Rapport de Pratique : Cryptographie Symétrique et Asymétrique

Rapport de Pratique : Cryptographie Symétrique avec GPG et OpenSSL

1. Objectif du travail

Ce travail a pour but de comprendre le fonctionnement du chiffrement symétrique, d’appliquer deux outils standards (GPG et OpenSSL) pour chiffrer et déchiffrer un fichier texte, et d’analyser le rôle des clés et des algorithmes dans la protection de la confidentialité.

2. Rappel théorique

• Chiffrement symétrique

Le chiffrement symétrique utilise la même clé pour chiffrer et déchiffrer. Exemples : AES, DES, 3DES.

• Algorithme AES

AES fonctionne en blocs de 128 bits (16 octets), avec des clés de 128, 192 ou 256 bits. Chaque bloc subit plusieurs étapes : SubBytes, ShiftRows, MixColumns, AddRoundKey.

• Outils utilisés

- GPG (GNU Privacy Guard) : outil de chiffrement conforme à la norme OpenPGP.

- OpenSSL : outil pour manipuler des certificats et réaliser du chiffrement.

3. Prérequis et Environnement

- Création d'un fichier texte nommé message.txt contenant : "Ceci est un test de cryptographie."

- Machine virtuelle (AttackBox / Kali Linux) avec GPG et OpenSSL installés.

4. Chiffrement avec GPG

• Commande standard (AES256) :

gpg --symmetric --cipher-algo AES256 message.txt

Fichier généré : message.txt.gpg

• Version ASCII (lisible texte) :

gpg --armor --symmetric --cipher-algo AES256 message.txt

Fichier généré : message.txt.asc

• Déchiffrement :

gpg --output message\_original.txt --decrypt message.txt.gpg

• Comparaison :

diff message.txt message\_original.txt

5. Chiffrement avec OpenSSL

• Commande de base :

openssl aes-256-cbc -e -in message.txt -out message\_chiffre

• Version sécurisée avec PBKDF2 et itérations :

openssl aes-256-cbc -pbkdf2 -iter 10000 -e -in message.txt -out message\_chiffre

• Déchiffrement :

openssl aes-256-cbc -d -in message\_chiffre -out message\_original.txt

• Comparaison :

diff message.txt message\_original.txt

6. Chiffrement Asymétrique (Task 3)

• Définition

Le chiffrement asymétrique repose sur une paire de clés : une clé publique (diffusée) et une clé privée (secrète).

• Confidentialité

- Alice chiffre avec la clé publique de Bob → seul Bob peut déchiffrer avec sa clé privée.

- Bob répond avec la clé publique d’Alice → seule Alice peut lire avec sa clé privée.

• Authenticité et Intégrité

- Bob chiffre avec sa clé privée → tout le monde peut vérifier avec sa clé publique.

• Algorithme RSA

1. Choisir p et q, puis calculer N = p × q

2. Calculer ϕ(N) = (p-1)(q-1)

3. Choisir e et d tel que e×d ≡ 1 mod ϕ(N)

4. Clé publique : (N, e), Clé privée : (N, d)

• Commandes OpenSSL :

openssl genrsa -out private-key.pem 2048

openssl rsa -in private-key.pem -pubout -out public-key.pem

openssl rsa -in private-key.pem -text -noout

• Chiffrement :

openssl pkeyutl -encrypt -in plaintext.txt -out ciphertext -inkey public-key.pem -pubin

• Déchiffrement :

openssl pkeyutl -decrypt -in ciphertext -inkey private-key.pem -out decrypted.txt

7. Conclusion

Comparaison des deux familles :

- Symétrique : rapide mais peu scalable.

- Asymétrique : lent mais garantit intégrité, authenticité, confidentialité.

Pratique : on combine les deux (clé échangée asymétriquement, puis données chiffrées symétriquement).

Fichiers générés :

message.txt, message.txt.gpg, message.txt.asc, message\_chiffre, message\_original.txt,

private-key.pem, public-key.pem, ciphertext, decrypted.txt