CR_2

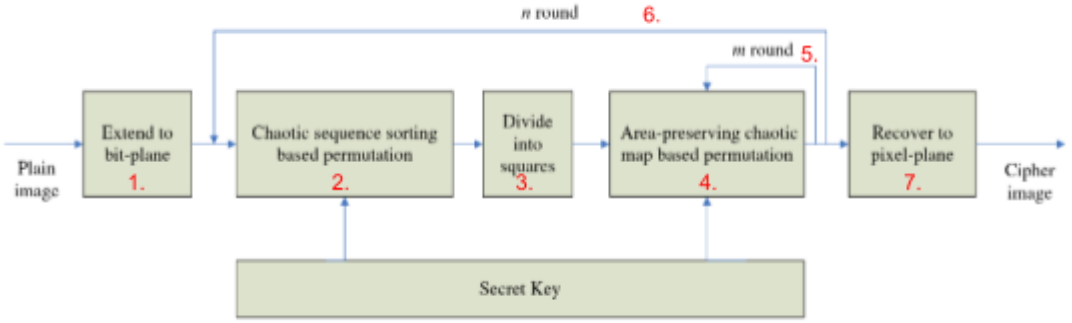
Dépôt Git du projet :
https://github.com/NeonVhenan/Projet_Image_IMAGINE

Dans la continuité de la réflexion démarrée lors des deux premières semaines,

Nous avons réalisé que l'article sur lequel nous voulions baser notre chiffrement ne nous convenait pas. En effet, différents éléments du principe de chiffrement nous était incompréhensibles, même après de longues heures d'étude (la taille des petites matrices nous semblait abstraite, ainsi que le fait que ces matrices doivent être 1-balanced alors que nous appliquons plutôt un shuffle dessus).

Nous sommes donc parties à la recherche d'un autre article sur lequel nous pourrions baser notre chiffrement. Nous avons trouvé l'article suivant : *A novel chaos-based bit-level permutation scheme for digital image encryption.*

Le principe est le suivant :



1. on prend l'image de taille nH * nW qu'on décompose en une image binaire de taille nH * nW * 8
2. on applique un algorithme de permutation des bit en utilisant un "chaotic sequence sorting" (séquence de tri chaotique)
   pour cela, on utilise :

   $$x_{n+1} = T_k(x_n) = \cos\left(k \cdot \cos^{-1} x_n\right), \quad x_n \in [-1, 1]$$

   avec x_0 et k, utilisé comme clé secrète de notre chiffrement
   on calcule la séquence de taille nH de la suite de Chebyshev ci-dessus dans X
   on tri cette liste de façon croissante dans Y, et on en déduit un vecteur de permutation de X à Y, ce vecteur de permutation, sera le vecteur de permutation appliqué aux lignes
   on recommence ensuite la même chose avec une séquence de taille nW * 8 pour permuter les colonnes

3. on divise ensuite l'image binaire sous forme de carrés de même taille N * N (blocs)
4. puis on permute les bits dans chacun de ces blocs en utilisant une permutation basée sur "Arnold Cat map"
   on utilise alors la formule suivante :

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} = \begin{bmatrix} 1 & p \\ q & pq+1 \end{bmatrix} \begin{bmatrix} x_n \\ y_n \end{bmatrix} \bmod N = \boldsymbol{A} \begin{bmatrix} x_n \\ y_n \end{bmatrix} \bmod N$$

   dans ce cas, p, q et M (le nombre d'itération), appartiendrait aussi à la clé secrète
   on aura alors le bit à la position (x_n, y_n) qui sera à la position (x_{n+1}, y_{n+1}) à l'étape suivante
5. On répète l'étape 4. M fois
6. On répète les étapes 2. à 5. N_0 fois
7. on reconstruit ensuite notre image binaire en une image en niveaux de gris

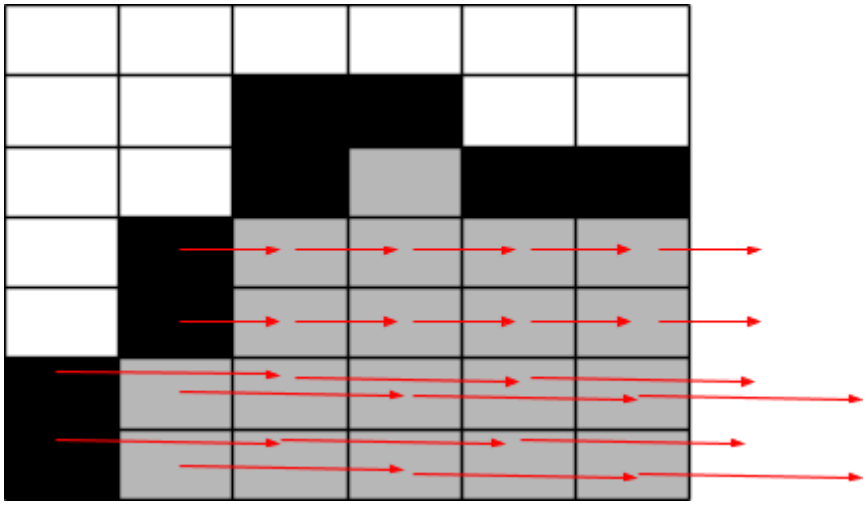
(Pour appliquer ce principe à une image en couleur, il nous suffira d'appliquer le principe au trois plans couleur (R, G et B)).

Notre clé secrète est donc composée de x_0, k, p, q et M.

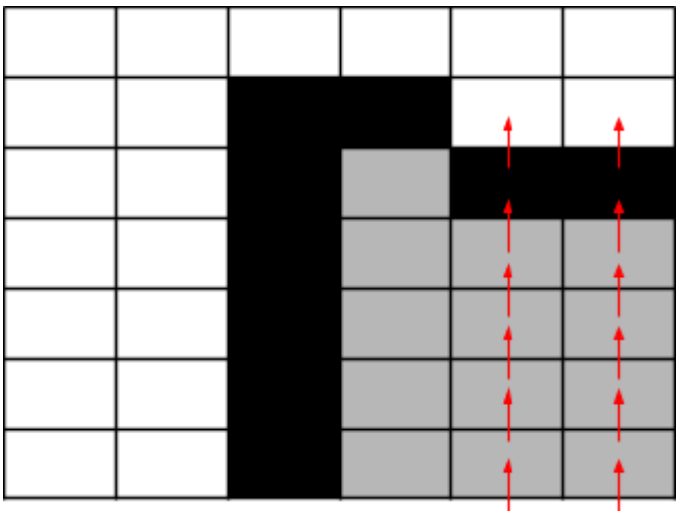
Pour le déchiffrement, on applique l'opération inverse : on recalcule le vecteur de permutation et on applique la permutation inverse (pour la première permutation), et on applique $\begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} = \begin{bmatrix} pq+1 & -p \\ -q & 1 \end{bmatrix} \begin{bmatrix} x_n \\ y_n \end{bmatrix} \bmod N.$ pour le deuxième type de permutation.

Le but de l'algorithme proposé, est d'obtenir de bon résultat en un temps minimum, et donc pour cela, ils comparent leur algorithme à un algorithme conventionnel de permutation seulement (qui se contente de permuter les pixels sans changer leur valeur), et à un algorithme de permutation et diffusion classique.

Idée concernant le recadrage : (en noir le cadre de notre peinture, en gris la peinture)



recalage horizontal



recalage vertical

Puis délimitation de la peinture avec le cadre noir, pour ne récupérer que les pixels de la peinture.