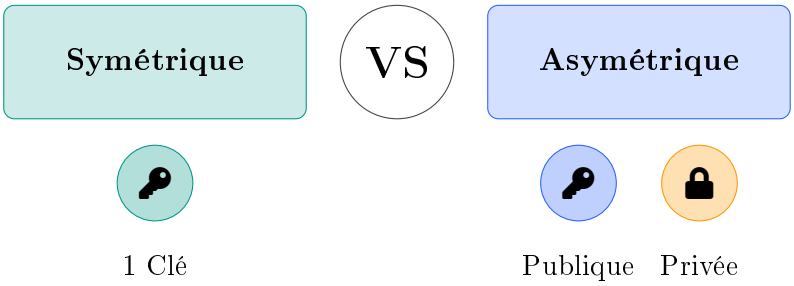
Cryptographie

Symétrique vs Asymétrique



Démonstration Pratique

Chiffrement et Déchiffrement de Fichiers

Auteur : Ahmed Dinari
Date : Novembre 2025
Module : Sécurité Informatique

Table des matières

1 Introduction **2**

2 Chiffrement Symétrique **2**
2.1 Principe 2
2.2 Algorithme AES (Advanced Encryption Standard) 2
2.3 Mode CBC (Cipher Block Chaining) 2
2.4 Code Python - Chiffrement Symétrique 3

3 Chiffrement Asymétrique **3**
3.1 Principe 3
3.2 Algorithme RSA 4
3.3 Padding OAEP 4
3.4 Code Python - Chiffrement Asymétrique 4

4 Comparaison : Symétrique vs Asymétrique **5**
4.1 Avantages et Inconvénients 5

5 Démonstration Pratique **5**
5.1 Scénario 5
5.2 Résultats 6

6 Chiffrement Hybride (Utilisation Réelle) **6**

7 Conclusion **7**

1 Introduction

La **cryptographie** est l'art de protéger les informations en les transformant en un format illisible pour quiconque ne possède pas la clé de déchiffrement.

Objectifs de la Cryptographie

- **Confidentialité** : Seules les personnes autorisées peuvent lire le message
- **Intégrité** : Le message n'a pas été modifié
- **Authentification** : L'identité de l'expéditeur est vérifiée
- **Non-répudiation** : L'expéditeur ne peut pas nier avoir envoyé le message

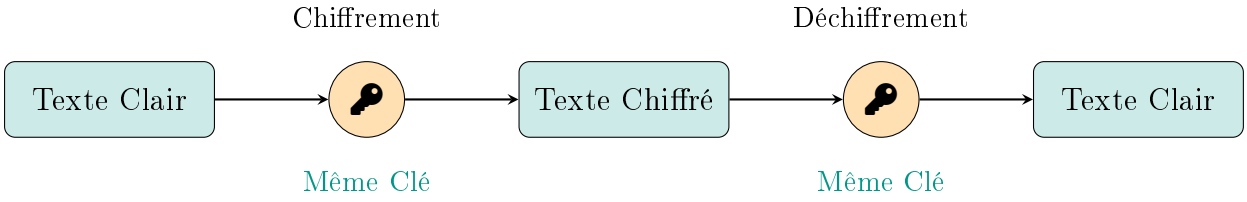
Il existe deux grandes familles de chiffrement :

1. **Chiffrement Symétrique** (AES, DES, 3DES)
2. **Chiffrement Asymétrique** (RSA, ECC, DSA)

2 Chiffrement Symétrique

2.1 Principe

Le chiffrement symétrique utilise **une seule clé** pour chiffrer ET déchiffrer les données.



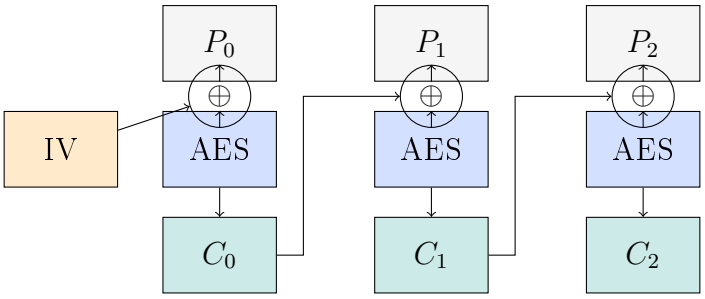
2.2 Algorithme AES (Advanced Encryption Standard)

AES est l'algorithme symétrique le plus utilisé aujourd'hui.

