

Rapport du Projet : CryptoTool



Introduction

Cette application a été développée pour offrir une suite complète d'outils de cryptographie et de gestion de données sensibles. Elle permet à l'utilisateur d'effectuer diverses opérations cryptographiques, notamment le hachage de données, l'encodage et le décodage, le cracking de mots de passe, ainsi que le chiffrement et le déchiffrement à l'aide des algorithmes RSA et ElGamal. L'objectif principal de cette application est de fournir une interface simple et intuitive pour ces opérations complexes, tout en garantissant la sécurité des informations manipulées.

L'architecture de l'application est composée d'un backend qui gère toute la logique cryptographique, et d'un frontend développé en Tkinter pour l'interface utilisateur graphique (GUI). Cette séparation permet une évolution facile de l'application, tout en maintenant une interface claire et conviviale.

1. Structure du Projet :

```
|── main.py      # Fichier principal pour démarrer l'application  
|──gui_hashing.py  # Interface pour le hachage (MD5, SHA-1, SHA-256, etc.)  
|──gui_encoding.py # Interface pour l'encodage et décodage (Base64, UTF-8, etc.)  
|──gui_cracker.py # Interface pour cracker des mots de passe avec des hash et un wordlist  
|──gui_asymmetric.py # Interface pour le chiffrement/déchiffrement RSA et ElGamal  
|──Encoding.py    # Classe pour gérer les méthodes d'encodage  
|──gui_symmetric.py # (Optionnel) Fonctions utilitaires pour la cryptographie (si nécessaire)  
|──wordlist.txt   # Exemple de fichier de mots de passe pour cracker (wordlist)  
|──Hashing.py     # Gère la logique de génération de hachage pour les algorithmes MD5, SHA-1, SHA-256, etc.
```

```
|──SymmetricEncryption.py # Gère les fonctions de chiffrement/déchiffrement  
symétrique (si ajouté, par exemple AES, Salsa20)  
|──elgamal.py      # Logique de chiffrement et déchiffrement ElGamal  
└──ASymmetricEncryption.py # Logique de chiffrement/déchiffrement RSA et  
signature/validation
```

2. Architecture du Projet

Backend

Le backend de l'application est responsable de la gestion de toutes les fonctions cryptographiques et de manipulation des données. Il comprend plusieurs modules Python dédiés à différentes fonctionnalités :

- Hashing (Hashing.py) : Ce module gère les algorithmes de hachage comme MD5, SHA-1, SHA-256, etc. Il permet de générer des valeurs de hachage à partir de chaînes de texte.
- Encoding (Encoding.py) : Ce module contient les fonctions d'encodage et de décodage (Base64, UTF-8, ASCII, etc.).
- SymmetricEncryption.py : Ce fichier (optionnel) est destiné à gérer les algorithmes de chiffrement symétrique, comme AES et Salsa20.
- Elgamal.py et ASymmetricEncryption.py : Ces fichiers contiennent la logique pour le chiffrement, le déchiffrement et la gestion des signatures RSA et ElGamal.

Frontend

Le frontend de l'application est développé avec la bibliothèque Tkinter, permettant de créer des interfaces graphiques interactives. Les fichiers suivants sont responsables de l'affichage des interfaces utilisateurs et de l'interaction avec le backend :

- main.py : Fichier principal pour démarrer l'application et lancer l'interface graphique.
- gui_hashing.py : Interface pour le hachage des données (MD5, SHA-1, etc.).
- gui_encoding.py : Interface pour l'encodage et décodage (Base64, UTF-8, etc.).
- gui_cracker.py : Interface pour cracker des mots de passe à partir de hachages.
- gui_asymmetric.py : Interface pour le chiffrement/déchiffrement RSA et ElGamal.

