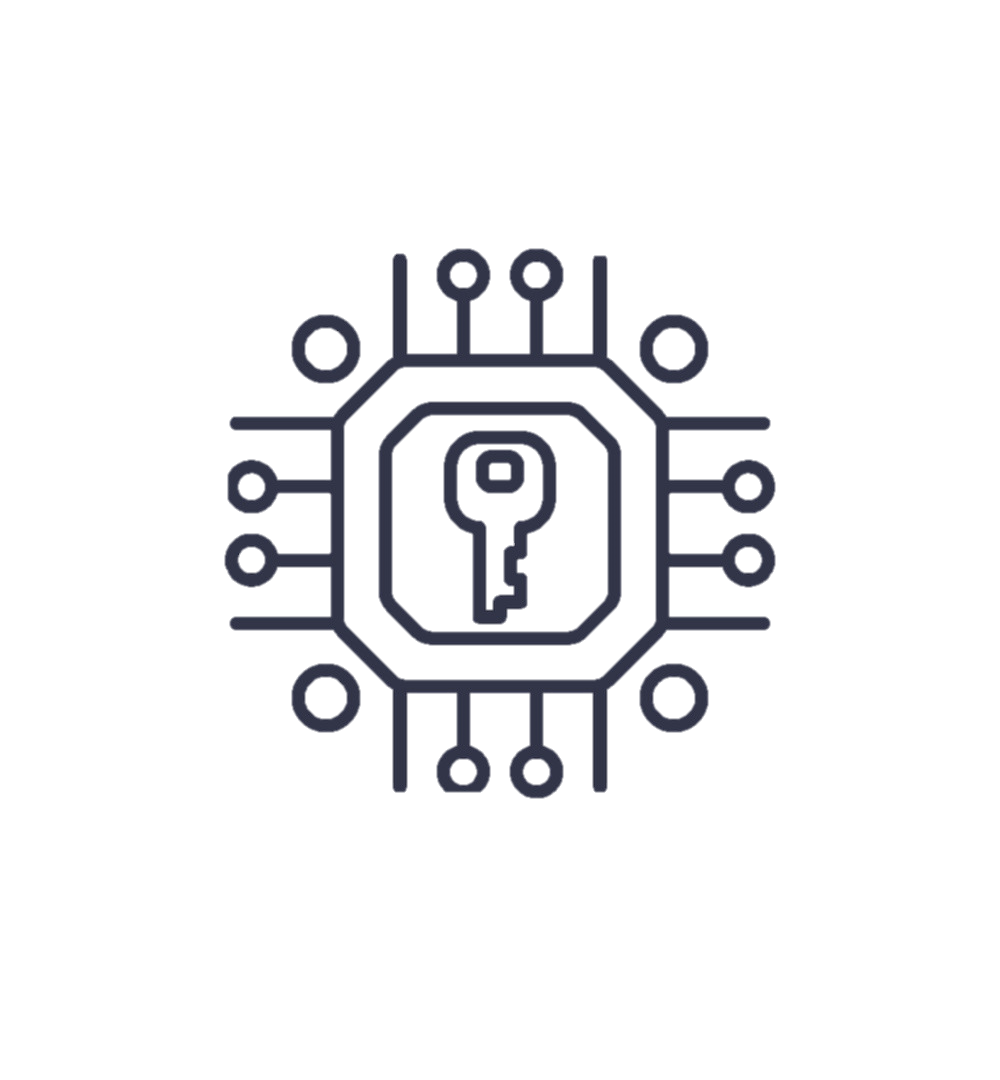
Rapport de SAE

SAE 3.04 : Cryptographie

DEMARET Sullivan - COULON Titouan

Groupe 22B

Réponses aux questions :

Partie 1 :

*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.*

On suppose que RSA est bien utilisé, donc que `p` et `q` soient d'une longueur suffisante et que la clé privée ne soit pas divulguée, Eve ne devrait pas compter sur le crackage brutal de RSA car ça lui bien trop de temps ( x millions d'années ) car elle devrait décomposer `n` en facteurs premiers.

