Documentation des TPs de Cryptographie

Zakariae Ouahdaho et Amine Krati

28 novembre 2025

Dépôt GitHub :
`https://github.com/ZakariaeOuahdaho/tp-crypto`

Introduction

Ce document présente l'organisation et le contenu des travaux pratiques de cryptographie réalisés. Chaque exercice est implémenté dans un fichier Python dédié disponible sur le dépôt GitHub.

1 TP 1 : Chiffrements Classiques

1.1 Exercice 1 : Chiffrement de César

Fichier : `cesar.py`
Description : Implémentation du chiffrement de César avec décalage paramétrable.
Fonctionnalités :

— Chiffrement d'une chaîne de caractères

— Déchiffrement avec le décalage utilisé

— Décalage paramétrable (variable)

1.2 Exercice 2 : Chiffrement de Vigenère

Fichier : `vigenere.py`
Description : Implémentation de l'algorithme de Vigenère.
Fonctionnalités :

— Chiffrement avec clé alphabétique

— Déchiffrement avec la même clé

— Exemple : message `"mastercsuemf"` avec clé `"clefvigenere"`

1.3 Exercice 3 : Transposition Classique

Fichier : `transposition.py`
Description : Chiffrement par transposition avec clé $(4, \{3, 2, 5, 1, 4\})$.
Fonctionnalités :

— Chiffrement par transposition de colonnes

— Déchiffrement avec la même clé

— Clé : longueur 4, ordre de permutation $\{3, 2, 5, 1, 4\}$

2 TP 2 : Chiffrement Symétrique Moderne

2.1 Exercice 1 : Utilisation de DES/3DES

Fichier : `des_tool.py`
Description : Utilisation d'OpenSSL ou d'une bibliothèque pour DES/3DES.
Fonctionnalités :

— Chiffrement de fichiers texte

— Échange de clés entre partenaires

— Déchiffrement avec la clé partagée

2.2 Exercice 2 : Implémentation DES

Fichier : `des_implementation.py`
Description : Implémentation complète de l'algorithme DES.
Fonctionnalités :

— Implémentation des 16 rounds DES

— Permutations initiales et finales

— Fonction de Feistel

— S-boxes et P-boxes

3 TP 4 : Cryptographie Asymétrique (RSA)

3.1 Exercice 1 : Échange de Clés RSA

Fichier : `rsa_exchange.py`
Description : Génération de clés RSA et échange sécurisé.
Fonctionnalités :

— Génération de paires de clés publique/privée avec OpenSSL

— Chiffrement de fichiers avec clé publique

— Déchiffrement avec clé privée

— Échange sécurisé entre partenaires

3.2 Exercice 2 : Cryptanalyse RSA

Fichier : `rsa_cryptanalysis.py`
Description : Décryptage du message chiffré RSA par factorisation.
Données :

— Message chiffré : `997593903573`

— Clé publique : $(e = 7, n = 1037594094337)$

Étapes :

1. Factorisation de n en deux nombres premiers p et q
2. Calcul de $\phi(n) = (p-1)(q-1)$
3. Calcul de la clé privée d tel que $e \cdot d \equiv 1 \pmod{\phi(n)}$
4. Déchiffrement : $M = C^d \bmod n$
5. Conversion du message en ASCII (par paires de chiffres)

Structure du Dépôt

```
tp-crypto/
          README.md
          TP1/
                cesar.py
                vigenere.py
                transposition.py
          TP2/
                des_tool.py
                des_implementation.py
          TP4/
              rsa_exchange.py
              rsa_cryptanalysis.py
```

Listing 1 – Arborescence du projet

Utilisation

Chaque fichier Python peut être exécuté indépendamment. Exemple :

```
# Chiffrement de C sar
python TP1/cesar.py

# Chiffrement de Vigen re
python TP1/vigenere.py

# Cryptanalyse RSA
python TP4/rsa_cryptanalysis.py
```

Dépendances

— Python 3.x

— Bibliothèques : `cryptography`, `pycryptodome`

— OpenSSL (pour les exercices utilisant des outils externes)

Conclusion

Ce dépôt regroupe l'ensemble des implémentations des algorithmes de cryptographie étudiés, couvrant les chiffrements classiques (César, Vigenère, Transposition), les algorithmes symétriques modernes (DES) et la cryptographie asymétrique (RSA).