3. Description des Fonctionnalités

L'application inclut plusieurs fonctionnalités :

- Hachage (Hashing) Permet à l'utilisateur de hacher une chaîne de texte avec différents algorithmes comme MD5, SHA-1 et SHA-256. Le résultat est affiché immédiatement après l'entrée du texte.
- Encodage/Décodage :Permet d'encoder ou de décoder des données dans différents formats comme Base64, UTF-8 et ASCII. Cela peut être utile pour manipuler des chaînes de caractères avant de les envoyer ou de les stocker.
- Cracker de Mots de Passe :Cette fonctionnalité permet à l'utilisateur de cracker des mots de passe en comparant un hash donné avec un fichier de mots de passe (wordlist). L'utilisateur peut sélectionner un hash, choisir un fichier de mots de passe et démarrer le processus de recherche.
- Chiffrement/Déchiffrement Asymétrique : Utilise les algorithmes RSA et ElGamal pour chiffrer et déchiffrer des messages. L'utilisateur peut également signer un message et vérifier la signature.

4. Méthodologies utilisées et Bibliothèques

Méthodologie

Le projet a été développé en utilisant une approche de développement agile, avec une concentration sur la modularité et la réutilisation du code. Chaque fonctionnalité a été implémentée de manière indépendante, ce qui a permis une gestion efficace du projet et une facilité de tests. Le développement a été réalisé dans un environnement local avec Python et Tkinter pour l'interface graphique.

Bibliothèques utilisées

Bibliothèques standards (incluses avec Python)

1. **tkinter**

Utilisée pour créer l'interface graphique (fenêtres, boutons, champs, etc.).

2. **hashlib**

Sert à appliquer des algorithmes de hachage comme MD5, SHA-1, SHA-256, etc.

3. **base64**

Utilisée pour encoder et décoder des chaînes en base64.

4. **os**

Sert à interagir avec le système de fichiers, comme l'ouverture de fichiers.

5. **math**

Fournit des fonctions mathématiques comme puissance et modulo.

6. **random**

Utilisée pour générer des nombres aléatoires, utile notamment pour ElGamal.

7. **re**

Sert à manipuler des expressions régulières, par exemple pour des décodages personnalisés.

8. **string**

Fournit des ensembles de caractères utiles (lettres, chiffres, etc.).

9. **sys**

Utilisée pour interagir avec le système (ex. sys.argv ou sys.exit()).

10. secrets

Permet de générer des mots de passe de manière sécurisée (plus sécurisé que random).

 **Bibliothèques externes (à installer via pip)**

11. **pycryptodome** (Crypto)

Permet de gérer les algorithmes RSA, AES, Salsa20, ElGamal, génération de clés, etc.

► Installation : pip install pycryptodome

12. **pyfiglet**

Utilisée pour afficher du texte stylisé en ASCII dans le terminal (ex. en-têtes).

► Installation : pip install pyfiglet

13. **pyinputplus**

Permet de faire des saisies utilisateur sécurisées dans le terminal.

► Installation : pip install pyinputplus

14. **stdiomask**

Sert à masquer la saisie de mot de passe dans le terminal (comme les étoiles ***).

