rc4.