CS-107 2025

Introduction à la Programmation : Mini-projet 1 en auto-évaluation

Introduction au projet

Rafael Pires
rafael.pires@epfl.ch

EPFL, Lausanne, 15.10.2025

Stéganographie

Challenge : Jouez les détectives

Capture de drapeau : `FLAG{…}`

Prix aux 10 premiers groupes à atteindre 100 points

But : atteindre au moins 60 points avant le 03.11.2025 (après cette date, le système d'évaluation sera fermé)

Ce projet n'est pas noté.

On peut néanmoins mettre des questions liées à ce projet dans l'examen.

Objectifs du projet

Coder un ensemble de méthodes permettant de :

- Cacher des messages textuels cryptés, ou des images, dans des images;
- Et de les dévoiler.

Objectifs pédagogiques :

- Auto-évaluer votre niveau sur un projet de plus grande envergure.
- Acquérir des premiers reflexes sur les méthodologies de tests d'un programme.
- Vous familiariser avec l'utilisation d'un debugger.
- Expérimenter le travail collaboratif et (optionnellement) l'utilisation d'un outil de gestion de version.
- Entraînement pour le 2e mini-projet, qui sera noté et plus conséquent.

Conseils pour le travail collaboratif

- **Git : Outil de gestion de version.**

- Il permet de gérer un **dépot** (repository, en anglais) en fournissant les fonctionnalités suivantes :

    - Plusieurs personnes peuvent modifier et gérer en même temps les mêmes documents.
    - Enregistrement de l'historique de modification des documents.
    - Possibilité de revenir en arrière à une version précédente quelconque.

Ressources.

Objectifs du cours d’aujourd’hui

- Présentation générale du projet
- La représentation des données dans le projet
- La cryptographie et la stéganographie
- Mise en place du projet
- Les pièges communs et conseils

Représentation des données : Texte

UTF-8 : Unicode Transformation Format – 8-bit.

Norme d'encodage de caractères la plus utilisée aujourd'hui.

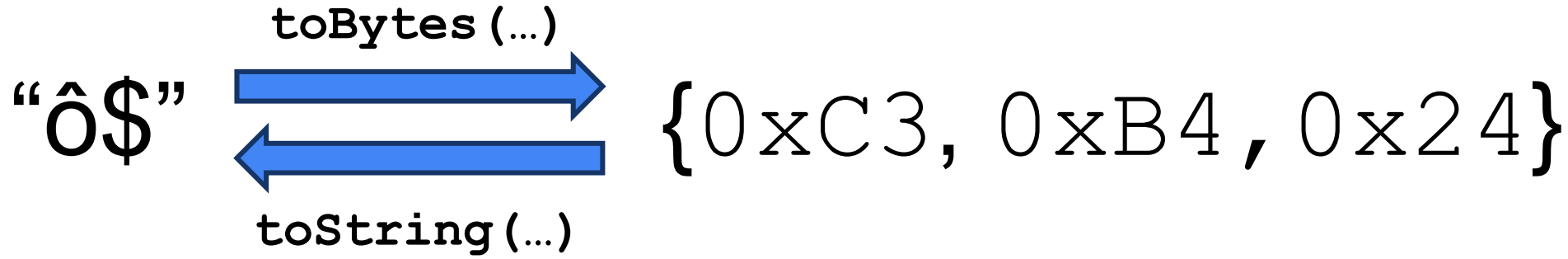
- Capable d'encoder plus d'un million de points de code Unicode valides.
- Encodage de largeur variable, allant de un à quatre octets.
- Les points de code les plus fréquents sont encodés utilisant moins d'octets.
- Rétrocompatible avec l'ASCII : les 128 premiers caractères (1 octet).