► Installation : pip install stdiomask

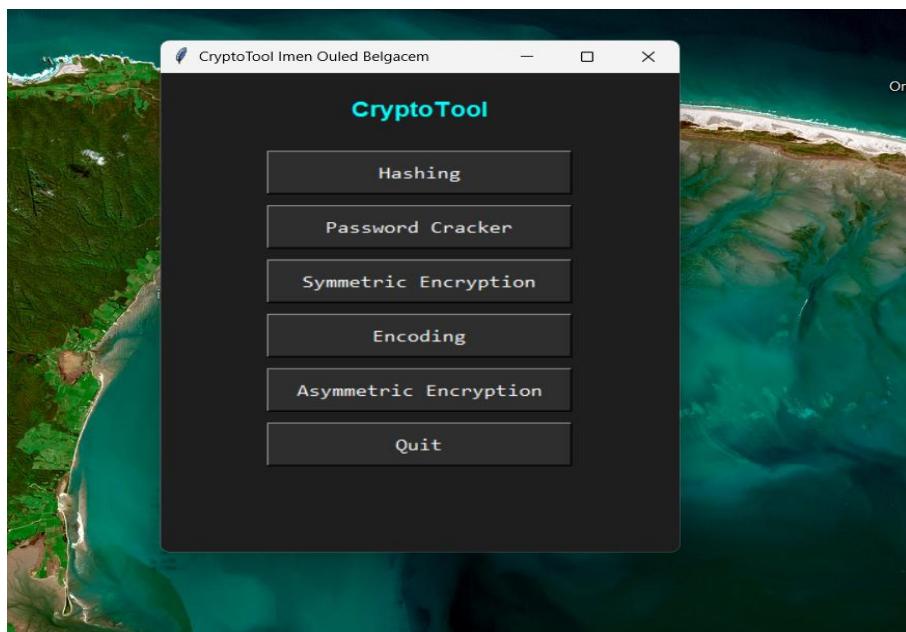
5. Livrables

Les livrables de ce projet incluent :

1. Code Source : Tous les fichiers Python et les scripts nécessaires pour faire fonctionner l'application.
2. Documentation Technique : Un guide expliquant l'architecture de l'application, les fonctions principales, et les étapes d'installation et de lancement.
3. Exemples de fichiers : Des fichiers comme `wordlist.txt` pour le cracking des mots de passe, ainsi que des exemples d'entrées et de sorties.
4. Rapport de Projet : Le rapport détaillant l'architecture, les fonctionnalités, et les méthodologies du projet.

6. Interface avec test

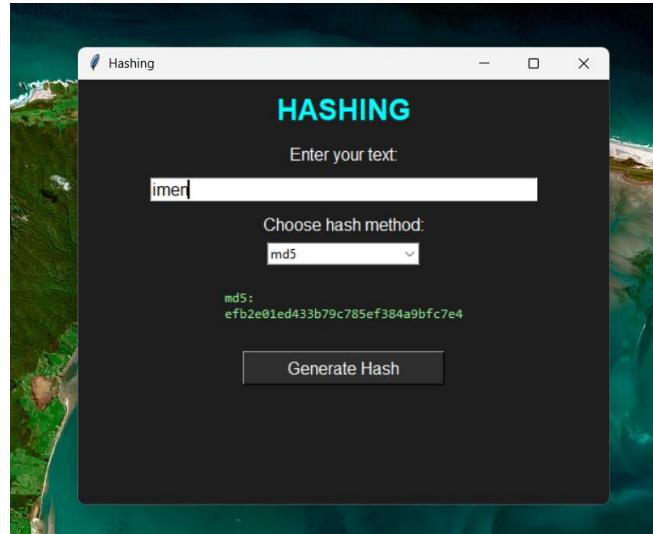
Interface principale:



Fonctionnalités Principales :

