SAE 3.09 : Cryptographie et sécurité

Evann YANG Thomas RABILLON

Tout d'abord, pour cette SAE, ils nous étaient demandés de déchiffrer 3 messages, chaque message à son type de cryptage prédéfini et chaque message déchiffré fournira un indice nécessaire pour déchiffrer le suivant. Le projet se fait en binôme et il nous est demandé de réaliser un programme python pour le déchiffrement

Premier message

Pour le premier message qui nous a été donné (voir fichier message_crypte1), il a fallu utiliser le chiffrement de cesar pour déchiffrer ce message.

Pour cela nous avons utilisé les fonctions :

```
def dechiffrement_message1(message_code, decalage):
    """
    Fonction qui permet de déchiffrer un message chiffré avec le chiffrement de César

    Args :
        message_code (String) : Le message chiffré
        decalage (int) : Le décalage utilisé pour le chiffrement

    Return:
        resultat (String) : Le message déchiffré
    """
    resultat = ""
    alphabet = {lettre: i for i, lettre in enumerate("ABCDEFGHIJKLMNOPQRSTUVWXYZ")}

    for lettre in message_code:
        if lettre.isalpha():
            index = (alphabet[lettre] + decalage) % 26
            for cle, valeur in alphabet.items():
                if valeur == index:
                    lettre_decrypte = cle
                    break
        else:
            lettre_decrypte = lettre
        resultat += lettre_decrypte
    return resultat


def combinaison_possible_cesar(message_code):
    """
    Fonction qui permet de déchiffrer un message chiffré avec le chiffrement de César avec toute les combinaisons possibles

    Args :
        message_code (String) : Le message chiffré
    """
    for nb in range(27):
        message = dechiffrement_message1(message_code, nb)
        print("décalage de ", nb + 1, ": ", message)
```

La fonction dechiffrement_message1 permet de déchiffrer un message crypté avec un chiffrement César mais ne permet pas de savoir quel est le décalement. Pour savoir de combien est le déclement on utilise la deuxième fonction combinaison_possible_cesar qui permet de renvoyer pour chaque décodage un message avec son nombre de décalement (ici on le lance 27 fois pour toutes les lettres de l'alphabet).

En suivant l'indentation on peut voir que ce message décrypté forme un mot (verticalement) : PANGRAMME.

Deuxième message

Pour le deuxième message donné par le résultat du message 1 (voir fichier message_crypte2), il a fallu utiliser le chiffrement de vigenere pour déchiffrer ce message. Pour ce faire on utilise une fonction dechiffrement_message_2 qui permet de déchiffrer un message codé sous vigenere à partir d'une clé.

```
def dechiffrement_message_2(message_code, key):
    """
    Fonction qui déchiffre un message chiffré en utilisant le chiffrement Vigenère avec une clé donnée

    Args :
        message_code (String) : Le message codé
        key (String) : La clé utilisée pour le déchiffrement

    Return:
        resultat (String) : Le message déchiffré
    """
    resultat = ""
    alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    longueur_cle = len(key)
    i = 0
    for lettre in range(len(message_code)):
        if message_code[lettre].isalpha():
            decalage = alphabet.index(key[i % longueur_cle])
            lettre = (alphabet.index(message_code[lettre]) - decalage) % 26
            resultat += alphabet[lettre]
            i += 1
        else:
            resultat += message_code[lettre]
    return resultat
```

Comme clé nous avons utilisé le message trouver sur le message 1 "PANGRAMME". De plus grâce à ce message, nous savons que le message précédent sera déchiffré à l'aide d'une substitution.

Lorsque nous ouvrons le dépôt du deuxième message, nous trouvons le troisième message à déchiffrer.

Troisième message

Pour le troisième message (voir fichier message_crypte3), nous savons qu'il faudra utiliser la substitution, comme indiqué dans le message précédemment décodé,

Après avoir lu le poème plusieurs fois, nous avons pensé qu'il pourrait nous donner des indices sur l'alphabet de substitution utilisé pour chiffrer le message. Nous avons remarqué que la première phrase du poème ressemblait à un pangramme, c'est-à-dire qu'elle contenait toutes les lettres de l'alphabet. Nous avons donc décidé de conserver uniquement la première apparition de chaque lettre dans la phrase. Cette méthode nous a donné une liste de lettres, mais il nous manquait une lettre : le Y. Pour résoudre ce problème, nous avons décidé de déchiffrer le message 26 fois, en utilisant le Y à chaque fois à une position différente dans la liste afin de vérifier sur toutes les lettres et à la bonne position.

