École d'ingénieurs du numérique

Janvier 2024
Cycle ingénieur
Auteurs : G10D
Module IF.1204 : Sciences du numérique

Manuel d'utilisation

Table des matières

1 Prérequis d'utilisation

Tout d'abord, pour lancer le code, il est nécessaire de compiler le fichier "Main.py", lors de la première compilation, il est normal d'avoir cette erreur :

FIGURE 1 – Erreur lors du lancement

Pour résoudre cette erreur, il faut aller dans terminal, puis entrer la commande "pip install -r requirements"

FIGURE 2 – Commande dans le terminal pour installer les dépendances

Désormais, il ne reste plus qu'à compiler le fichier "Main.py", l'interface suivante devrait apparaitre :

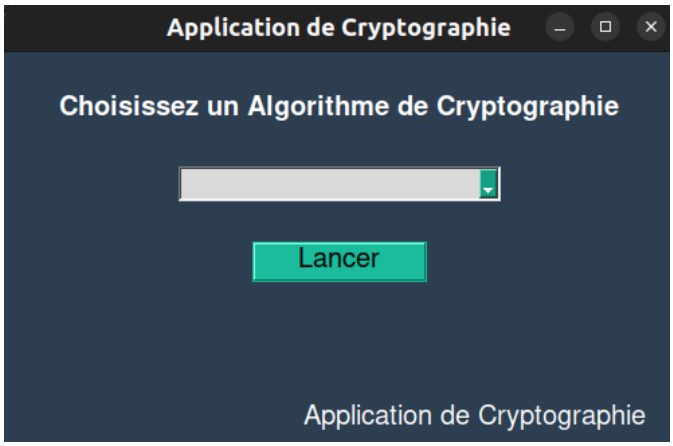

FIGURE 3 – Interface

2 Description de la simulation

Une fois que l'on à accès à l'interface, il est possible pour l'utilisateur de simuler différents algorithmes de cryptographie.

2.1 Algorithme RSA

Le RSA fonctionne en créant deux clés distinctes mais mathématiquement liées : une clé publique et une clé privée. La clé publique, comme son nom l'indique, peut être partagée ouvertement, tandis que la clé privée doit être gardée secrète. La clé publique est utilisée pour chiffrer les données, et seule la clé privée correspondante peut les déchiffrer, assurant ainsi la confidentialité et l'intégrité des données transmises.

Dans notre simulation, l'algorithme RSA est implémenté de manière à permettre à l'utilisateur de générer des clés, de chiffrer un message avec la clé publique, puis de le déchiffrer avec la clé privée correspondante. L'interface graphique fournit des champs pour entrer le message, afficher le message chiffré et enfin dévoiler le message déchiffré.

2.2 Cryptographie sur les courbes elliptiques (ECC)

ECC repose sur la difficulté de résoudre le problème du logarithme discret dans le contexte des courbes elliptiques. Contrairement à RSA qui utilise la factorisation des grands nombres, ECC travaille avec des points sur une courbe elliptique et des opérations mathématiques définies sur ces points.

Notre simulation utilise ECC pour générer des clés et chiffrer des messages. La clé privée est utilisée pour signer le message (authenticité), et la clé publique correspondante sert à vérifier cette signature. L'application permet également d'effectuer un chiffrement symétrique des messages en utilisant AES, ce qui démontre une utilisation pratique de l'ECC dans la sécurisation des communications.

2.3 Algorithme Fernet

Fernet représente une application particulière du chiffrement symétrique, utilisant AES en mode CBC (Cipher Block Chaining) avec un HMAC pour l'authentification des messages. Dans un système de chiffrement symétrique comme Fernet, la même clé est utilisée pour chiffrer et déchiffrer les données, ce qui le différencie clairement des méthodes asymétriques comme l'ECC ou RSA.

Dans notre simulation, l'algorithme Fernet permet aux utilisateurs de générer une clé de chiffrement, de chiffrer un message avec cette clé, puis de le déchiffrer. Cette démonstration offre un aperçu du fonctionnement d'un système de chiffrement symétrique et de son utilisation dans la sécurité des données.

2.4 Distribution quantique de clé (QKD)

La QKD utilise des propriétés quantiques, telles que l'intrication et la superposition, pour garantir la sécurité. Si un tiers tente d'espionner la clé, l'état quantique des particules est modifié, alertant les parties de la présence d'un espion.

Notre simulation simule le processus de QKD en générant des séquences de bits et en les comparant pour établir une clé partagée. Bien que la simulation ne puisse pas reproduire les propriétés quantiques réelles, elle offre un modèle simplifié pour comprendre comment les clés sont partagées et sécurisées dans la QKD. L'application génère des bits et des bases aléatoires, simulant l'envoi de qubits, et compare ensuite les bases pour déterminer les bits à utiliser dans la clé partagée. Cette simulation illustre l'idée fondamentale derrière la QKD sans nécessiter une compréhension très approfondie de la mécanique quantique, ce qui la rend accessible. Le résultat est donc par la suite affiché en console.

2.5 Chiffrement de Vigenère

Le chiffrement de Vigenère s'effectue en alignant des lettres d'un mot clé avec le texte clair et en appliquant un décalage de César différent pour chaque lettre du texte. La sécurité du système repose sur la longueur et la complexité du mot clé. Plus le mot clé est long et moins il a de répétitions, plus le chiffrement est difficile à casser par analyse de fréquence.

Dans notre simulation, l'utilisateur peut saisir un message et un mot clé pour le chiffrement de Vigenère. L'interface ne fournit pas de champs pour entrer ces données et afficher ensuite le message chiffré. Seule une démonstration est faite dans la console avec la phrase "Bonjour, ceci est un message secret !", qui est chiffrée puis déchiffrée.

3 Références

1. REF1
2. REF2
3. REF3
4. REF4
5. REF5
6. REF6
7. REF7
8. REF8
9. REF9
10. REF10
11. REF11
12. REF12
13. REF13
14. REF14
15. REF15
16. REF16
17. REF17
18. REF18
19. REF19