Séance 2 - Travaux Pratiques
Cryptanalyse à chiffrés connus (chiffrements historiques)

Yann Rotella

2026

L'objectif de cette séance est d'appliquer en programmation certaines cryptanalyses vues en cours et dans le premier TD. Pour bien comprendre les cryptosystèmes, il faudra les programmer en `Python`, vérifier que le déchiffrement fonctionne bien. Une bonne manière de vérifier que l'on ne fait pas de bêtise est de comparer les résultats des fonctions avec les autres. La cryptanalyse de Vigenère est donnée en exercice complémentaire.

Échauffement - seul.e - 30 minutes

Exercice 1. *César.*
Pour se simplifier la vie, nous allons considérer que nous allons chiffrer uniquement des textes écrits en français, sans accents.

(1) Coder une fonction qui prend en entrée une chaîne de caractère et supprime toute la ponctuation et les espaces (tout ce qui n'est pas des caractères latins en majuscules ou en minuscule). On aura besoin de `ord` et `chr` comme méthodes et du code ASCII.

(2) Coder une fonction qui transforme cette chaîne de caractères en une chaîne avec que des majuscules.

(3) Coder une fonction qui prend en entrée un entier entre 0 et 25 et qui chiffre la chaîne de caractères avec le chiffrement de César.

(4) Coder la fonction de déchiffrement.

(5) Coder la fonction `cryptanalyseFrequence` qui prend en entrée une chaîne de caractères, compte l'apparition maximale et renvoie la clef secrète associée.

Cryptanalyse d'Enigma - en groupe - 2h30

Exercice 2. *Enigma - Simuler la machine.*
On va commencer par avoir notre propre code d'Enigma.

(1) Sous forme de 5 listes, mettez en dur dans 5 listes globales les permutations des rotors définis au TD 1. Attention, peut être que commencer à compter à partir de 0 peut être intéressant.

(2) Coder une fonction qui prend en entrée une valeur entre 0 et 25 représentant la position du rotor ainsi qu'une valeur entre 0 et 4 représentant un rotor, une autre valeur entre 0 et 25 représentant le fil électrique allumé et applique la bonne permutation associée.

(3) Coder une fonction qui prend en entrée un nombre entre 0 et 9 et renvoie le choix des trois rotors (non ordonné)

(4) Coder une fonction qui prend en entrée un nombre entre 0 et 5 et renvoie l'ordre des trois rotors choisis.

(5) Coder une fonction qui prend en entrée un nombre entre 0 et 5, un autre entre 0 et 9 et trois valeurs représentant les valeurs actuelles des rotors et calcule la permutation associée (c'est-à-dire Enigma sans le tableau de connexion).

(6) Terminer Enigma (toujours sans le tableau de connexion) en y incluant la position d'entrainement des rotors comme paramètre afin de pouvoir encoder des textes longs. Vérifiez que vous pouvez chiffrer et déchiffrer votre texte.

Exercice 3. *Enigma - Cryptanalyse.*
Le but de cet exercice est de réaliser d'abord une attaque exhaustive sans le tableau de connexion, puis de réaliser l'attaque de la cryptanalyse de Rejewski.

(1) Sans le tableau de connexion, quelle est l'**entropie** de la clef secrète ?

(2) Pour pouvoir tester plus rapidement les clefs, nous allons supposer que l'attaquant connaît les positions d'entrainement des rotors (ceci est uniquement afin de pouvoir faire plusieurs tests rapidement). Réalisez alors l'attaque par recherche exhaustive. De quoi avez-vous besoin ? Quel temps prend votre attaque ? Si on suppose inconnu la position d'entrainement des rotors, quel temps prendrait l'attaque ?

(3) Nous allons maintenant rajouter le critère distinguant de Rejewski et de sa cryptanalyse. Quel temps prendrait l'attaque par recherche exhaustive sur votre ordinateur si l'on rajoute 1, 2, ..., 13 paires sélectionnées dans le tableau de connexion ?

(4) Programmer une fonction qui renvoie une configuration aléatoire du tableau de connexion pour n paires de lettres, puis modifier votre machine Enigma pour avoir l'Enigma modèle M3 complète.

(5) Programmer une fonction qui tire aléatoirement une permutation au hasard. On pourra aller voir le Fisher-Yates shuffle sur Wikipedia.

(6) En utilisant des permutations au hasard prises précédemment, programmer une fonction qui prend en entrée une liste de paires de lettres, dont la première est tirée aléatoirement et la deuxième est construite à partir de la permutation. La fonction doit renvoyer la liste (peut-être incomplète) des tailles de cycles de la permutation. À partir de combien de paires pouvez-vous avoir, avec grande probabilité, la taille des cycles ?

(7) En supposant que le deuxième et le troisième rotor n'arrivent jamais à leur position d'entrainement et avec le choix des rotors connu, donnez combien de caractéristiques possibles vous pouvez avoir ? Est-ce que cela tient en mémoire RAM ?

(8) Programmez l'attaque de Rejewski et retrouvez, avec les hypothèses précédentes, la position des rotors. On veillera à transmettre suffisamment de messages clefs.

(9) Si on suppose que les messages clefs du jour sont choisis aléatoirement, est-ce possible de retrouver avec ces seules informations, la configuration du tableau de connexion ?

Exercice complémentaire

Exercice 4. *Chiffre de Battista (Vigenère).*
On va maintenant programmer l'attaque vue en cours sur ce chiffrement.

(1) Coder une fonction qui prend en entrée une clef sous la forme d'une chaîne de caractères en majuscules, un message en français en majuscules et sans ponctuation ni accent et renvoie le chiffré correspondant avec le chiffre de Vigenère.

(2) Coder la fonction de déchiffrement. Vérifiez que vous obtenez le même texte en sortie.

(3) Coder une fonction qui prend en entrée un texte en majuscules et qui renvoie une liste contenant tous les trigrammes qui se répètent dans le texte.

(4) Coder une fonction qui prend en entrée un trigramme qui se répète dans le texte et qui renvoie la liste des distances dans le texte entre ces trigrammes.

(5) Coder une fonction qui prend en entrée un nombre et renvoie la liste de ses diviseurs.

(6) Combiner toutes ces fonctions afin d'en coder une autre qui prend en entrée un texte en majuscules et renvoie la liste de tous les diviseurs, comptés avec multiplicités, qui peuvent être trouvés dans le texte.

(7) En déduire une fonction qui réalise la cryptanalyse de chiffre de Battista. Tester votre fonction sur des textes plus ou moins longs et des clefs plus ou moins grandes.