

HAI927I  
Projet Image

Compte Rendu n°2  
Sujet n°3 - Musée Sécurisé Virtuel

COUNILLE Alexandra  
&  
LIN-WEE-KUAN Malika

# PARTIE I : PRÉVISIONS, THÉORIES ET ANALYSES

## Travail prévisionnel : répartition des tâches et gestion du temps

### Tâches à réaliser

- Implémentation de l'algorithme de chiffrement et de déchiffrement
- Implémentation de l'algorithme de détection de contours
- Implémentation de l'algorithme de transformation
- Création de photos échantillons pour réaliser une récupération de l'image chiffrée
- Transition vers un système en réalité augmentée, création d'une application mobile

### Gestion du temps

- Création de l'algorithme de chiffrement/déchiffrement et de détection des contours :  
*1 semaine (8/11 - 14/11)*
- Création de l'algorithme de transformation :  
*1 semaine (15/11 - 21/11)*
- Implémentation des algorithmes et mise en pratique sur une photo :  
*1 semaine (22/11 - 28/11)*
- Transition vers un système en réalité augmentée avec un premier prototype :  
*1 semaine (29/11 - 5/12)*
- Création d'une application fonctionnelle et finalisation du projet :  
*1 semaine (6/12 - 12/12)*

### Répartition des tâches

#### Du 8/11 au 14/11

- Création de l'algorithme de chiffrement/déchiffrement : *Malika*
- Création de l'algorithme de détection de contours : *Alexandra*

#### Du 15/11 au 21/11

- Création de l'algorithme de transformation : *Malika*
- Récupération de l'image chiffrée sur une photo et traduction des blocs : *Alexandra*

#### Du 22/11 au 28/11

- Implémentation des algorithmes : *Malika*
- Réalisation des tests et analyse des résultats sur une photo : *Alexandra*

#### Du 19/11 au 5/12

- Transition vers un système en réalité augmentée (prototype) : *Alexandra*
- Réalisation des tests et analyse des résultats en AR : *Malika*

#### Du 6/12 au 12/12

- Création et finalisation de l'application : *Malika, Alexandra*

### Outils et langages utilisés

C++, C# (*développement des algorithmes*)

Unity (*développement de l'application en AR*)

## Analyse des méthodes existantes de chiffrement par permutation

### 1) Méthode 1 - Grille tournante :

Pour faire l'algorithme de cette méthode, on a besoin de générer une grille et son cache puis de faire un parcours pour remplir la grille.

Étant donné qu'il faut 4 passes pour fabriquer le message chiffré ou pour le déchiffrer, la complexité de cette méthode est en  $O(n^4)$ .

Afin de déchiffrer le message, il est nécessaire de connaître une partie du message en clair pour pouvoir retrouver la bonne combinaison des "trous" dans la grille. Cette méthode est donc vulnérable à l'attaque par mot probable.

Exemple de cache avec une grille de 6x6 :

1	2	3	4	5	6
4	5	6	7	8	9
7	8	9	10	11	12
10	11	12	13	14	15
13	14	15	16	17	18
16	17	18	19	20	21

### 2) Méthode 2 - Échange d'indice des symboles :

La complexité de cette méthode est en  $O(n)$  : il suffit d'un seul parcours du message en clair pour construire le message chiffré à partir de la clé. La faiblesse de cette méthode est qu'elle ne résiste pas à l'attaque à texte clair connu : si quelqu'un possède un couple clair-chiffré de la taille de la clé, il pourra alors deviner cette dernière et l'utiliser sur l'ensemble du message, puisque celui-ci est chiffré par bloc de la taille de cette clé.

### 3) Méthode 3 - Permutation de colonne (transposition rectangulaire) :

Le cryptage et le décryptage s'effectuent avec une complexité en  $O(n^2)$  (avec  $n$  la longueur du message).

- Une première boucle est nécessaire afin de créer la grille qui contiendra le message à chiffrer ou à déchiffrer ;
- Une deuxième boucle permet le chiffrement ou le déchiffrement du message dans la grille.

#### 4) Méthode 4 - Chiffrement en dent de scie :

Le principe de ce chiffrement est de réécrire le message dans une grille de hauteur définie par la clé. Chaque symbole est réécrit dans la grille en diagonale. La dernière étape est de réécrire le message ligne par ligne de gauche à droite. La longueur du message doit être un multiple de  $2(N-1)$  avec  $N$  le nombre de lignes de la grille. Il est possible de compléter les cases manquantes avec des symboles.

Ex :  $M = \text{JESUISUNMESSAGEX}$

Le message comporte 16 symboles, la clé peut être alors égale à 3 car la longueur du message est un multiple de 4 ( $2 * (3-1) = 4$ )

J				I				M				A			
	E		U		S		N		E		S		G		X
		S				U				S				E	

En ré-écrivant les symboles ligne par ligne, le message crypté donne alors :

JIMAEUSNESGXSE

Cependant, il est important de noter que la clé ne peut pas être supérieure ou égale à la taille du message, autrement, le message chiffré sera égal au message en clair.