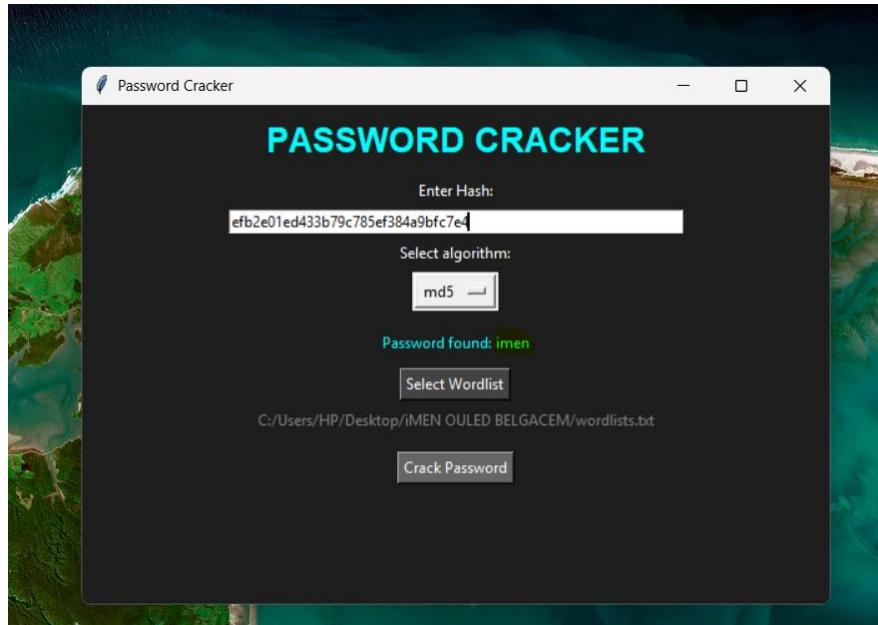
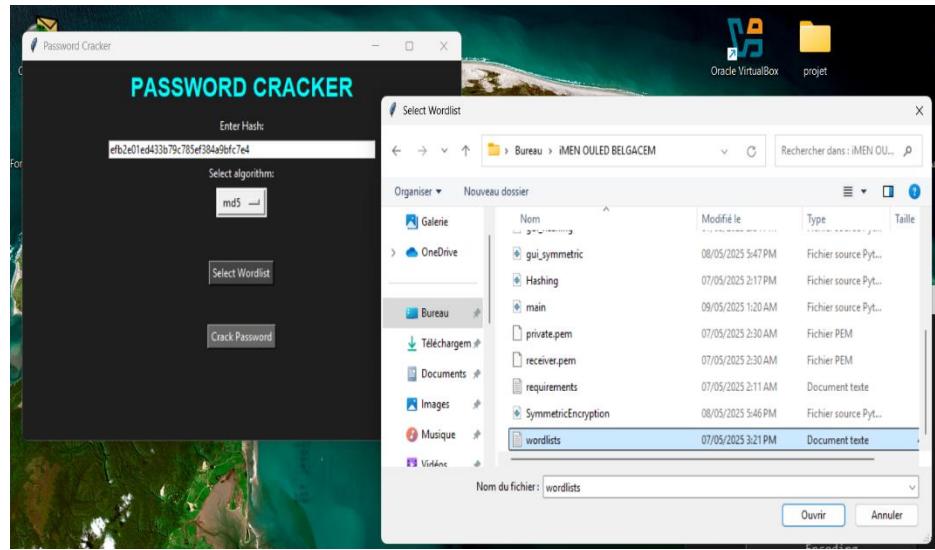
1. Outils de Hachage :

- Support de **MD5, SHA-1, SHA-256, SHA-512**.
- Conversion rapide de texte en empreinte cryptographique.