Nous avons donc codé 3 fonctions pour pouvoir décoder ce message :

```
def trouve_cle_alphabet(message_code):
    """ous aurez au total 3 messages à déchiffrer. Chaque message déchiffré vous
fournira un indice nécessaire pour déchiffrer le suivant. Les méthodes de chiffre-
ment utilisés ont été vu en cours, mais vous ne savez pas lesquelles!
Le projet se fera en binome. Il vous est demandé de réaliser un programme
python pour le déchiffrement.
    Fonction qui permet de trouver les caractères uniques dans le message et les stocke dans une liste pour former un alphabet
    Args:
        message_code (str): Message à partir duquel l'alphabet serra fait.
    Returns:
        resultat: Liste de caractères représentant un alphabet
    """
    lettre_vu = set()
    resultat = []
    for indice in range(len(message_code)):
        if message_code[indice] != " " and message_code[indice] not in lettre_vu:
            lettre_vu.add(message_code[indice])
            resultat.append(message_code[indice])
    return resultat

def decode_message_3(cle):
    """
    Fonction qui permet de réaliser un décalage des caractères de l'alphabet en insérant "Y" au début de la liste et décalant d'une p
    Puis appele la fonction decode_message_3_bis à chaque itération

    Args:
        cle (list): Liste représentant l'alphabet initial
    """
    for indice in range(26):
        if indice == 0:
            cle.insert(indice, "Y")
        else:
            cle.pop(indice - 1)
            cle.insert(indice, "Y")
        print("index :", indice)
        print(decode_message_3_bis(message_code, cle), "\n")

def decode_message_3_bis(message_code, cle):
    """
    Fonction qui permet de déchiffre un message à l'aide de l'alphabet donné

    Args:
        message (str): Message à déchiffrer
        cle (list): Liste représentant l'alphabet de substitution

    Returns:
        str: Message déchiffré
    """
    resultat = ''
    for lettre in message_code:
        if lettre in cle:
            index = cle.index(lettre)
            resultat += chr(index + 65)
        else:
            resultat += lettre
    return resultat
```

La fonction trouve_cle_alphabet() trouve les caractères uniques dans un message et les stocke dans une liste pour former un alphabet qui est la clé. La fonction decode_message_3() effectue un décalage des caractères de l'alphabet en insérant "Y" au début de la liste et en décalant d'une position à chaque itération. Elle appelle ensuite la fonction decode_message_3_bis() à chaque itération. La fonction decode_message_3_bis() déchiffre un message à l'aide de l'alphabet donné(clé). Pour cela, elle parcourt le message et remplace chaque lettre par la lettre de l'alphabet de substitution qui lui correspond.

En résumé, les fonctions trouve_cle_alphabet() et decode_message_3_bis() sont utilisées pour déchiffrer un message chiffré par substitution. La fonction decode_message_3() effectue un décalage de l'alphabet de substitution pour essayer de trouver le message original.

Voici les messages codé et décodé à la suite :

MESSAGE 1 CODÉ

BDQE PG OTQYUZ EQ OMOTQ GZ FDQEAD MOODAOTQ M GZ MDNDQ FAGF DQOAGHQDF P'AD ZQ ZQSXUSQ BME XM VQGZQ BAGOQ RQGUXXG SDMZP QEF EAZ EQODQF YMXSDQ EM FMUXXQ YQZGQ DAZPQE QF OAXADQQE EAZF XQE NMUQE CG'UX BADFQ MZUEQQE QF EGODQQE, XQGDE EMHQGDE EAZF RADFQE. YMUE MFFQZFUAZ M ZQ BME XQE ODACGQD, YQYQ EU XM RMUY FUDMUXXQ FQE QZFDMUXXQE, QZ MGOGZ OME FG ZQ PAUE EGOOAYNQD

MESSAGE 1 DÉCODÉ

PRES DU CHEMIN SE CACHE UN TRESOR ACCROCHE A UN ARBRE TOUT RECOUVERT D'OR NE NEGLIGE PAS LA JEUNE POUCE FEUILLU GRAND EST SON SECRET MALGRE SA TAILLE MENUE RONDES ET COLOREES SONT LES BAIES QU'IL PORTE ANISEES ET SUCREES, LEURS SAVEURS SONT FORTES. MAIS ATTENTION A NE PAS LES CROQUER, MEME SI LA FAIM TIRAILLE TES ENTRAILLES, EN AUCUN CAS TU NE DOIS SUCCOMBER

MESSAGE 2 CODÉ

AE IOW ZQBLXR WASIXQ WJR YKJ KGYUJAGY UU OXSLN TXRCUQYM IY IRCTQ HPNF RR RQBIIIGOFN XQ WTCEKK DQ OIH MHXDUDQW BAYNVUDQYM NR MRRPQD SU CXVMUQV HOHLWLQ CYT LRY GRQYMTRRY RPBMVXTVUES QF EXNFO UEHAMAEM RV MQEWPGR IRCTQ HTREOVRQ XE HUOYKIFGXXOA

MESSAGE 2 DÉCODÉ

LE VIF ZEPHIR JUBILE SUR LES KUMQUATS DU CLOWN GRACIEUX IL CACHE DANS LA REPETITION LE SECRET DE CES MURMURES MALHEUREUX NE GARDEZ DU PREMIER SOUFFLE QUE LES PREMIERES APPARITIONS ET AINSI DEVOILEZ LE MESSAGE CACHE DERRIERE LA SUBSTITUTION

MESSAGE 3 CODÉ

ETLWK, WKOD LWFX PLPSF! BF VKIF L ZKOTSRT FDC: FBRXFEFCH

MESSAGE 3 DÉCODÉ

BRAVO, VOUS AVEZ GAGNE! LE CODE A FOURNIR EST: ELIZEBETH

https://fr.wikipedia.org/wiki/Elizebeth_Friedman (https://fr.wikipedia.org/wiki/Elizebeth_Friedman)