"ô" → {0xC3, 0xB4}

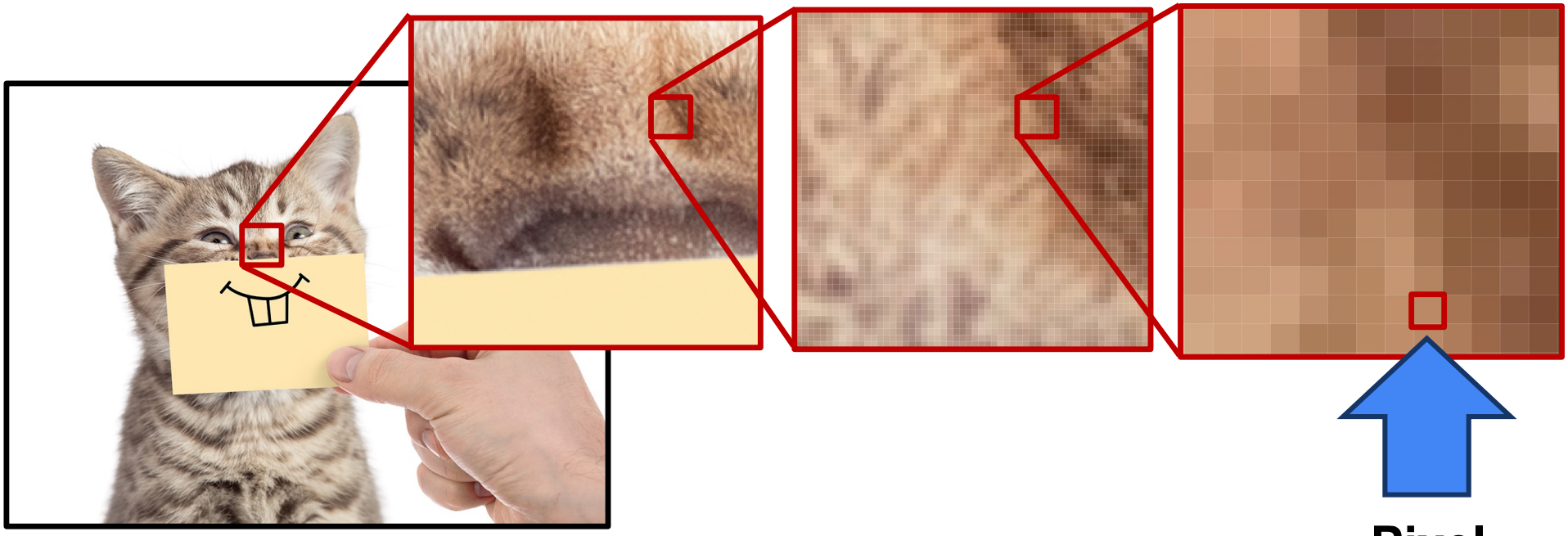
"$" → {0x24}

Les chaînes de caractères en Java

- En Java, les chaînes de caractères sont représentées par le type **`String`**, mais leur encodage peut varier d'une machine à l'autre.
- Dans ce projet, nous manipulerons les chaînes de caractères sous un format plus fondamental, les **tableaux de `byte`**, avec un encodage fixé à **UTF-8**.
- Pas de panique : on vous fournit les méthodes suivantes dans **`Text.java`**.