Propriété	AES-128	AES-192	AES-256
Taille de clé	128 bits	192 bits	256 bits
Nombre de tours	10	12	14
Taille de bloc	128 bits	128 bits	128 bits
Sécurité	Haute	Très haute	Maximale

2.3 Mode CBC (Cipher Block Chaining)

Dans notre démonstration, nous utilisons le mode **CBC** :



— **IV** (Initialization Vector) : Vecteur aléatoire pour le premier bloc
— Chaque bloc dépend du bloc précédent ⇒ plus de sécurité

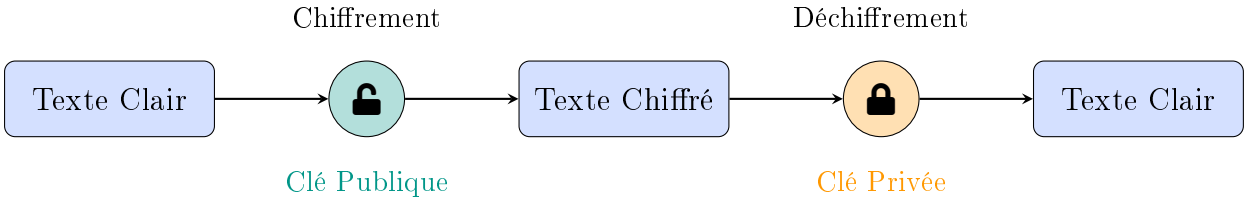
2.4 Code Python - Chiffrement Symétrique

```python
from cryptography.hazmat.primitives.ciphers import Cipher,
    algorithms, modes
import os

# Generer une cle AES de 256 bits
key = os.urandom(32)

# Generer un IV aleatoire
iv = os.urandom(16)

# Creer le cipher AES-CBC
cipher = Cipher(algorithms.AES(key), modes.CBC(iv))

# Chiffrer
encryptor = cipher.encryptor()
ciphertext = encryptor.update(padded_data) + encryptor.finalize()

# Dechiffrer
decryptor = cipher.decryptor()
plaintext = decryptor.update(ciphertext) + decryptor.finalize()
```

3 Chiffrement Asymétrique

3.1 Principe

Le chiffrement asymétrique utilise **deux clés différentes** :
— **Clé Publique** : Partagée avec tout le monde, utilisée pour **chiffrer**
— **Clé Privée** : Gardée secrète, utilisée pour **déchiffrer**



3.2 Algorithme RSA

RSA (Rivest-Shamir-Adleman) est basé sur la difficulté de factoriser de grands nombres premiers.

Mathématiques de RSA

Génération des clés :

1. Choisir deux grands nombres premiers p et q
2. Calculer $n = p \times q$
3. Calculer $\phi(n) = (p-1)(q-1)$
4. Choisir e tel que $\gcd(e, \phi(n)) = 1$ (généralement $e = 65537$)
5. Calculer $d = e^{-1} \mod \phi(n)$

Clés :

— Clé publique : (n, e)
— Clé privée : (n, d)

Opérations :

$$\text{Chiffrement} : C = M^e \mod n$$
$$\text{Déchiffrement} : M = C^d \mod n$$

3.3 Padding OAEP

Pour plus de sécurité, RSA utilise le padding **OAEP** (Optimal Asymmetric Encryption Padding) :

— Ajoute de l'aléatoire au message
— Protège contre les attaques par texte choisi
— Utilise des fonctions de hachage (SHA-256)

