Projet Cryptographie
L'algorithme de chiffrement Solitaire

Contenu

Présentation de l'algorithme de chiffrement solitaire:

Bruce Schneier a mis au point à la fin du précèdent millénaire un algorithme de cryptage de donnée basé sur un jeu carte de 54 cartes. Ce dernier fut baptisé entre autre algorithme de chiffrement Solitaire. Le mode de fonctionnement est assez facile à comprendre, il comporte quelques étapes à suivre que voici.

Etape 0 : mélanger un paquet de carte de manière aléatoire.

Etape 1 : Déplacer le Joker noir sous la carte sur laquelle il se trouvait

Etape 2 : Descendre de deux cartes le Joker rouge par rapport à la position où il se trouvait

Etape 3 : Couper le paquet en 3 partie, de tel sorte que :

> La première partie, comporte toutes les cartes ce trouvant entre le haut du paquet, et le premier Joker trouvé exclu.
>
> La seconde tranche, comporte les cartes ce trouvant entre le premier Joker trouver (à partir du haut du paquet), et le second Joker inclus.
>
> La dernière tranche, c'est le reste du paquet, à savoir toutes les cartes qui se trouvent sous le second Joker à partir du haut du paquet.

On intervertie la première et la dernière tranche, et on replace les 3 paquets l'un sur l'autre, pour ne faire qu'un paquet comme au début.

Etape 4 : Les cartes ont une valeur entière comprise entre 1 et 54 inclus.

> De 1 à 13, ce sont des trèfles.
> De 14 à 26, ce sont des carreaux.
> De 27 à 39, ce sont des cœurs.
> Et de 40 à 52, des piques.
> Sans oublier 53, Joker Noir, et 54, joker Rouge.
>
> Toujours valet, dame et roi à la fin de chaque couleur
>
> Exemple :
>
> Cette carte  vaut 13.
>
> Celle-ci  vaut 46.
>
> Ou encore une dernière,  vaut 35…

On regarde la valeur de la dernière carte du paquet (celle qui se trouve tout à fait en dessous). Selon sa valeur, on revient en haut du paquet, et on avance d'autant de carte que fut la valeur de la carte que nous avons lue.

> Exemple : La dernière carte est un 7 de pique (vaut 46), on va chercher la 46^{ième} carte du paquet (toujours en partant du dessus).

De là, je prends toutes les cartes qui se trouve entre la carte qui suit celle que l'on a trouvé et la dernière carte exclue, et on place ce morceau en haut du paquet.

Etape 5 : Dernière étape. Ouf, on en voit le bout. Comme dans l'étape précédente, on va lire la valeur de la première carte, et on se déplace d'autant de carte qu'est la valeur que l'on a lue. Une fois retirer le nombre de cartes qui se trouve au-dessus, on regarde la carte qui reste, qui ce trouvera au-dessus du tas restant. Cette carte, c'est celle que l'on recherchait.

C'est gagné … à condition que la carte que l'on a prise ne soit pas un Joker.

Il est vrai que j'ai dit que c'est fini, mais, … en réalité, ça l'est à deux conditions. La première, est que la carte que l'on a prise ne soit pas un Joker. La seconde, que le texte à crypter soit composer d'un seul caractère.

Si ce n'est pas le cas, on note la valeur de la carte que l'on a trouvé (si ce n'est pas un Joker) avant de la replace dans le paquet, et on recommence les étapes de 1 à 5, avec comme paquet initial, le paquet à l'état qu'il était à l'étape 5 du mélange d'avant. Il faut récupérer autant de cartes valides que l'on a de caractères à crypter.

Voici un petit exemple de recherche de carte que vous pouvez retrouver sur l'application que j'on verra plus tard :

E0 générer un paquet:
11 36 25 2 4 54 39 48 42 50 5 52 23 46 10 13 9 1 29 38 15 8 16 14 44 26 3 45 18 43 28 41 27 34 40 19

49 51 6 22 30 37 24 7 33 35 17 21 32 53 31 20 47 12

E1 le Joker noir descend:

11 36 25 2 4 54 39 48 42 50 5 52 23 46 10 13 9 1 29 38 15 8 16 14 44 26 3 45 18 43 28 41 27 34

40 19 49 51 6 22 30 37 24 7 33 35 17 21 32 31 53 20 47 12

E2 Le Joker Rouge descend:

11 36 25 2 4 39 48 54 42 50 5 52 23 46 10 13 9 1 29 38 15 8 16 14 44 26 3 45 18 43 28 41 27 34

40 19 49 51 6 22 30 37 24 7 33 35 17 21 32 31 53 20 47 12

E3 Intervertir les parties par rapport aux Jokers :

20 47 12 **54** 42 50 5 52 23 46 10 13 9 1 29 38 15 8 16 14 44 26 3 45 18 43 28 41 27 34 40 19 49

51 6 22 30 37 24 7 33 35 17 21 32 31 **53** 11 36 25 2 4 39 48

E4 La valeur de la dernière carte est 48. Donc il faut lire la 48ième carte.
20 47 12 54 42 50 5 52 23 46 10 13 9 1 29 38 15 8 16 14 44 26 3 45 18 43 28 41 27 34 40 19 49 51 6

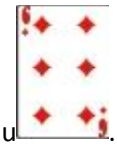
22 30 37 24 7 33 35 17 21 32 31 53 **11** 36 25 2 4 39 48

En place les cartes entre la 49ième et la 53ième positions en haut du paquet.

36 25 2 4 39 **20** 47 12 54 42 50 5 52 23 46 10 13 9 1 29 38 15 8 16 14 44 26 3 45 18 43 28 41 27

34 40 19 49 51 6 22 30 37 24 7 33 35 17 21 32 31 53 11 48
E5 La première carte vaut 36. Il faut donc lire la valeur de la 36ième carte+1 :
36 25 2 4 39 20 47 12 54 42 50 5 52 23 46 10 13 9 1 29 38 15 8 16 14 44 26 3 45 18 43 28 41 27 34

40 19 49 51 6 22 30 37 24 7 33 35 17 21 32 31 53 11 48

La 37ième vaut un 19 -> 6 de carreau  .

Et on recommence. Encore un tour, à partir du dernier paquet, et à l'étape E1.

E1 :

36 25 2 4 39 20 47 12 54 42 50 5 52 23 46 10 13 9 1 29 38 15 8 16 14 44 26 3 45 18 43 28 41 27 34 40

19 49 51 6 22 30 37 24 7 33 35 17 21 32 31 **53** 11 48

Et ainsi de suite, à chaque fois, on récupère la carte à chaque tour…

Dans l'alphabet, on a 26 caractères, que l'on peut écrire en minuscule ou en majuscule. Ce qui veut dire 26 minuscules, et 26 majuscules. Soit donc 52 caractères. Comme nous n'utilisons que 52 cartes pour le cryptage, plus deux Joker, j'ai décidé pour la suite du projet de faire correspondre toutes les cartes de valeur comprise entre 1 et 26 à leurs indices alphabétique dans l'ordre croissant. Et pour les cartes de valeurs entre 27 et 52, de les faire correspondre aux lettres alphabétiques majuscules.

Exemple :

19 -> s
36 -> J

Maintenant que l'on a généré une clef de la même taille que le mot ou le texte à crypter, on crypte le texte caractère par caractère par rapport par la clef par une addition des deux valeurs.

TEXTE+CLEF=CODE

Exemple : On prend le mot **test**. La clef générée est **sJtS**.

Le caractère t+ la clef associée s =le code généré M

Le caractère e+ la clef associée J =le code généré O

Le caractère s+ la clef associée t =le code généré M

Le caractère t+ la clef associée S =le code généré m

Résultat **MOMm**

Une copie du paquet initialement utiliser doit être conservé. En effet, pour décrypter, il faut pouvoir régénérer la clef de cryptage.

Exemple : Supposons que l'on veut transmettre le message codé. Le code étant **MOMm**, et le paquet initial : **11 36 25 2 4 54 39 48 42 50 5 52 23 46 10 13 9 1 29 38 15 8 16 14 44 26 3 45 18 43 28 41 27 34 40 19 49 51 6 22 30 37 24 7 33 35 17 21 32 53 31 20 47 12**

La clef générer par ce paquet sera **sJtS**.

Grace au code et à la clef de cryptage, on fait une soustraction caractère par caractère du code par la clef.