*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 SDES utilise des clés de 10 bits de longueur, donc 2^10 = 1 024 clés disponibles, il faut faudrait donc 1 024/2 = 512 essais en moyenne pour "brute-forcer" le SDES avec un cassage brut simple, ce qui est très peu avec les ordinateurs actuels.

*3) Est-ce que double SDES est-il vraiment plus sur ? 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’essai pour trouver la clé*

L'algorithme double SDES utilise des clés de 10 bits de longueur, donc pour le double SDES et ses 2 clés, 2^(2\*10) = 1 048 576 clés disponibles, il faut faudrait donc 1 048 576/2 = 524288 essais en moyenne pour "brute-forcer" le double SDES avec un cassage brut simple, ce qui est mieux mais pas exceptionnel. Pour "brute-forcer" de façon plus astucieuse, il faudrait utiliser la technique du "meet in the middle" ce qui réduirait le nombre de clés à (2^10)\\*2 = 2048 essais, ce qui est très peu et montre que sa sécurité est faible mais il faut pour cela récupérer un message crypté et sa version en claire, ce qui est très compliqué pour Eve ( en théorie ). Le but de cette attaque est de tester toutes les combinaisons de clés pour chiffrer et déchiffrer et comparer avec le texte chiffré "original".

Expérience de test des fonctions :

Clé1 et Clé2 aléatoire entre 0 et 2^10, texte clair toujours le même et 10 essais pour calculer la moyenne.

encode(), decode() : Quelques millisecondes d’exécution pour crypter et decrypter arsene ou lettres en SDES ou double SDES.  
 cassage\_brutal () : Je n’ai pas pu laisser la fonction retrouver la clé mais elle prend au moins 2 heures (et surement beaucoup plus encore) pour retrouver arsene crypté en double SDES et moins de 10 secondes pour le simple SDES.

cassage\_astucieux() : environ une cinquantaine de secondes pour casser le double SDES sur chacun des textes en 1100 essais environ mais le problème du cassage astucieux est le temps et le stockage pas le nombre d’essais.

Partie 2 :

*1) Est-ce vraiment un problème ? Justifiez votre réponse.*

Oui, c’est un problème car d’un nombre de clés possibles égale à 2^10 soit 1024 on passe à 2^256 soit 1.16x10^77 clés possibles ce qui rend (quasi) impossible le cassage brutal.

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

*A) Le temps d’exécution du chiffrement/déchiffrement d’un message avec chacun des deux protocoles. Ici vous devez le mesurer expérimentalement et donc fournir le code Python associé.*

Ligne 100 environ du fichier SDES.py :   
Les durées ont été multipliés par 1000 pour avoir des valeurs différentes de 0 mais l’AES est environ 10 fois plus rapide.

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

Sachant que je peux calculer 1000 clés en 0.3s pour AES, cela fait environ 3000 clés testées par seconde et sachant qu’il y a 2^256 clés possibles, je mettrai 2^256/(3000\*60\*60\*24\*365) = 1.22x10^66 années à cracker AES en brute force, et même en cassage astucieux, cela représente une durée beaucoup trop grande ainsi qu’un stockage quasi infini ( pour une attaque meet in the middle ).

*Ici il faut uniquement estimer le temps nécessaire (sinon vous ne pourriez pas rendre votre rapport à temps !). Vous préciserez votre configuration et vous fournirez le détail des calculs*

*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 pourraient être par canal auxiliaire, évitant de s’attaquer à la complexité mathématique de ces fonctions mais nécessitant de trouver une faille dans la méthode de Alice et Bob et pas dans le protocole en lui-même mais sans accès à leurs machines, elle ne pourra pas utiliser d’autres méthodes.

Partie 3 : /

Partie 4 :  
  
Répartitions des tâches :

Sullivan DEMARET : Partie 2 ( Images ) + Partie 3 + Partie 4  
 Titouan COULON : Partie 1 + Partie 2 (AES)