Rapport

SAE Crypto 3.04

ETAVE Nathan

&

LEBEAUPIN Louis

21A/B

2023/2024

Université d'ORLÉANS

IUT ORLÉANS

INFORMATIQUE

Table des matières :

Rendu n°1

Message n°1

Pour déchiffrer ce massage, nous avons utilisé la méthode du chiffrement de César. Le chiffrement de César (chiffrement symétrique) est une technique de cryptage où chaque lettre dans le texte est déplacée d'un certain nombre de positions dans l'alphabet. Dans notre cas, le nombre de décalage à effectuer pour retrouver la lettre d'origine est de **12**.
Nous avons trois programmes qui permet de déchiffrer un message chiffré avec César, un qui nécessite de renseigner la clé, un autre avec une approche "stupide" qui va lister les 26 possibilitées de déchiffrement et une version "intelligente" qui va renvoyer le déchiffrement ayant le plus de correspondance avec des mots d'un dictionnaire français de (336451 mots).

Message chiffrés:

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 claire:

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 n°2

Pour déchiffrer le message 2, nous avons utilisé la méthode de Vigenère.
Dans le chiffrement de Vigenère (chiffrement symétrique), la clé utilisée pour chiffrer le message a la même longueur que le message lui-même. Chaque lettre du message est associée à une lettre correspondante dans la clé, et ce lien détermine le décalage appliqué à chaque caractère du message. Dans notre cas, la clé est "**PANGRAMME**" trouvé dans le message 1, nous allons donc multiplier la clé pour qu'elle ait la même taille que le message.

Message chiffrés:

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 claire:

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 n°3

Pour déchiffrer le message 3, nous avons utilisé la méthode de substitution monoalphabétique (chiffrement symétrique). Cette méthode consiste à mélanger un alphabet pour en faire une clé. On utilise ensuite cette clé pour chiffrer et déchiffrer un message. Pour ce faire, on remplace chaque lettre du message par la lettre correspondante dans la clé. Par exemple, si la clé est "NZERTYUIOPQSDFGHJKLMWXCVBA", alors la lettre "A" sera remplacée par "N", la lettre "B" sera remplacée par "Z", la lettre "C" sera remplacée par "E", etc. Pour déchiffrer le message, on fait l'opération inverse. Pour déchiffrer le message 3, nous avons utilisé la clé suivante :
"**RXUTBEJHDYOAWPVGMQNSLCKZIF**".

Message chiffrés:

EALOK, OKCT LOFX PLPSF! UF VKIF L ZKCASYA FTD: FUYXFEFDH

Message claire:

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

Rendu n°2

Partie 1: premières tentatives

Question n°1

- En supposant que RSA soit utilisé correctement, Eve peut-elle espérer en venir à bout? En vous appuyant sur votre cours, justifiez votre réponse.

En supposant que le RSA soit utilisé correctement, Eve ne pourra pas en venir à bout. Le chiffrement RSA est un algorithme de cryptographie asymétrique. Cet algorithmie utilise une clé publique pour chiffrer et une clé privée pour déchiffrer. La clé publique est connue de tous et la clé privée est connue uniquement de la personne qui veut déchiffrer le message. La clé publique est calculée à partir de la clé privée. Il est donc impossible de retrouver la clé privée à partir de la clé publique. Les seules façons de retrouver la clé privée sont de tester toutes les clés possibles ou de la demander à la personne qui la possède. Pour tester toutes les clés possibles, il faudrait énormément de temps car le nombre de clés possibles est très élevé.

Question n°2

- En quoi l’algorithme SDES est-il peu sécurisé? Vous justifierez votre réponse en analysant le nombre d’essai nécessaire à une méthode “force brute” pour retrouver la clé.

L'algorithme symétrique SDES est très peu sécurisé, car il utilise une clé très courte de 10 bits. Celui-ci a essentiellement pour but éducatif et non pas pour être utilisé dans un contexte réel. Comme cet algorithme utilise une clé de 10 bits, ce qui signifie qu'il y a 2^10 clés possibles. Cela représente 1024 clés possibles. La méthode "force brute" consiste à tester toutes les clés possibles. Il faudrait donc 1024 essais pour retrouver la clé, ce qui est très peu pour un algorithme de chiffrement. La moyenne de temps pour trouver la clé est de 512 essais. De plus, le nombre de tours de l'algorithme est de 2, ce qui est peu pour un algorithme de chiffrement, par comparaison, l'algorithme DES (le non simplifié) utilise 16 tours.