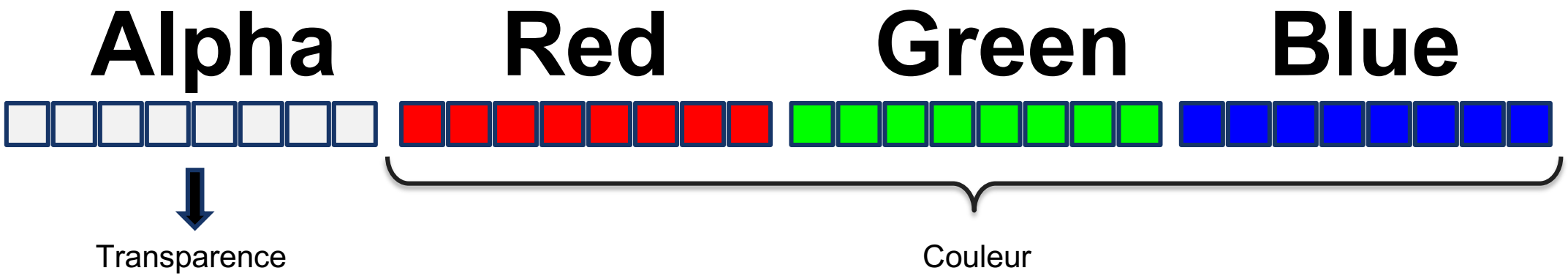
Représentation des données : Images

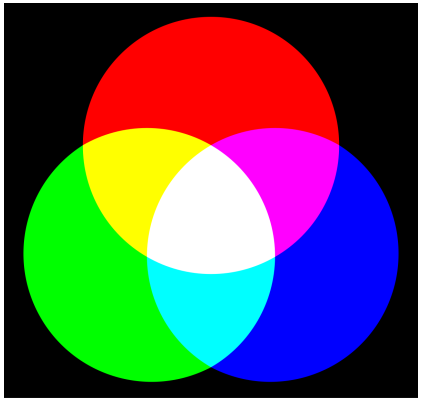


Pixel

Point de l'image qui a une seule couleur

ARGB





Noir = (0, 0, 0)
Blanc = (255, 255, 255)
Bleu = (0, 0, 255)

1 pixel = 32 bits = 4 x 8 bits = 4 nombres entre 0-255

Objectifs du cours d’aujourd’hui

- Présentation générale du projet
- La représentation des données dans le projet
- La cryptographie et la stéganographie
- Mise en place du projet
- Les pièges communs et conseils

Cryptographie

La science de sécuriser des informations en les transformant de manière à les rendre illisibles pour toute personne non autorisée, grâce à des techniques de **chiffrement** et de **déchiffrement**.

Vocabulaire :

- **Message en clair (*plaintext*)** : message à chiffrer.
- **Message chiffré/crypté (*cipher text*)** : version chiffrée du message.
- **Crypto-système (*cipher*)** : algorithme permettant de chiffrer/déchiffrer un message.
- **Clé (*key*)** : information secrète utilisée pour chiffrer/déchiffrer un message.

Cryptographie

Crypto-systèmes à implémenter :

- César
- Vigenère
- XOR
- OTP (One-time pad)
- CBC (Cipher block chain)

Les descriptifs détaillés sont dans l'énoncé, avec des exemples d'exécution attendue dans le fichier fourni **`Main.java`**.

Chiffrement de César

- Chiffrement par décalage de lettres dans l'alphabet
- Exemple avec un décalage de 3 :
  - Message : BONJOUR
  - Clé : +3
  - Chiffré : ERQMRXU
- Facile à casser : il n'y a que 25 possibilités
- Sert surtout à illustrer le principe de substitution

Chiffrement de César

Important : le but est de chiffrer des textes exprimés comme tableaux de byte.

L'alphabet utilisé est donc l'ensemble des bytes possibles : toutes les valeurs entre -128 et 127.

Chaque byte de la chaîne à chiffrer est décalé de la valeur d'une clé donnée (aussi un byte), modulo la taille de l'alphabet.

- si le byte vaut 97 et la clé 3, le byte encrypté vaudra 100
- décrypter revient à crypter avec la clé inverse, -3