Mise en œuvre

Après avoir lu l'énoncé, j'ai déjà réfléchit à quel langage conviendrai pour développer cette application. J'ai choisi le PHP. Je ne suis pas un PRO PHP, loin de là même, je suis à peine initier à ce langage. Mais aujourd'hui, la mode est aux web apps, SaaS, cloud… J'ai donc voulut me lancer dans l'aventure, avec quelque chose d'autre que JAVA ou C/C++

On peut dire que j'ai pris un risque avec ce choix, mais comme j'avais reçu l'énoncé assez tôt, j'ai pu prendre le temps pour m'initier et voir si le projet pourra aboutir ou pas. De plus, sur un serveur web, il pourra me servir de service pour crypter et décrypter des données ou des mots de passes, un peu à la façon Open PGP. Ce qui envisage une vie au projet après le projet, et c'est ce qui est différent par rapport à la plus part des langages, qu'il faut compiler sur sa machine le code, qui du coup généralement après la présentation, il fait partie des archives.

Les possibilités qu'offre ce projet sont nombreuses. J'en ai mis en place certaines, mais il reste encore beaucoup de choses qui peuvent être faites.

Parmi les idées que j'ai pour le projet, c'est la fonction d'envoi par mail du code et du paquet (possible et faisable, mais nécessite d'être installé sur un serveur SMTP. Sinon, les emails ne partent pas). Idem pour l'envoie du paquet séparément du code, par SMS. J'utilise déjà un service qui me permet d'envoyer des SMS automatiquement. Il me faut juste coder l'API pour que ça marche, et le serveur SMTP pout l'envoi.

Il y a le codage de fichiers texte. Il suffit de lire le fichier, et de coder le texte. Cette fonctionnalité est intégrée au projet.

Une autre fonction intégrée au projet, le cryptage d'image complète OU partiel. Vous vous demandez ce que je veux dire par partiel. Ceci veut dire que vous pouvez coder qu'une zone de l'image si vous le voulez, plutôt que toute. Exemple, une image qui comporte le visage de quelqu'un, les plaques d'une voiture … une information à camoufler sur toute l'image, c'est très simple, plutôt que de crypter toute l'image, ne cryptez qu'une seule zone.

Une machine de Turing réputer assez robuste pour pouvoir décrypter ce genre de code, n'arrivera pas à trouve de suite une logique dans le codage. Le nombre de paquet différent est de l'ordre de 54! (54 factoriel). Autant dire quasi infini. Mais la faille reste sur le fait que si quelqu'un intercepte le paquet, et le code, et qu'il connaît le mode de cryptage, il lui sera facile de décrypter le code. C'est pourquoi, j'ai codé une fonction qui crypte aussi le paquet, par un algorithme de Feistel.

L'algorithme tel que décrit, ne prends pas en compte les caractères non alphabétiques de base ; ni caractères accentués, ni ponctuations, ni symboles… Il y aura alors de la perte sur le code. Je me suis inspiré de l'algorithme de compression LZ77 qui sert à sauvegarder ces données et les restituer au décryptage.

Le cryptage d'un pixel consiste à crypter un octet. Plusieurs façons de faire sont possibles. En ayant écrite une pour crypter les images, j'ai voulu élargir les possibilités vers le cryptage de tous les fichiers, en le cryptant octet par octet. Mais PHP n'est pas un vrai langage de programmation bas niveau, mais un langage de script. De ce fait, il y a des limites, entre autre et surtout quand il s'agit de gérer la mémoire. Ainsi, la lecture de fichier en binaire est possible selon le fichier. Mais pour ce qui est de l'écriture, il est impossible de faire une écriture octet/octet. Pour une lecture/écriture bit par bit, en JAVA ou C++, on utilisera un buffer par où passera les données. Mais PHP ne gère pas ce type d'outils. Il aura fallu utiliser du PYTHON par exemple pour y arrivée, ou peut-être du PERL voir du C (oui on peut faire du C dans un serveur WEB/HTTP/HTTPS.

Code source

Squelette

Pour des raisons d'organisation, l'application est décomposée en une dizaine de fichier. Certains d'entre eux sont partagés, d'autre pas.

Accueil.php

Le fichier à lancer en premier, c'est le point de départ du programme. D'ici, vous décidez de crypter des mots, des phrases, des fichiers textes, ou encore des images. Idem pour le décryptage, tout ce fait d'ici. Lors du cryptage d'un mot, vous pouvez aussi choisir le mode debug ou encore pas à pas pour suivre la compilation du fichier.