Question n°3

- Est-ce que le double SDES est-il vraiment plus sûr? Quelle(s) information(s) supplémentaire(s) Eve doit-elle récupérer afin de pouvoir espérer venir à bout du double DES plus rapidement qu'avec un algorithme brutal? Décrivez cette méthode astucieuse et précisez le nombre d'essais pour trouver la clé.

Le double SDES est en théorie plus sécurisé que le SDES, car il utilise deux clés de 10 bits. Cela signifie qu'il y a 2^10 puis 2^10 clés possibles, ce qui représente 2^20 clés possibles. La méthode "force brute" consiste à tester toutes les clés possibles. Il faudrait donc 2^20 essais au maximum pour retrouver les deux clés, ce qui est certes plus que le SDES, mais cela reste très peu pour un algorithme de chiffrement. Une méthode astucieuse pour retrouver les deux clés est d'avoir le message déchiffré et le message chiffré avec le double SDES. Il suffit ensuite de tester toutes les clés possibles sur le message déchiffré et de tester toutes les clés possibles sur le message chiffré avec le double SDES. Si les deux résultats sont identiques, alors il s'agit des deux clés utilisées pour le double SDES.

Partie 2: Un peu d'aide

Question n°1

- Le protocole a été mis à jour et utilise maintenant l'algorithme AES avec des clés de taille 256 bits. Est-ce vraiment un problème? Justifiez votre réponse.

La mise à jour du protocole n'est pas un problème en lui-même, car l'algorithme AES est un algorithme de chiffrement symétrique. Cela signifie que la clé de chiffrement et la clé de déchiffrement sont identiques. Il est donc possible de retrouver la clé de chiffrement à partir de la clé de déchiffrement. Mais la taille de la clé est de 256 bits, ce qui signifie qu'il y a 2^256 clés possibles. Cela représente environ 1.158 x 10^77 clés possibles. La mise à jour du protocole est donc un vrai problème, car la méthode "force brute" consistant à tester toutes les clés possibles prendrait énormément de temps. Il faudrait donc 1.158 x 10^77 essais au maximum pour retrouver la clé, ce qui à l'heure actuelle est impossible pour un algorithme de brute force. Par exemple, si un ordinateur avait la capacité de tester 1 milliard de clés par seconde, il faudrait environ 3.67 x 10^57 années pour tester toutes les clés possibles. Cela représente environ 2.6 x 10^47 fois l'âge de l'univers. Il est donc impossible de retrouver la clé de chiffrement à partir de la clé de déchiffrement.

Question n°2

- Nous allons tenter d’illustrer expérimentalement les différences entre les deux protocoles. Vous évaluerez:

1. Le temps d’exécution du chiffrement/déchiffrement d’un message avec chacun des deux protocoles.

En utilisant le code python des fichiers "experiences", nous avons mesuré le temps d'exécution du chiffrement/déchiffrement d'un message avec chacun des deux protocoles. Les graphiques générés pour comparer le chiffrement AES et RSA nous montre que le temps d'exécution du chiffrement AES est plus rapide que le temps d'exécution du chiffrement RSA. En moyenne le chiffrement AES prend environ 0.0001 secondes et le chiffrement RSA prend environ 0.0006 secondes, AES est donc 6 fois plus rapide que RSA.

Pour le déchiffrement, le graphique généré nous montre que le temps d'exécution du déchiffrement AES est plus rapide que le temps d'exécution du déchiffrement RSA. En moyenne le déchiffrement AES prend environ 0.001 secondes et le déchiffrement RSA prend environ 0.03 secondes, AES est donc 30 fois plus rapide que RSA.

Ce qui est logique car AES est un algorithme de chiffrement symétrique et RSA est un algorithme de chiffrement asymétrique. Cela signifie que la clé de chiffrement et la clé de déchiffrement sont identiques pour AES. Alors que pour RSA, la clé de chiffrement et la clé de déchiffrement sont différentes. Il est donc plus rapide de déchiffrer un message avec AES qu'avec RSA.