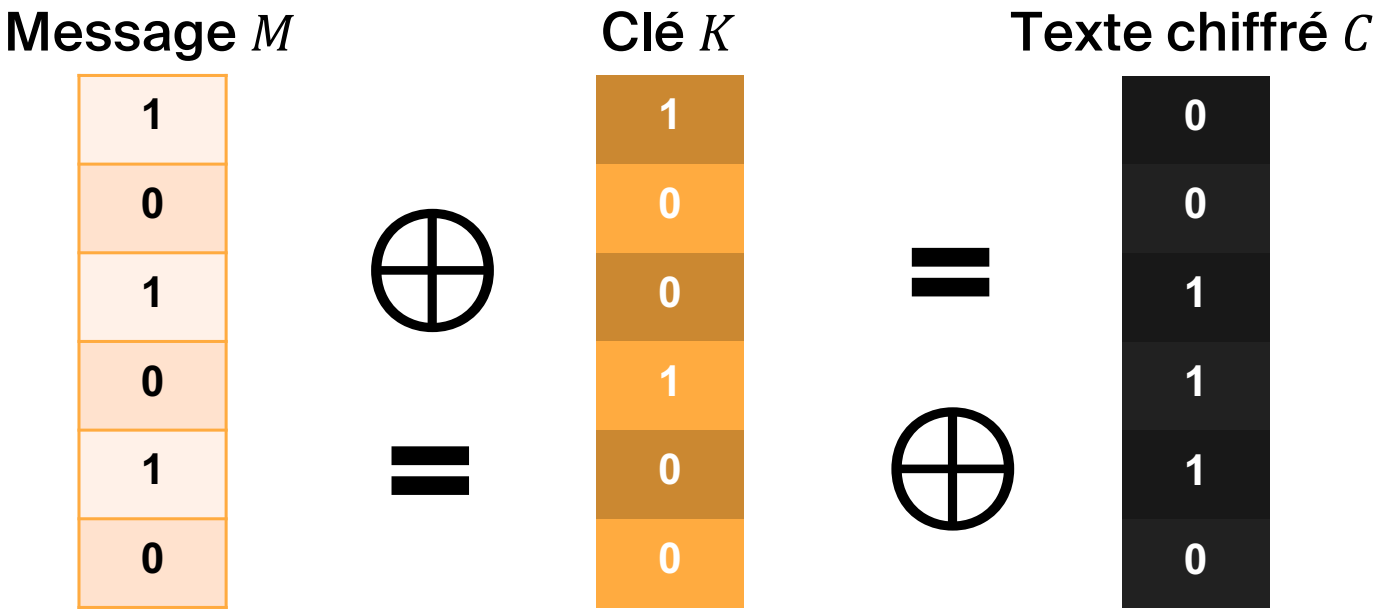
Vigenère

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Message M		+	Clé K		=	Texte chiffré C	
C	2		S	18		U	20
O	14		E	4		S	18
U	20		C	2		W	22
C	2		R	17		T	19
O	14		E	4		S	18
U	20		T	19		N	13

$$C = (M + K) \bmod n \longrightarrow D = (C - K) \bmod n$$

Clé à usage unique *(« one-time pad »)*



$C = M \oplus K$ ⟶ $D = C \oplus K$

$D = M \oplus K \oplus K$

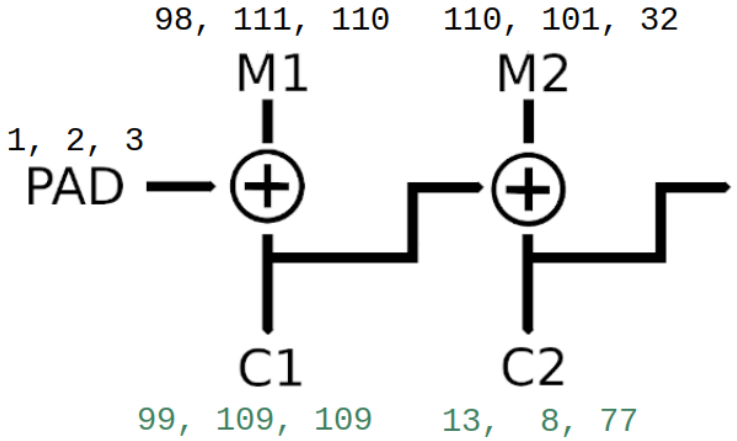
$D = M$

Pas de débordement 👍

Crypto-système CBC (simplifiée)