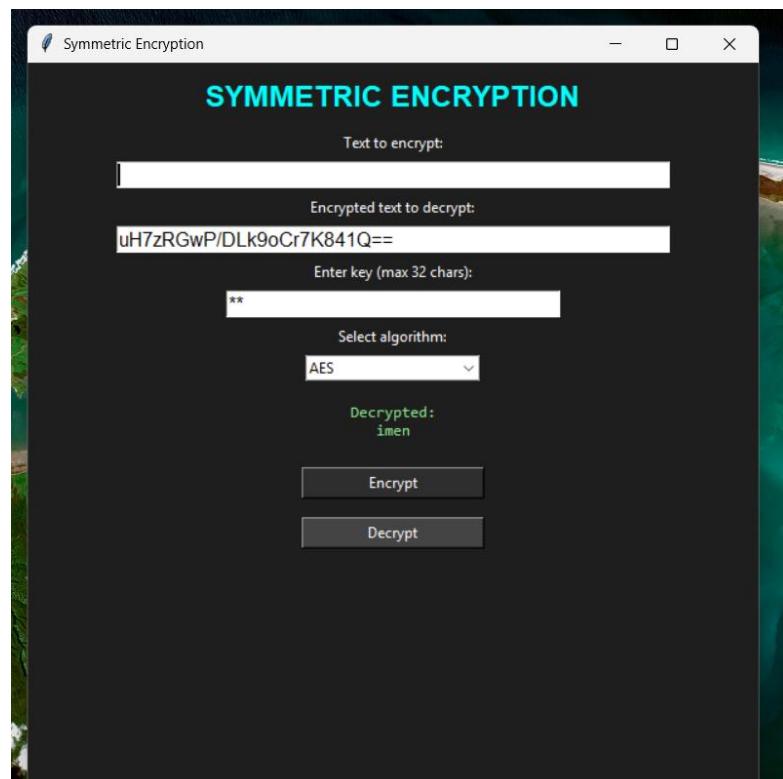
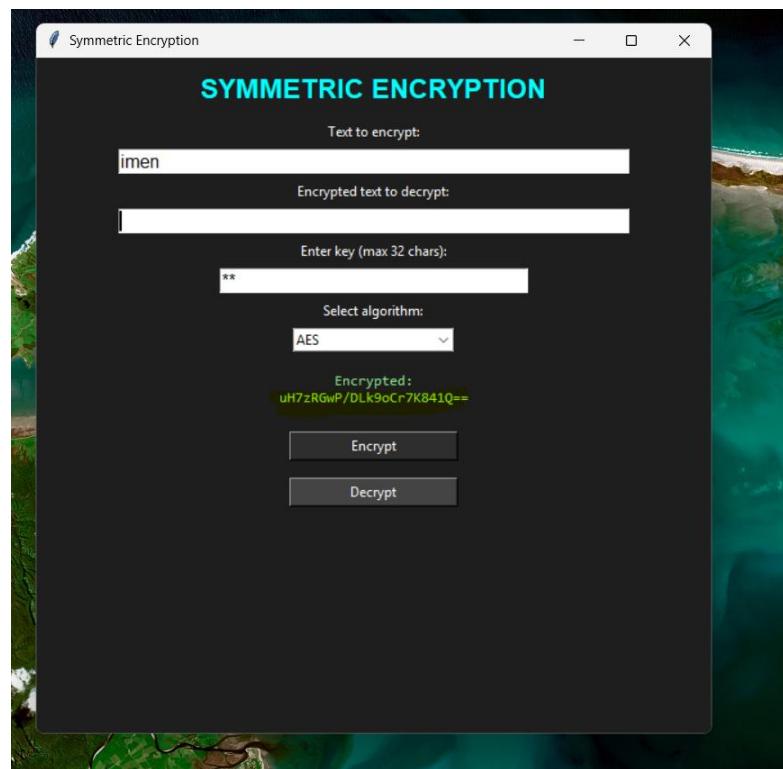
2. Password Cracker :

- Attaque par dictionnaire sur les hachages **MD5, SHA-1, SHA-256**.
- Support de wordlists personnalisées.



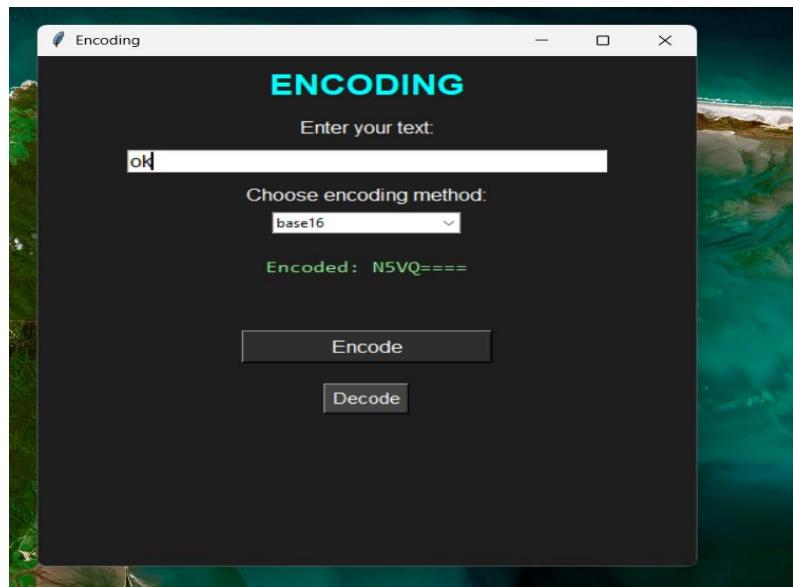
3. Chiffrement Symétrique :

- **AES-128** (mode ECB) et **Salsa20** .
- Gestion automatique du padding et affichage clair du texte chiffré en Base64.