2. Le temps de cassage d’AES (même pour un cassage astucieux) si vous deviez l’exécuter sur votre ordinateur.

Avec un ordinateur ayant un processeur AMD Ryzen 9 5900HX cadencé à 4.6 GHz en fréquence de boost maximale avec 8 coeurs, il faudrait environ 2.467 x 10^69 secondes, soit environ 7.821 x 10^61 années pour tester toutes les clés possibles.

Détail des calculs:

- Nombre de clés possibles: 2^256
- Nombre de cycles d'horloge par seconde: 4.6 x 10^9
- Nombre de tours de déchiffrement pour AES-256: 14
- Nombre de cycles d'horloge pour un tour de déchiffrement AES sur un octet dans l'extension du jeu d'instructions x86 (AES-NI): 3.5
- Taille du bloc de données: 16 octets
- Nombre de coeurs: 8

En omettant les cycles intermédiaires, par exemple les cycles de chargement, de stockage, de comparaison, etc. On peut estimer le temps en secondes pour tester toutes les clés possibles avec la formule suivante:

(Nombre de clés possibles x Taille du bloc de données x Nombre de tours de déchiffrement pour AES-256 x Nombre de cycles d'horloge pour un tour de déchiffrement AES sur un octet) / (Nombre de cycles d'horloge par seconde x Nombre de coeurs) = Temps en secondes

((2^256) x 16 x 14 x 3.5) / ((4.6 x 10^9) x 8) = 2.467 x 10^69 secondes

Ensuite, pour convertir le temps en secondes en années, il faut diviser le temps en secondes par 3.154 x 10^7:

(2.467 x 10^69) / (3.154 x 10^7) = 7.821 x 10^61 années

Question n°3

- Il existe d’autres types d’attaques que de tester les différentes possibilités de clés. Lesquelles? Vous donnerez une explication succincte de l’une d’elles.

D'autres types d'attaques sont possibles, par exemple l'attaque de l'homme du milieu. L'attaquant se place entre l'émetteur et le récepteur et intercepte les messages. Il peut ensuite les modifier et les renvoyer au récepteur. L'attaquant peut également se faire passer pour l'émetteur et envoyer des messages au récepteur. L'attaquant peut également se faire passer pour le récepteur et envoyer des messages à l'émetteur.

Une autre attaque à envisager est l'attaque par canal auxiliaire. L'attaquant va alors utiliser des informations annexes pour retrouver la clé de chiffrement. Par exemple, l'attaquant peut utiliser les bruits électromagnétiques des processeur, la consommation électrique du processeur, etc. pour retrouver la clé de chiffrement.

Analyse des images

La solution de l'énigme est la suivante:
1110011101101101001100010011111110010010101110011001000001001100.

Pour retrouver la clé, nous sommes passés par plusieurs étapes. Tout d'abord, nous avons analysé le code hexadécimal des deux images. Nous avons alors remarqué que même si elles semblaient identiques, elles n'avaient pas le même code hexadécimal. Dès lors, nous avons fait un code python qui permettait d'afficher la valeur de chaque pixel des deux images. La première image avait des valeurs de pixel comprises entre 0 et 255 qui varient beaucoup entre pair et impair. Alors que la deuxième image avait également des valeurs de pixel comprises entre 0 et 255, mais qui varient beaucoup moins entre pair et impair, avec seulement quelques pixel pair. Nous avons alors compris qu'il s'agissait de la stéganographie du bit de poids faible. Nous avons donc écrit un code python pour extraire le bit de poids faible de chaque pixel des deux images. Mais pour cela, il a fallu trouver dans quelle sens parcourir les pixels. Ici il faut les parcourir de gauche à droite et de haut en bas. Ce code python renvoie alors une suite de 0 et de 1, mais seuls les 64 premiers bits sont intéressants. Nous avons donc extrait les 64 premiers bits de cette suite, ce qui nous a donné la clé ci-dessus.

Code python :

```python
from PIL import Image
def dechiffrer_image(path):
    image = Image.open(path)
    pixels = image.load()
    (largeur, hauteur) = image.size
    return ''.join([str(pixels[x, y] % 2) for y in range(hauteur) for x in range(largeur)])[:64]
```

Partie 4: Un peu de recul

Question n°1

- Alice et Bob utilisent toujours la même clé. Est-ce une bonne pratique?

Non, ce n'est pas une bonne pratique. Le mieux serait d'utiliser une clé différente pour chaque message. En effet, si par malheur un attaquant arrive à récupérer la clé, il pourra déchiffrer les messages passés et futurs. Il est donc nécessaire de faire tourner les clés. Cela permet de limiter les dégâts si un attaquant arrive à récupérer une clé.