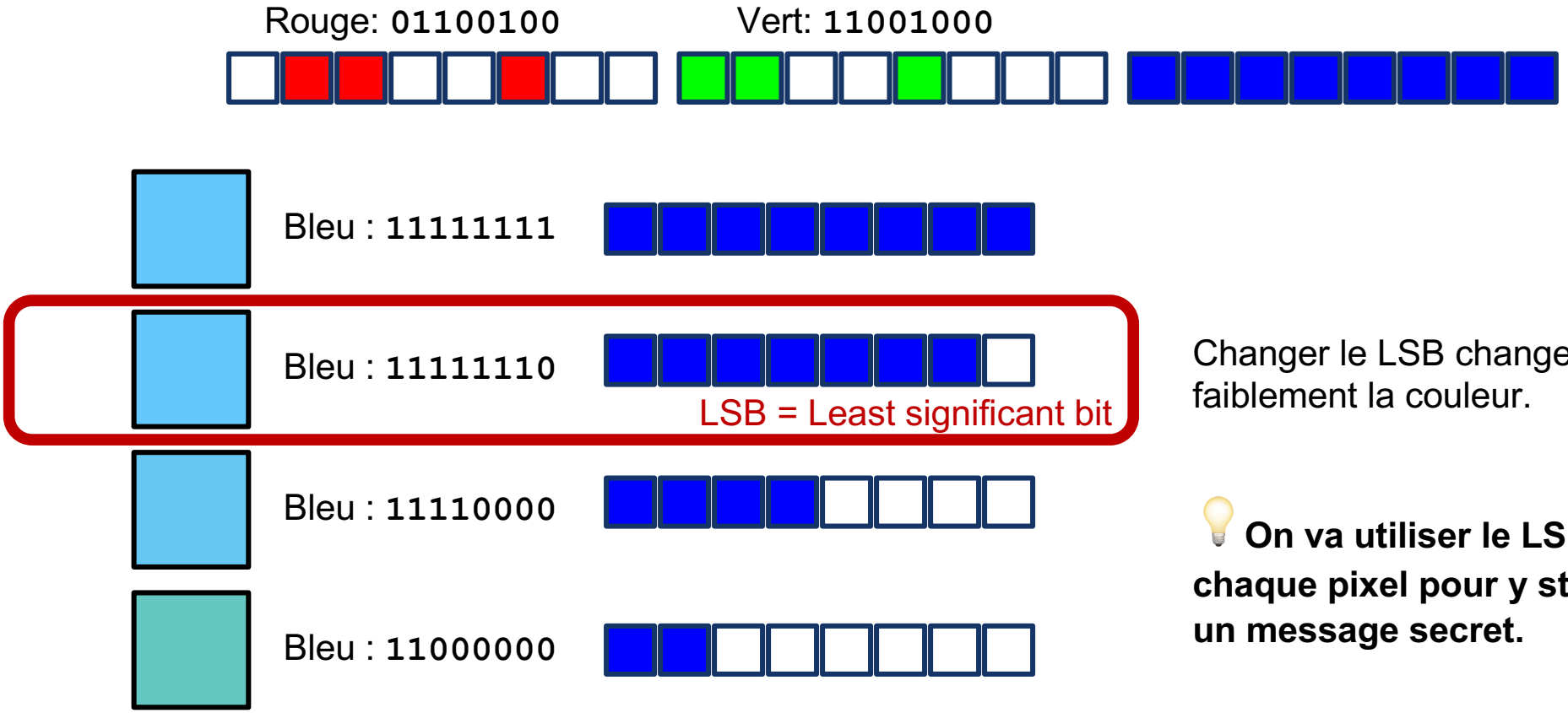
- Décompose le message en blocs de la taille du pad.
- Effectue un XOR entre les blocs et le pad.

```
byte[] plain  = {98, 111, 110, 110, 101, 32};
byte[] pad    = {1, 2, 3};
byte[] cipher = Encrypt.cbc(plain, pad); // {99, 109, 109, 13, 8, 77}
```



XOR		
0	0	0
0	1	1
1	0	1
1	1	0

Stéganographie



Changer le LSB change très faiblement la couleur.

💡 **On va utiliser le LSB de chaque pixel pour y stocker un message secret.**

Stéganographie : images

- **But** : cacher une image en noir et blanc dans les LSB des pixels.