Crypter.php

Ce fichier est une interface qui regroupera toutes les données et fonctions pour crypter, et afficher le résultat. Il hérite des fichiers Randomchar.php/Strings.php/Fichiers.php/Jeu.php

Ici, la demande de génération de paquet de carte aléatoire ce fait, la clef de cryptage aussi, et les opérations de cryptages ce font. Le cryptage de fichier ce fait d'ici aussi.

Decrypter.php

Similaire au fichier **crypter.php**, mais pour décrypter. Hérite aussi randomchar.php/strings.php/fichiers.php/jeu.php

Debug.php

Fait presque la même chose que crypter.php, mais à la différence qu'ici on a une illustration des étapes de cryptage.

Courriel.php

Affiche les données résultats avec le paquet à utiliser, et le code si l'on crypte. Hérite du fichier Generateur.php

Generateur.php

Pour crypter les paquets, il fallait trouver une fonction qui permet de fournir des générateurs. C'est dans ce fichier que ce trouve toutes les fonctions pour tester et générer des listes. Ce fichier peut aussi être utilisé seul. Comment? Il suffit d'éditer le fichier, et de décommenter la ligne 5 du fichier.

Randomchar.php

C'est dans ce fichier que sont gérées toutes les fonctions pour réaliser les étapes de 1 à 5 de l'énoncé. Il inclut hérite le fichier Jeu.php

Jeu.php

Ici, c'est le monde du paquet de carte. C'est ici que toutes les fonctions de gestions et manipulations de carte et de paquet de carte sont écrites.

Strings.php

Pour gérer les caractères et les chaines de caractères, il faut écrire des fonctions spécifiques à nos besoins. C'est dans ce fichier que ce trouve ces fonctions (comme entre autre les fonctions d'addition du texte et de la clef, ou la suppression de caractère non alphabétique…

Fichiers.php

Ici, ce trouve toutes les fonctions de gestion de fichier, à savoir lecture, écriture, upload ou encore le codage et décodage d'octet…

Cimages.php

C'est le fichier qui contient l'interface graphique pour crypter une image, avec toutes les options qui lui sont associés. Aucun cryptage n'y a lieu. On ne fait qui lire les informations entrée par l'utilisateur, qui seront transmissent au fichier TraiteImage.php.

Dimages.php

C'est la même chose que Cimages.php, mais c'est pour saisir les informations nécessaires au décryptage.

TraiteImage.php

D'ici, on manipule les images. Après l'upload de ces dernières, on va trouver dans ce fichier tous les outils pour manipuler les images (lecture d'un pixel, parcours d'image, recherche de couleurs… Mais contrairement à ce que l'on pourra penser, c'est bien d'ici que l'on gère le cryptage et décryptage d'image, mais pour crypter/décrypter les pixels, je fais appel au fichier Fichiers.php, et en particulier aux fonctions de codage d'octet (car rappelons le, un pixel en vrai couleur, est composer de 3 octets, que l'on peut décomposer et traiter.

Classes et outils

Pour réaliser le projet, j'ai utilisé XAMPP pour émuler un serveur WEB avec compilateur PHP, notepad++ pour écrire le code source et la doc sur internet, en compagnie de mon navigateur favori, php.net/manual m'a beaucoup servi pour quasiment tout ce qui est partie PHP, et pour les autres parties (HTML, JS, CSS …), ce sont des recherches sur Google.com

Après avoir lu le sujet, pour émuler le paquet de carte, ça m'a semblé approprié d'utiliser une pile, ou quelque chose de la sorte. J'ai donc choisie les listes chainées, en anneau. De ce fait, j'ai écrit une classe Carte (oui, à mon grand étonnement, PHP gère la notion d'objet). Une carte est constitué d'une valeur et un suivant. La classe et toutes les fonctions qui concernent la manipulation des cartes et du paquet (déplacement, recherche…) sont écrites dans le fichier Jeu.php.

J'ai écrit une autre classe, la classe point, qui est utilisé lors du cryptage/décryptage partiel pour manipuler le point. La classe est écrite dans le fichier TraiteImages.php

Ce sont les deux seuls classes que j'ai utilisé pour ce programme.

