Évaluation : éléments de la cryptographie et de la cryptanalyse

Informations

**Évaluation** : 25 % de la session

**Type de travail** : individuel

**1. Chiffrement de césar.**

**Question**

Vous devez expliquer le problème **majeur** de sécurité associé avec l'algorithme du chiffrement de César. Et, quel principe est démontré par ce problème ?

**Réponse :**

* Le problème majeur de sécurité que l’on a avec l’algorithme du chiffrement de César est la faible résistance aux attaques par force brute. Étant donné que nous avons seulement 26 lettres possibles et que l’algorithme de César repose sur un décalage fixe des lettres de l’alphabet, il en ressort qu’une personne peut utiliser toutes les possibilités.
* Le principe illustré est celui de Kerckhoff qui nous fait comprendre que la clé est importante et non l’algorithme et non le fonctionnement.

**Programmation**

**Capture d’écran 1.**

**Capture d’écran 2.**

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Le contenu généré par l’IA peut être incorrect.

**2. Chiffrement de substitution.**

**Question**

Qu'est-ce qui n'est pas caché par le chiffrement par substitution ? Quel type d'attaque ça nous permet de faire ?

**Réponse :**

* **La fréquence des lettres**
* **Attaque par analyse fréquentielle**

**Programmation**

Vous devez inclure un script, nommé SubstitutionCypher.py, qui démontre une implémentation du chiffrement de substitution. Le script doit avoir une fonction qui génère un flux de clés, une fonction qui chiffre et une qui déchiffre le message.

Inclure une capture d’écran démontrant l’exécution de votre script.

**Capture d’écran 3.**

**3. Chiffrement One Time Pad.**

**Question**

Pourquoi cet algorithme est-il considéré comme incassable ?

**Réponse :**

* One Time Pad est considéré comme incassable car chaque message en claire est possible. Il n’y a pas de moyen en cryptanalyse de déterminer lequel des messages est le bon. En effet, pour plusieurs clés utilisées, on obtient un message lisible sans toutefois savoir si c’est le message initial. On aura donc une impression d’avoir cassé l’algorithme pourtant ce n’est pas la bonne clé.

**Programmation**

Vous devez inclure un script, nommé OneTimePad.py, qui démontre une implémentation du chiffrement One Time Pad. Le script doit avoir une fonction qui génère un flux de clés, une fonction XOR qui chiffre et déchiffre le message.

Inclure une capture d’écran démontrant l’exécution de votre script.

**Capture d’écran 4.**

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Le contenu généré par l’IA peut être incorrect.

**4. Chiffrement de flux**

**Question**

Quels compromis avons-nous dû faire par rapport au One Time Pad ?

**Réponse :**

* Possibilité de réutiliser le flux de clés
* Aucune exigence sur le flux de clé.
* Par conséquent, pas de vrai aléatoire

**Programmation**

Vous devez inclure un script, nommé StreamCipher1.py, qui démontre une implémentation du chiffrement de flux. Le script doit avoir une classe qui génère un flux de clés, une fonction qui chiffre et déchiffre le message.

Inclure une capture d’écran démontrant l’exécution de votre script.

**Capture d’écran 5.**

Vous devez inclure un script, nommé StreamCipher2.py, qui démontre une attaque de texte connu sur l’implémentation du chiffrement de flux. Une attaque où Eve fait envoyée à Alice un message à Bob, un message qu’elle connait le contenu pour pouvoir se générer un flux de clé. Ce qui, par la suite, va lui permettre de déchiffrer une partie des autres messages envoyés entre Alice et Bob. Le script doit afficher le message d’Eve qu’Alice envoie à Bob, le message déchiffré de Bob, un deuxième message d’Alice à Bob et la partie qu’Eve a réussi à déchiffrer.

Inclure une capture d’écran démontrant l’exécution de votre script.

**Capture d’écran 6.**

**5. Chiffrement par blocs.**

**Question**

Quel est le problème avec le mode ECB du chiffrement par bloc ?

**Réponse :**

**Programmation**

Vous devez inclure un script, nommé DES.py, qui démontre une implémentation du chiffrement DES avec le mode CBC. Le script doit afficher les blocs chiffrés de votre message ainsi que le message déchiffré.

Inclure une capture d’écran démontrant l’exécution de votre script.

**Capture d’écran 7.**

**6. La distribution de clés Diffie-Helleman.**

**Question**

Même s'ils n'en font pas mention dans leur article, quelle notion fut introduite par Diffie et Hellman avec leur schéma de distribution de clé ?

**Réponse :**

**Programmation**

Vous devez inclure un script, nommé DiffieHellman.py, qui démontre une implémentation de l’échange de clé Diffie-Helleman. Le script doit avoir une fonction pour trouver des nombres premiers et une fonction pour trouver un générateur. Le script doit également afficher le nombre *j* d’Alice, le nombre *k* de Bob et les clés communes, *g\_ab*, générés par Alice et Bob.

Inclure une capture d’écran démontrant l’exécution de votre script.

**Capture d’écran 8.**

**7. RSA.**

**Question**

Donner deux raisons pourquoi on n'utilise pas, de manière générale, les algorithmes à clé publique pour chiffrer les messages.

**Réponse :**

* Les algorithmes à clé publique ne peuvent chiffrer que de petits volumes de données.
* Les algorithmes à clé publique sont beaucoup plus lents comparé aux algorithmes à clé symétrique.

**Programmation**

**Capture d’écran 9.**

Vous devez inclure un script, nommé RSA2.py, qui démontre comment l’on peut casser RSA avec la factorisation des entiers. Le script doit afficher les valeurs *p*, *q*, *e*, *n* et *d* d’Alice, les valeurs *p* et *q* trouver par Eve, le *lambda\_n* trouver par Eve, la clé secrète *d* d’Alice trouvé par Eve, ainsi que le message déchiffré par Eve.

Inclure une capture d’écran démontrant l’exécution de votre script.

**Capture d’écran 10.**