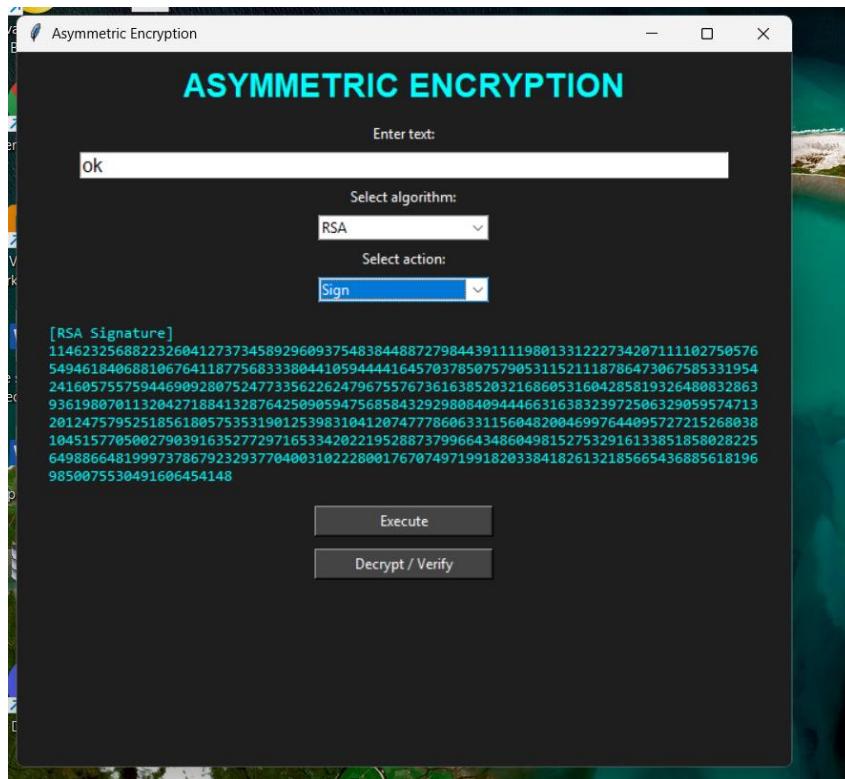
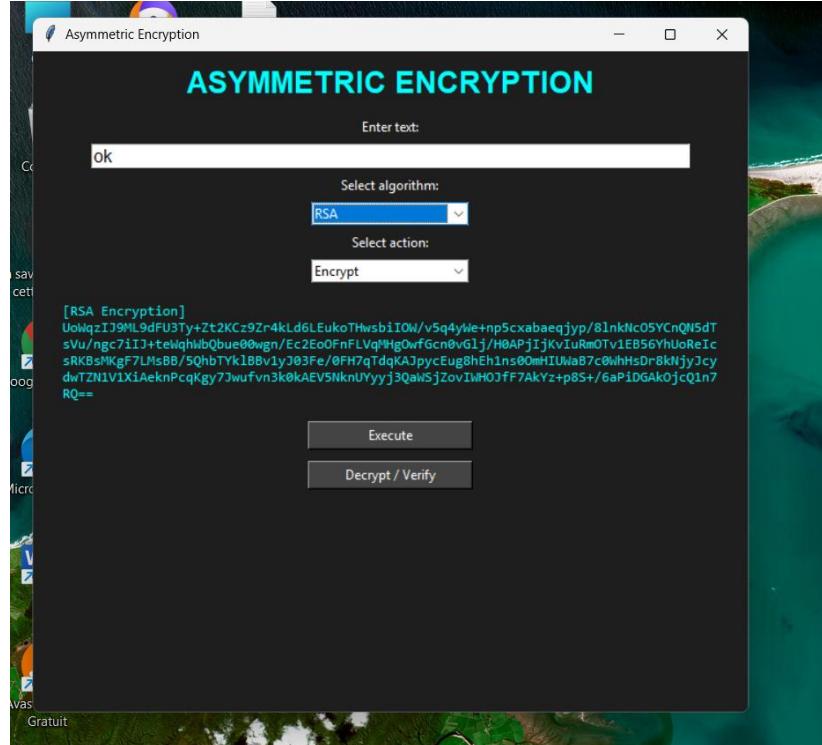
4. Encodage:

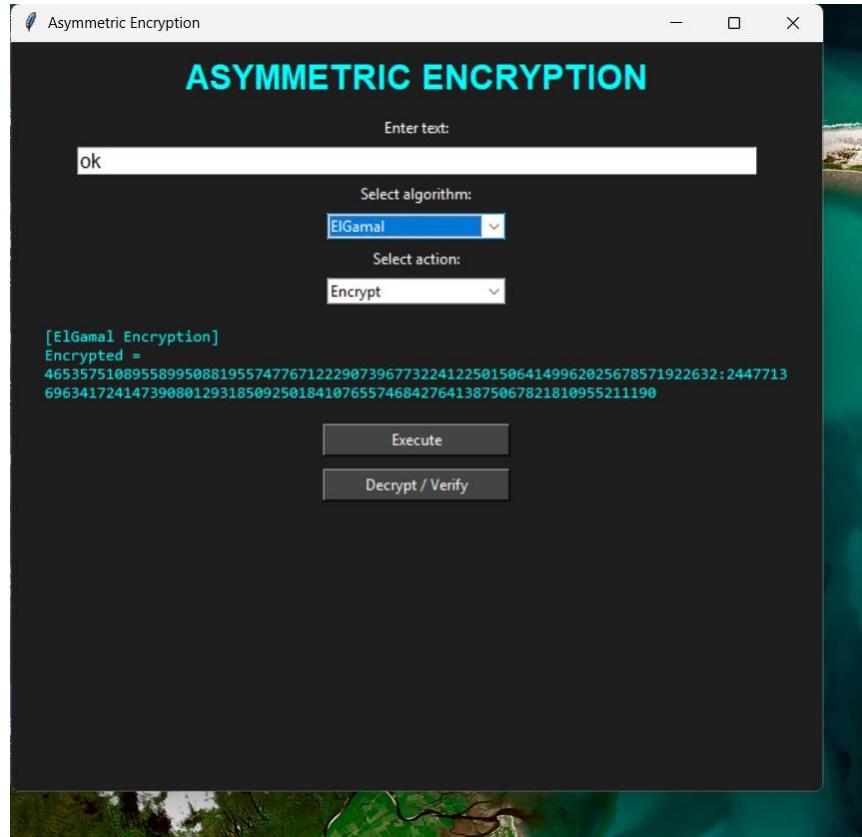
- **Base16, Base32, Base64** pour la conversion de données.
- **UTF-8, ASCII .**



5. Chiffrement Asymétrique :

- **RSA** (2048 bits) avec chiffrement OAEP et signatures numériques.
- **ElGamal** pour un chiffrement sécurisé basé sur les problèmes logarithmiques discrets.





7. Conclusion

Ce projet présente une application puissante et modulaire pour gérer différentes tâches cryptographiques et de sécurité. Grâce à son interface utilisateur conviviale et à l'utilisation de bibliothèques éprouvées, l'application permet de réaliser des opérations complexes comme le hachage, l'encodage/décodage, et le chiffrement/déchiffrement de manière simple et intuitive. La modularité du projet permet d'ajouter facilement de nouvelles fonctionnalités et de maintenir une architecture évolutive.

Le projet a permis de mettre en pratique des concepts avancés de cryptographie tout en offrant une solution accessible pour l'utilisateur final.