J'ai téléchargé les images d'une page wikipédia. Il s'agissait d'une plaque avec toutes les cartes. Pour la découper, j'ai essayé de le faire en PHP, après quelques heures, je n'y arrivais pas (je n'avais pas encore envisagé de faire des manipulations sur les images). J'ai donc décidé de faire le programme avec Matlab, et pour écrire l'image (car le découpage était pas très compliqué) ; j'ai fait une nuit blanche pour aboutir à ce résultat. Ce qui explique ainsi que la coupe des images ne soit pas parfaite. Le code source Matlab, ainsi que l'original de l'image ce trouve dans le dossier outils avec le projet.

Comment ça marche ?

Nous allons voir un peu comment fonctionne cette usine à gaz. Pour cela, nous allons faire quelques simulations.

Crypter un fichier texte

Pour commencer, il faut aller sur la page **Accueil.php** -clic- **Crypter** -clic radio- **Fichier**. D'ici par défaut le type texte est déjà choisie. Il faut alors choisir son fichier texte, et envoyer le formulaire. Le formulaire est alors transmis au fichier **Crypter.php**. Comme ce n'est pas un simple texte, il faut alors uploader et lire le fichier. Il y a donc quelques étapes supplémentaires comparé au cryptage de mot ou de paragraphe.

On fait alors appel à la fonction ***upload(destination, fichier)*** du fichier Fichiers.php. En plus de faire l'upload, cette fonction remplace tous les caractères spéciaux pour éviter le bug, et effectue quelques opérations comme le déplacement… En retour, on a les erreurs ou **TRUE** si tout est OK.

Si le retour est **TRUE**, on va alors lire le fichier grâce à la fonction read_file(adresse). Le résultat de cette lecture sera alors stocker dans la variable **PHRASE**.

A ce stade, il n'y a plus de différence entre un cryptage de texte, et de fichier vidéo, car à la suite des opérations, on considère que l'on a un texte.

Il faut à présent lire et réécrire le texte. En effet, il faut que le texte ne comporte que des caractères alphabétiques de base. Pas d'accents, pas de ponctuation … juste du texte brute. C'est la fonction **rewritestring(texte)** qui s'en charge, et renvoie le texte brute. Cette fonction est gérer dans le fichier Strings.php

rewritestring lira les caractères un par un. Pour chaque caractère lu, je fais appel à la fonction checklettre(caractère) qui retourne TRUE si le caractère est dans l'alphabet, et FALSE si non. C'est ici qu'entre en jeu aussi la sauvegarde des caractères spéciaux.

J'ai mis en place une méthode inspiré de la compression LZ77. L'algorithme consiste à compter dès que l'on commence le nombre de caractères consécutif avec checklettre à TRUE, et ceux à FALSE.

Exemple : soit le texte "*l'algo que j'ai ajouté*". Je parcours les caractères un par un, et j'essaie de faire ce que j'ai expliqué précédemment. Et j'obtiens quelque chose de ce genre :
13; 0,; a,1; 1,'; a,4; 1, ; a,3 ;1, ; a,1; 1,'; a,2; 1, ; a,5; 1,é;

Explication : Le 13 au début, veux dire qu'il y a 13 fragments. Les fragments sont séparés par des "**;**". Chaque fragment commence par un caractère, le A ou par un chiffre, le 1 (sauf le second fragment qui commence par un zéro, car il s'attendait à commencer par un caractère spécial.

Si le fragment commence par une lettre, il est forcément suivi par un nombre, qui est le nombre de caractère à réécrire du texte brut avant d'incrémenter des caractères spéciaux. Donc si l'on déroule le cas présent, on écrit un caractère et on s'arrête ; le "L". Dans le segment suivant, on commence par un chiffre, le 1. On va alors ajouter tout le texte de la seconde partie du bloc, et on passe au bloc suivant. Le texte devient alors "l'". On continu, et on aura "l'algo", puis "l'algo "… Jusqu'à la fin avoir fusionner les 13 segments. Ce déroulement d'algorithme est celui du décryptage, mais il était nécessaire de le faire dans ce sens pour mieux l'expliquer.

Nous voilà à présent avec deux variables, une avec le texte brut, et l'autre avec la sauvegarde des caractères spéciaux.

