Etapes de developpement: retrouver le produit n=p*q cle RSA

Étape 1:

→ Connaitre le fonctionnement des clés RSA (vu en cours).

→ Déterminer les principales fonctions (décomposer la clé en deux fateurs premiers).

Étape 2:

entre chaque sous étape beaucoup de tests sont réalisé.

→ Premiers essais avec des clés de 7bit (5x17 = 85) avec une méthode de force brut, essayer de diviser par tous les nombres a partir de 2.

→ Première amélioration ne pas tout tester jusqu'au nombre mais que jusqu'à la moitier.

→ Deuxième amélioration essayer de divisier par deux et si cela ne fonctionne pas, aller de deux en deux pour ne tester que les nombres impaires, cela divise le nombre de tests par deux.

→ Finalement ne pas diviser par deux, mais aller jusqu'a la racine carré du nombre.

→ Troisième améliroation jusqu'a maintenant la méthode utilisé était une méthode 'bottom up' (on part de deux jusqu'à la racine carré), alors que souvent les nombres sont plus proches de la racine carré que de deux.

Étape 3:

→ J'ai opté pour une utilisation dans le terminal, plus classique avec quelques touches de couleurs en utilisant des

codes ANSI.

→ Création de la Fonction main qui gère les principales parties du programme et les premières entrées utilisateur.

→ Création de la Fonction affichage pour éviter les redondances et augmenter la modularitée.

→ Création de la Fonction execute, cette fonction permet en fonction du choix de l'utilisateur d'utiliser la méthode choisie, ceci dans le but de pouvoir facilement ajouter des choix d'algorithme pour casser les clés.

→ Création de la fonction cassage_cle_brut qui prend une liste de nombre premiers dans un fichier et qui vas tester toute les possibilités.

→ ajout de lignes de code pour comparer le temps que mettent les algorithmes.

Documentation:

À l'execution on vous demande la clé que vous souhaitez décomposer, puis la méthode que vous souhaitez utiliser. Le programme fait les calculs et vous renvoie le résultat.