Le problème de cette méthode est donc qu'elle est très facilement cassable par brute-force car la quantité de clé est restreinte.

## Comparaison des méthodes

### Rapidité

Si nous comparons la rapidité des différentes méthodes citées précédemment, nous pouvons nous rendre compte que la plus rapide est la méthode des échanges d'indices des symboles du message. Une seule passe est nécessaire afin de chiffrer ou déchiffrer le message car il s'agit de simples permutations dictées par la clé.

Le chiffrement en dent de scie est également une méthode rapide, mais est très facile à casser, nous n'allons donc pas utiliser cette méthode.

Les autres algorithmes possèdent une complexité quadratique mais offrent une forte sécurité qui compense ce défaut.

### Sécurité

Si on veut un cryptage qui est difficile à déchiffrer, la méthode de la grille tournante semble être la plus efficace à cause de la complexité du déroulement de son algorithme. C'est donc la méthode la plus sécurisée parmi toutes celles citées précédemment. Malheureusement, le coût en temps est trop élevé.

Mis à part le chiffrement en dent de scie, les autres algorithmes proposent une sécurité suffisante qui est tout autant adaptée à notre sujet.

## **Choix de l'algorithme à implémenter et justification**

Comme nous allons implémenter une de ces méthodes sur de la réalité virtuelle, nous avons besoin d'un algorithme rapide pour éviter trop de latence entre la détection de l'image chiffrée et son déchiffrement, tout en conservant un minimum de sécurité. Nous avons donc décidé d'implémenter la méthode de la permutation des indices qui nous offre le meilleur compromis entre rapidité et sécurité.

## **Recalage par projection géométrique à partir d'une photo prise depuis un téléphone portable**

### **Création de la photo**

Une photo valable à traiter sera une photo qui contiendra une image chiffrée imprimée. Cette image devra être entièrement visible sur la photo, par ailleurs la qualité de la photo sera un facteur important quant au bon discernement des blocs de couleur de l'image (qui représentent les pixels sur une version numérique).

Nous choisirons une clé qui sera identique pour chaque image que nous allons chiffrer. Une fois fait, nous pourrions imprimer ces images pour les faire apparaître sur une photo. Afin de simplifier la détection sur quelle partie de la photo traiter pour effectuer le déchiffrement, nous allons entourer les images chiffrées d'un cadre épais et sombre (noir ou très proche du noir).

### **Détection de l'image à déchiffrer sur la photo**

Une idée serait d'utiliser une méthode de recalage géométrique : il s'agit d'une détection des contours afin de récupérer la délimitation du cadre du tableau chiffré que l'on veut récupérer sur une photo qui nous permettra d'identifier les pixels à traiter et de pouvoir cadrer l'image déchiffrée par dessus. C'est également une façon de détecter un motif pour l'utilisation de la réalité virtuelle.

### **Déchiffrement de l'image chiffrée détectée sur la photo**

Pour procéder au déchiffrement, nous pourrions créer une copie de cet élément chiffré en récupérant les pixels concernés à l'intérieur du cadre détecté avant. Étant donné que l'image ne sera pas parfaitement droite sur la photo, un pixel du tableau peut être à cheval sur deux ou plusieurs pixels de la photo. Nous pourrions donc calculer la valeur des pixels grâce à des méthodes de transformations.

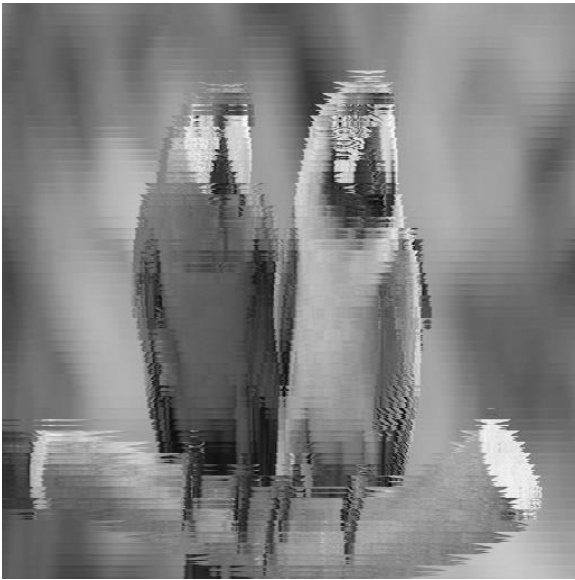

### **Restitution de l'image déchiffrée**

Une fois l'image déchiffrée obtenue, nous pourrions créer une copie de la photo d'origine en plaçant l'image déchiffrée à l'emplacement où elle était chiffrée à la base pour la remplacer. Il sera probablement nécessaire d'utiliser la même méthode de transformation utilisée précédemment mais inversée afin de bien recalculer l'image à l'emplacement défini.

## PARTIE II : ÉCRITURE DES ALGORITHMES ET RÉSULTATS

### Algorithme de chiffrement/déchiffrement par permutation

#### Méthode d'échanges des indices des symboles



	
Chiffrement avec une clé de taille 10.	Déchiffrement de l'image précédente.