Revenons au fichier Crypter.php Nous avons le texte avec uniquement des caractères alphabétiques. Nous pourrons donc le crypter. Mais problème, nous n'avons pas encore de paquet carte, ni de clef de cryptage. Pour générer le paquet de carte, j'ai pensé au début d'utilisé la programmation parallèle, elle aurai permet de simuler un remplissage avec des valeurs aléatoires autre qu'avec random(), mais PHP ne gère pas (ou je n'ai pas sût gérer) ce type de programmation. J'ai donc fait quelque chose de classique. Du coup, voilà comment ça se passe, une boucle répéter … tant que (do … while). Pour remplir la chaine, on remplit un tableau de 54 cases (de 1 à 54). Pour un résultat de random, si tab[random()] est vise, on y écrit cpt, et on incrémente cpt, jusqu'à ce que cpt =54 . On obtient ainsi un tableau de cartes aléatoire.

Depuis ce tableau, on génère la liste, la fonction genchaine relit les cartes l'une derrière l'autre, et renvoie la queue de la liste. C'est là que l'on peut commencer la génération de clef. La fonction generekey applique les étapes 1 à 5 pour manipuler les cartes et retourne la clef de cryptage de longueur égale à celle du texte brute.

A présent, s'il s'agit du cryptage d'un fichier, le code et la sauvegarde des caractères non alphabétique sont enregistrer dans deux fichiers dédier et séparer.

Le résultat est affiché à l'écran, mais n'est pas complet. Pour décrypter depuis mon application, le paquet est incomplet. Mais le code et la sauvegarde des caractères spéciaux sont les bons.

Pour des raisons de sécurités, je préfère, si l'utilisateur le souhaite, crypter aussi le paquet. Ainsi, une autre personne qui veut décrypter avec son application le code, ne pourra pas utiliser le paquet, il sera obsolète. Pour le crypter, je propose une liste de générateur, avec une fonction d'addition, la multiplication et le factoriel ne propose aucun générateur…

Voilà, vous venez de crypter un texte ou un fichier texte.

Décrypter un mot

La saisie du texte est la même que celle utilisait pour le cryptage, c'est pourquoi, je en vais pas refaire les explications de lecture de fichier, ici, le code est directement stocker dans une variable. Le paquet est aussi saisi depuis la même interface. Mais attention, il ne faut surtout pas laisser d'espace avant le premier chiffre du paquet, et de préférence aucun à la fin non plus.

Une fois le décryptage lancé, la chaine de caractère qui correspond au paquet est découpée en tableau par les espaces qui séparent les valeurs. La première case, correspond au code de décryptage, et le reste, le paquet. Si la première valeur est un 0, le paquet n'est pas crypté. Sinon, le paquet est crypté, et cette valeur est celle du cryptage. Il suffit alors générer le code avec ce même générateur, et de rechercher les correspondances de valeurs pour recomposer le paquet.

Une fois nous avons le paquet, on régénère une clef de la longueur du code avec ce paquet. Et là la fonction inverse est lancée pour recomposer le texte brut. Pour le texte avec prise en charge des caractères spéciaux, il y a la fonction régénère qui est décrite plus haut.

Crypter une zone d'image

Pour le découpage d'une image, j'ai eu beaucoup de mal au début, et c'est pourquoi je l'ai fait avec matlab pour découper les cartes. Mais arrivé ici, je ne pouvais plus me permettre de ne pas maitriser cette partie. J'ai alors dû me débrouiller pour y remédier, il m'a fallu faire beaucoup de recherches, pour comprendre comment parcourir une image, comment lire les couleurs de l'image, et comment écrire une image avec les bonnes couleurs comprendre qu'il faut utiliser du vrai couleur ou du niveau de gris.

Pour le cryptage, et le décryptage, c'est le cryptage de pixel, en appliquant la clef aux composantes du pixel fois 5. J'ai essayé de le faire sans multiplication, mais le résultat n »tait pas convaincant. Il est beaucoup plus efficace si l'on multiplie la valeur de la clef par 5.

J'ai aussi eu beaucoup de mal à trouver comment je pourrai faire pour placer les panneaux ( ) qui délimites la zone de cryptage ou de coupe. En effet, il fallait combiner à la fois le javascript et le CSS. Dans le reste de l'application, le javascript m'a causé beaucoup de difficultés, même si ce n'est pas la partie la plus visible du projet, mais qui le rends tout de même bien plus fonctionnel.

