# Compte rendu TP 1-2 — Fonctions de hachage

**Auteurs :** Milan LOÏ, Jules GAYET  
**Date :** 03/10/2025  
**Matière :** BUT2 Informatique – Cryptographie

## 1) Organisation du dépôt & fichiers livrés

**-** *MauvaiseFonctionHashage.java* : contient une première fonction de hachage naïve (basée sur les codes ASCII). Sert à illustrer les collisions fréquentes et la mauvaise répartition.

**-** *PreimagesCollisions.java* : implémente la détection de collisions et de préimages pour une fonction de hachage donnée.

**-** *HStar.java* : propose une fonction de hachage améliorée (base 37, modulo grand nombre premier) et des tests sur des dictionnaires de mots.

**-** *attack.java* : script effectuant une attaque par collision, cherchant la première collision dans un ensemble de mots (avec comptage du nombre d’essais).

**-** *ods5.txt* : fichier dictionnaire de mots utilisés comme base de test.

## Conventions d’implémentation

• Alphabet traité : lettres a-z/A-Z uniquement (les autres caractères sont pris tels quels).

• Structure du code : chaque fichier .java contient une classe principale avec méthode main exécutant les tests directement.

• Gestion des collisions : les mots sont stockés dans une Map<Integer, List<String>> où la clé correspond au hash calculé.

• Langage utilisé : Java 21, exécution avec javac \*.java puis java NomClasse.

## 2) Principe des algorithmes implémentés

### 2.1 Mauvaise fonction de hachage

• Calcule le hash comme somme des codes ASCII des caractères.

### Une image contenant texte, Police, capture d’écran, nombre Le contenu généré par l’IA peut être incorrect.

### 2.2 Recherche de préimages et collisions

• PreimagesCollisions.java construit une table de hachage à partir d’un fichier ou d’une liste de chaînes.

Une image contenant texte, capture d’écran, Police

Le contenu généré par l’IA peut être incorrect.

### 2.3 Fonction de hachage améliorée (HStar)

• Nouvelle approche : combinaison de positions pondérées.

• But : meilleure dispersion des valeurs, donc moins de collisions.

Une image contenant texte, capture d’écran, Police

Le contenu généré par l’IA peut être incorrect.

### 2.4 Script d’attaque (attack.java)

• Effectue des essais successifs de hachage jusqu’à trouver une collision.

• Affiche :

- le nombre total d’essais;

- les deux chaînes en collision;

- la valeur du hash commun.

• Sert à estimer la résistance de la fonction au birthday attack.

Une image contenant texte, capture d’écran, noir et blanc, noir

Le contenu généré par l’IA peut être incorrect.

## 4) Conclusion

Le dépôt contient une suite complète d’expérimentations sur les fonctions de hachage : du prototype naïf jusqu’à une version plus robuste. Les tests montrent que :

• la fonction simple provoque un grand nombre de collisions;

• la version améliorée HStar distribue mieux les valeurs;

• le script d’attaque permet de quantifier la résistance à la collision.

Pour aller plus loin :

• Implémenter des fonctions cryptographiques standard (SHA-256, MD5 pour comparaison).

• Automatiser les tests sur de grands dictionnaires.

• Ajouter une analyse statistique (histogramme de distribution, taux de charge des buckets).

• Rendre le module compatible avec des entrées binaires (non-textuelles).