La première étape a été de commencer le chiffrement avec une clé créée à la main pour vérifier que la méthode est bonne (voir exemple ci-dessous).

Exemple : `tabKey2[10]={9,7,4,1,0,8,3,5,2,6};`

Ici, la ligne 1 sera échangée avec la 9, la ligne 2 avec la 7 etc...

Cependant, avec une petite clé, l'image chiffrée est encore reconnaissable. Il nous faut alors générer une plus grande clé, donc (par facilité et car c'est aussi le but) créer une clé aléatoire selon une taille donnée. Mais au vu des résultats (voir image ci-dessous), nous pouvons nous rendre compte de la présence de doublons dans la clé (correction prévue pour le prochain rendu).

	
Chiffrement avec une clé de taille 100.	Déchiffrement de l'image précédente.

## Algorithme de détection des contours

### Calcul du gradient

Nous voulons pouvoir détecter efficacement la position de l'image que nous voulons déchiffrer, nous avons donc besoin de bien délimiter la zone à traiter. L'image sera alors entourée d'un cadre noir. Le calcul du gradient va permettre la détection de celui-ci.

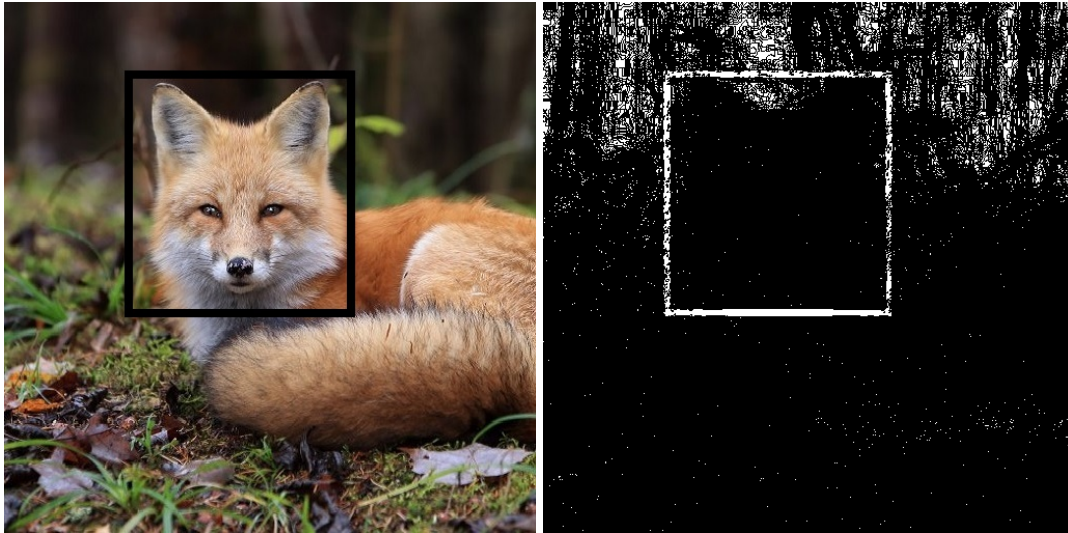
Nous allons donc calculer la dérivée verticale et horizontale de la photo.

Pour simplifier les calculs, nous allons convertir l'image RGB vers YCrCb, en ne conservant que la partie de la luminescence Y, où :

$$Y = 0.299 \times R + 0.587 \times G + 0.144 \times B$$

Ainsi, nous n'avons à traiter qu'une seule composante plutôt que trois.

Il suffit alors de calculer la dérivée horizontale et verticale en récupérant la différence de valeur de Y entre le pixel courant et le pixel suivant à sa gauche, et entre le pixel courant et le pixel suivant en dessous respectivement.



*Image originale test en RGB et sa dérivée en binaire*

Il est possible de réduire le bruit en utilisant une fermeture et ainsi révéler davantage le contour du cadre. Aussi, contrairement à l'image ci-dessus, le cadre est supposé être lui-même entouré de la couleur blanche de la feuille imprimée qui contiendra l'image chiffrée. Le contraste sera donc plus marqué, les contours du cadre seront plus visibles et moins "troués", et l'opération de fermeture sera alors plus efficace.

## **Travail prévu pour la semaine du 15/11 au 21/11**

- Terminer le chiffrement (réparer les bugs) et faire le chiffrement sur une image en couleur
- Effectuer des tests sur des photos pour comparer la précision à laquelle les cadres seront détectés et ainsi mieux adapter nos méthodes à ces cas-là ;
- Création de la fonction de transformation géométrique ;
- Récupérer les pixels à l'intérieur du cadre ;
- Mise en pratique sur une photo avec une image à déchiffrer.

## **Sources**

- Chiffrements:
  - <https://www.apprendre-en-ligne.net/crypto/python/transposition/index.html>
  - [https://en.wikipedia.org/wiki/Rail\\_fence\\_cipher](https://en.wikipedia.org/wiki/Rail_fence_cipher)
  - <https://www.apprendre-en-ligne.net/crypto/corriges/grtour4.html>
- Autres:
  - [Recalage d'image : http://www.traitement-signal.com/recalage.php](http://www.traitement-signal.com/recalage.php)