Commençons. Si l'on a choisi le codage d'une zone d'image, on a forcément choisie le point haut gauche, et le point bas droit. Pour exécuter le codage, on crée justement 3 objets points. Le premier, point, correspondra au point haut gauche (appelé $phg dans les sources), le second au point bas droit (appelé $pdb) et le dernier, le point appelé dans le code $tmp qui correspond aux dimensions de la zone à crypter/ décrypter.

Une fois les variables initialisées, le boulot commence. Il faut générer un paquet et une clef de cryptage. Pour le paquet, nous l'avons déjà vue comment ça marche dans la partie de cryptage de texte. Pour la partie génération de clef, il suffit de générer une clef de taille « $tmp->x * $tmp->y ». En plus français, on génère une clef de taille égale aux dimensions de de la zone à crypter. Une fois que c'est fait. Le programme parcourt la zone à crypter comme un tableau, et applique la fonction de cryptage d'octet. Il était prévu de proposer deux versions de cryptage, une rapide, et une lente. Quelle est la différence entre les deux ? La version rapide (qui est appliqué dans le projet) génère une clef de la taille de la zone à crypter. Le cryptage ce fait en multipliant la clef par 5 pour chacune des 3 composantes du pixel RGB. La version lente (pas faite dans ce projet), aurai consisté à générer une clef 3 fois plus longue que celle du cryptage rapide. Et de là, attribuer à chacune des composantes du pixel sa propre clef.

Une fois le cryptage fait, le programme réécrit la matrice dans un fichier .png. Le choix du PNG est uniquement pour conserver les propriétés des fichiers .gif, .png, .jpg… En d'autre terme, c'est le format le plus complet qui conserve les propriétés des autres formats (en dehors de la compression).

Une fois fini, le résultat apparait, avec un aspect bizarre à la place de la zone choisie de crypter, et qui peut rappeler l'époque CANAL+. Tout comme pour le cryptage de texte, on voie le paquet, mais pas de code, ni de caractère spéciaux cette fois. Le cryptage du paquet est toujours possible, et utilise aussi la même méthode.

Décrypter une zone d'image

Le début, est comme ce que l'on a pu voir avant, on récupère les point de la zone d'image à décrypter, c'est à l'utilisateur de saisir ces informations. On compose le paquet, comme je l'ai expliqué pour décrypter un texte. On génère la clef. Et on parcourt l'image telle une matrice, comme on l'a fait sur le cryptage d'image. Cette fois ci, on ne cryptera pas mais on décryptera l'octet. On écrit l'image, et c'est fait. Ça parait vraiment facile comment j'explique ça, mais si les étapes d'avant on bien était comprises, celle si ne devrai pas causer problème.

Couper une image

Cet outil n'est pas vraiment bien mis en évidence, et peut passer presque inaperçu. Je n'ai pas vraiment voulut trop passer du temps dessus, car ce n'ai pas vraiment une chose importante. Depuis l'interface utilisateur de codage par zone, on peut aussi choisir de couper l'image. Le code est simple, il se base sur ce que j'ai expliqué, il consiste à créer une image avec les dimensions de $tmp, et à y recopier directement les pixels sans les modifier depuis l'image d'origine... C'est une façon de rajouter un outil de plus qui peut être utile au projet. C'est un des outils que j'ai développé pour comprendre comment manipuler l'image. Il ne provient absolument pas d'un site sur internet, mais d'un simple outil que j'ai écrit pour mieux comprendre comment ça marche.

Conclusion

Je pense avoir parlé de tout ce que j'ai développé dans cette application (des points les plus pertinents). J'ai essayé de m'inspirer au maximum de ce que l'on a vu en cours pour sa réalisation. Cette initiation au cryptage m'a était bien instructive. J'aimerai bien l'utiliser dans l'avenir pour des usages personnels. J'ai approfondi ma maitrise du PHP. J'ai d'ailleurs trouvé le PHP assez abordable avec des notions de C et de JAVA. Mais par contre, j'ai vraiment eu beaucoup de mal avec le JavaScript et le CSS. J'ai donc pour aboutir à un résultat contourné les problèmes en usant de ruses. Comme je l'ai dit, contrairement aux apparences, j'ai vraiment travaillé beaucoup sur le graphisme, pour faire des effets et des positionnements pas toujours maitrisé.