- La taille de l'image cachée (charge) doit être inférieure ou égale en hauteur et en largeur à celle de l'image de couverture.
- Les positions des pixels de la charge sont maintenues.

Stéganographie : messages textuels

- **But** : cacher un texte représenté par une séquence de bytes linéairement dans les LSB des pixels.


```
"Hello" → {72, 101, 108, 108, 111} →
{01001000, 01100101, 01101100, 01101100, 01101111}
```

Objectifs du cours d’aujourd’hui

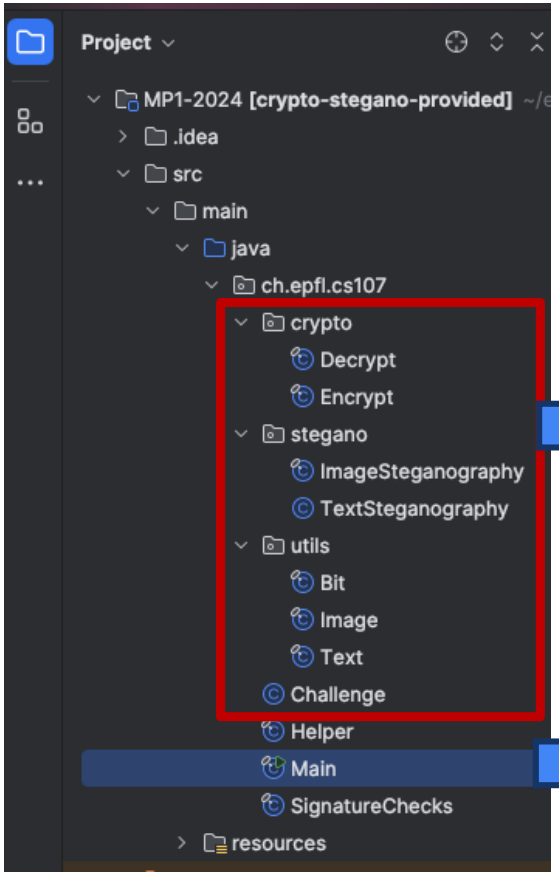
- Présentation générale du projet
- La représentation des données dans le projet
- La cryptographie et la stéganographie
- Mise en place du projet
- Les pièges communs et conseils

Enoncé et mise en place

- **Enoncé :** Liens disponibles sur cette page.

- **Mise en place** : Petit tutoriel sur cette page.

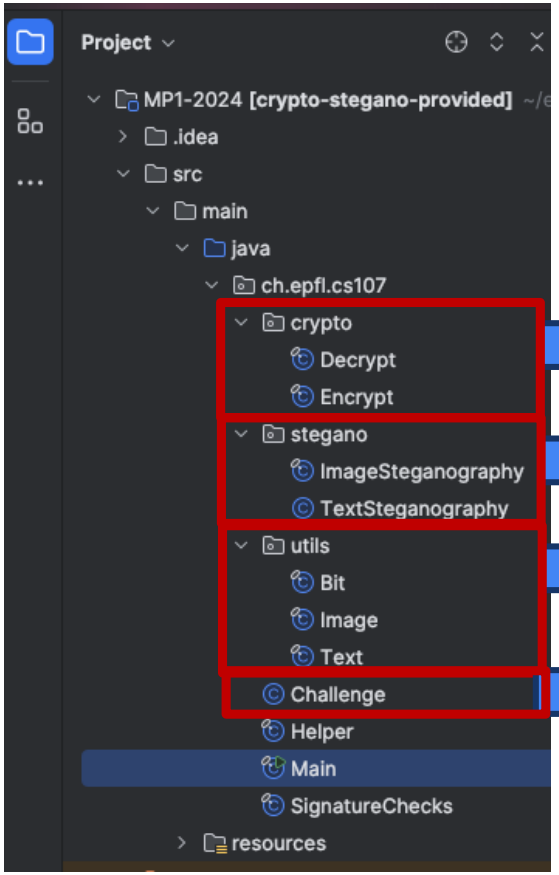
Matériel fourni



À compléter : tous les fichiers des dossiers **`utils`**, **`crypto`** et **`stegano`** et **`Challenge.java`**. Entêtes fournis.