Question n°2

- Le protocole PlutotBonneConfidentialité est inspiré d'un vrai protocole réseau. Décrivez la partie associée à la certification des clés qui est absente de PlutotBonneConfidentialité.

Le protocole PlutotBonneConfidentialité est inspiré du protocole réseau TSL/SSL. En effet, TSL/SLL utilise également un protocole cryptographique asymétrique pour échanger une clé de session. Cette clé va également servir à chiffrer les communications futures à l'aide d'un protocole cryptographique symétrique.
Mais le protocole PlutotBonneConfidentialité omet la partie de certification des clés. En effet, dans le protocole TSL/SSL, les clés sont certifiées par une autorité de certification. Cela permet de vérifier l'autenticité des clés. Dans le protocole PlutotBonneConfidentialité, les clés ne sont pas certifiées, il n'est donc pas possible de vérifier l'autenticité des clés.
Ce manque de certification signifie qu'une clé peut très bien être usurpée par un attaquant.

Question n°3

- Il n’y a pas que pour l’échange de mots doux q’un tel protocole peut se révéler utile. Donnez au moins deux autres exemples de contexte où cela peut se révéler utile.

Ce type de protocole peut être utile dans de nombreux domaines. Par exemple, dans le domaine bancaire où il est important de sécuriser les communications pour réaliser tout type d'opération bancaire. Ou encore dans le domaine militaire, les communications entre les différents acteurs du domaine doivent être sécurisées afin de ne pas divulguer d'informations sensibles.
Ce type de protocole peut également être utile dans le domaine de la santé, les données médicales sont confidentielles et doivent être protégées, ce type est donc un parfait candidat pour les sécuriser.

Question n°4

- Connaissez-vous des applications de messagerie utilisant des mécanismes de chiffrement similaires? (on parle parfois de chiffrement de bout en bout)? Citez-en au moins deux et décrivez brièvement les mécanismes cryptographiques sous-jacents.

L'application de messagerie Whatsapp utilise un chiffrement de bout en bout. Le chiffrement de bout en bout signifie que les messages sont chiffrés sur l'appareil de l'émetteur et déchiffrés sur l'appareil du récepteur. Les messages voyagent donc chiffrés sur le réseau. Cela permet de protéger les messages des attaques de l'homme du milieu par exemple.
Il existe également Telegram qui utilise un chiffrement MTProto. MTProto consiste à chiffrer les messages avec AES-IGE avant de les envoyer sur le réseau. C'est également un chiffrement de bout en bout, mais fait maison par Telegram.

Question n°5

- Récemment, différents projets de loi et règlements (CSAR, EARN IT Act) visent à inciter voire obliger les fournisseurs de services numériques à pouvoir déchiffrer (et donc analyser) les communications de leur.e.s utilisateur.rices. Discutez des arguments en faveur ou contre ces législations, notamment en matière de vie privée.

Les lois comme le CSAR et l'EARN IT Act visent à encadrer le contenu diffusé sur les réseaux, mais cette régulation peut être perçue comme une atteinte à la vie privée. Toutefois, en régulant ce contenu, elles cherchent aussi à protéger les utilisateurs contre des contenus inappropriés ou illégaux. Le souci majeur de ces lois réside dans le fait que les fournisseurs pourraient potentiellement décrypter les communications de tous leurs utilisateurs, ce qui ouvrirait la porte à la censure. Il est crucial que derrière cette volonté de protéger les utilisateurs, ne se cache pas une volonté de censure ou de surveillance de masse.

Dépot GitHub

https://github.com/LouisL18/SAE-Cryptographie

Répartition des tâches

	Nathan	Louis
Rendu n°1		
Message n°1	X	X
Message n°2	X	X
Message n°3	X	X
Rendu n°2		
Partie 1: premières tentatives		
Question n°1		X
Question n°2	X	
Question n°3	X	X
Partie 2: Un peu d'aide		
Question n°1	X	
Question n°2	X	
Question n°3	X	X
Partie 3: Analyse des messages	X	X
Partie 4: Un peu de recul		
Question n°1	X	X
Question n°2	X	X
Question n°3	X	X
Question n°4	X	X
Question n°5		X