3.4 Code Python - Chiffrement Asymétrique

```python
from cryptography.hazmat.primitives.asymmetric import rsa, padding
from cryptography.hazmat.primitives import hashes

# Generer une paire de cles RSA 2048 bits
private_key = rsa.generate_private_key(
    public_exponent=65537,
    key_size=2048
)
public_key = private_key.public_key()

# Chiffrer avec la cle publique
ciphertext = public_key.encrypt(
    message,
    padding.OAEP(
        mgf=padding.MGF1(algorithm=hashes.SHA256()),
        algorithm=hashes.SHA256(),
        label=None
```

```
    )
)

# Dechiffrer avec la cle privee
plaintext = private_key.decrypt(ciphertext, padding.OAEP(...))
```

4 Comparaison : Symétrique vs Asymétrique

Critère	Symétrique	Asymétrique
Nombre de clés	1 (même clé)	2 (publique/privée)
Vitesse	Rapide ★★★	Lent ★
Taille des clés	128-256 bits	2048-4096 bits
Échange de clés	Difficile	Facile
Taille max données	Illimitée	190 octets (RSA-2048)
Algorithmes	AES, DES, 3DES	RSA, ECC, DSA
Utilisation	Fichiers, Disques	Signatures, HTTPS

4.1 Avantages et Inconvénients

Symétrique

Avantages :
- ✓ Très rapide
- ✓ Efficace pour gros volumes
- ✓ Clés courtes

Inconvénients :
- × Échange de clé difficile
- × Gestion des clés complexe

Asymétrique

Avantages :
- ✓ Échange de clé sécurisé
- ✓ Signatures numériques
- ✓ Non-répudiation

Inconvénients :
- × Lent
- × Clés longues
- × Limite de taille

5 Démonstration Pratique

5.1 Scénario

Dans notre démonstration, nous avons :

1. **Créé un fichier confidentiel** : `fichier_original.txt`
2. **Chiffré le fichier** avec AES (symétrique) ⇒ `fichier_chiffre.txt`
3. **Déchiffré le fichier** ⇒ `fichier_dechiffre.txt`

5.2 Résultats

Fichier Original

```
==========================================
FICHIER CONFIDENTIEL
==========================================
Nom: Ahmed Dinari
Sujet: Travaux Pratiques - Sécurité
Date: Novembre 2025
```

Fichier Chiffré (Illisible)

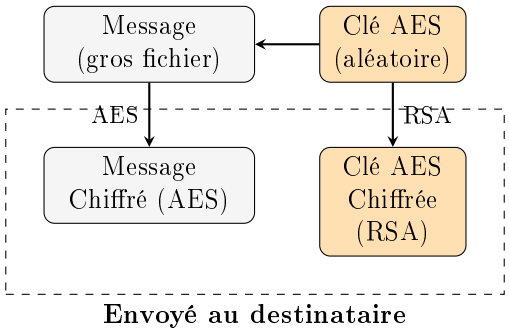
```
tlJzhikFecZWMIfhqcXY12iKbn/AfPiXSa5dvu
onT2zmEflP0N6/TlZV8SU3+11CrClSc7oZkEZr
55XG8Xkl2wQy...
```

Fichier Déchiffré (Identique à l'original)

```
==========================================
FICHIER CONFIDENTIEL
==========================================
Nom: Ahmed Dinari
Sujet: Travaux Pratiques - Sécurité
Date: Novembre 2025
```

6 Chiffrement Hybride (Utilisation Réelle)

En pratique, on combine les deux méthodes :



Exemple : HTTPS (TLS/SSL)

1. Le serveur envoie sa **clé publique RSA**
2. Le client génère une **clé AES aléatoire**
3. Le client **chiffre la clé AES** avec RSA et l'envoie
4. Toutes les données sont **chiffrées avec AES**

⇒ Combinaison de la **rapidité d'AES** et de la **sécurité de RSA** !

7 Conclusion

Points Clés à Retenir

- **Symétrique** : 1 clé, rapide, pour les fichiers
- **Asymétrique** : 2 clés, sécurisé, pour l'échange
- **Hybride** : Combine les deux (HTTPS, Email)

Merci pour votre attention !

Questions ?