À lancer : **`Main.java`**. À compléter pour bien tester !

Tâches



- Les tâches 2 et 3 peuvent être codées **indépendamment**.

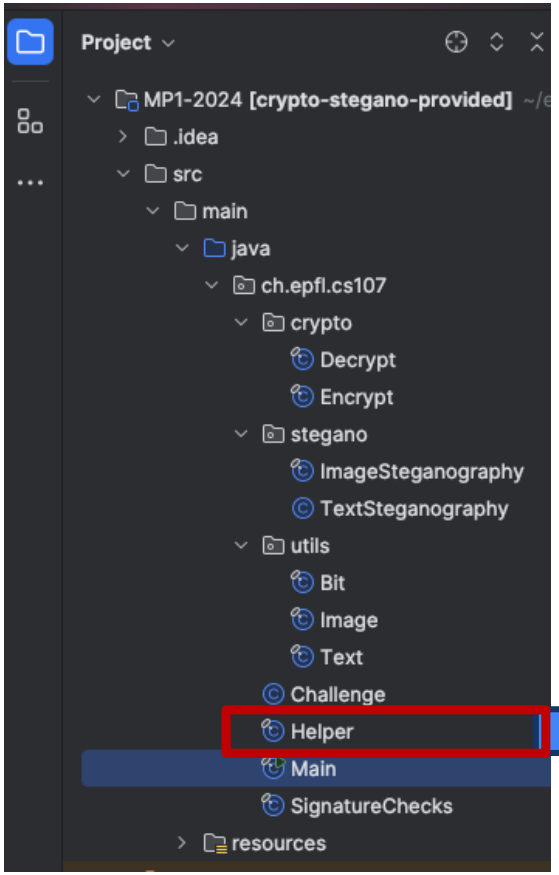
Tâche 2 : Méthodes de cryptographie sous `crypto`.

Tâche 3 : Méthodes de steganographie sous `stegano`.

Tâche 1 : Méthodes utilitaires sous `utils`.

Tâche 4 : Challenge sous `Challenge.java`.

Appels aux méthodes dans un autre fichier



- Syntaxe : **`NomDeClasse.nomDeMethode(…)`**
- Exemple : `Helper.read("image.png")`

N'oubliez pas de regarder **`Helper.java`**.

Correcteur automatique

- Bien tester localement avant de soumettre.
- Ne soumettre que du code qui compile.
- Faire attention aux cas limites (voir le mécanisme des assertions dans l'énoncé).
- Ne jamais utiliser d'appels système comme `System.exit()`.
- Préparez le fichier "`submission.zip`" d'après les consignes.

Remarques importantes :

- le retour du grader peut être lent (dans certains cas limites atteindre les 10 minutes)
- les tests ont une composante aléatoire qui fait que deux soumissions du même code peuvent avoir des notes légèrement différentes.

Merci de nous alerter sur Ed en cas de souci avec l'utilisation de cet outil !

Conseils pour aborder le mini-projet 1

- **N'oubliez pas de vous inscrire dans un groupe.**
- Dès que le grader sera ouvert le vendredi 17 matin, **vous ne pourrez plus vous inscrire dans un groupe**.

- Commencez par lire l'introduction, les consignes et les compléments théoriques.
- Lisez le descriptif des tâches avec soin avant d'entamer le codage.
- Testez systématiquement en local, avant de soumettre au correcteur automatique.
- N'hésitez pas à enrichir le fichier `Main.java` fourni pour pousser les tests plus loin.
- Ne tentez pas de résoudre tous les problèmes en même temps.
- Ne négligez pas les apprentissages et séries à venir au profit de ce projet.

Essayez ces exercices avant le TP de vendredi.

Questions ?

Nous sommes à disposition sur le forum **Ed** pendant la semaine.
N’hésitez pas à poser des questions.

Bon projet à toutes et tous !

rafael.pires@epfl.ch

Merci