La crypte des Stego sort de l'ombre

12 janvier 2017

1 le background

La stéganographie est l'art de cacher des informations dans un message, qu'il soit visuel, sonore, tactile, ... Il ne faut pas la confondre avec la symbologie qui consiste simplement à prendre une représentation indirecte d'une chose, qui est généralement difficile à décrire. Par exemple, les saints sont représentés avec une auréole, qui est un symbole visuel indiquant leur sainteté. Il n'y a donc en général rien de caché dans un symbole, à part si vous ne connaissez pas sa signification.

Il ne faut pas non plus confondre la stéganographie avec la cryptographie, même si l'objectif est le même, à savoir cacher quelque chose. En fait, la stéganographie est d'un point de vue humain plus « puissante » que la cryptographie car un contenu crypté est immédiatement repéré comme tel, alors qu'un message caché par stéganographie l'est dans un contenu en apparence anodin.

Depuis que l'homme sait écrire, il a très souvent utilisé cet art dans ses textes. Généralement, cela revient à lire seulement certains mots ou certaines lettres pour découvrir le message caché. On dénombre par exemple de nombreux ouvrages d'ésotérisme basés sur ce principe pour offusquer les connaissances secrètes réservées aux initiés (c'est-à-dire ceux qui connaissent le schéma de masquage). Les complotistes diraient certainement que la Bible, le Coran et la Torah sont ainsi mais c'est simplement dû au fait que dans plusieurs milliers de pages, il y a forcément une façon tordue de lire qui fasse apparaître à peu près n'importe quoi, dont le fameux chiffre 42 ! Il y a également de nombreuses lettres à double lecture, comme la célèbre lettre de George Sand à Alfred de Musset (qui se trouve être un faux). Allez voir sur

`http://www.cryptage.org/lettre-george-sand.html`

Elle est assez édifiante, dans tous les sens du terme !

Plus récemment, l'homme a utilisé les images numériques pour cacher des copyright mais aussi des messages (comme dans les films d'espionnage). Il existe différentes techniques mais la plus simple consiste à modifier légèrement certains pixels, sans que cela soit visible pour l'oeil humain. Par exemple, quand un pixel est codé sur trois octets `(R,V,B)`, si on passe la valeur d'une composante de 235 (11101011) à 234 (11101010), ou bien de 122 (01111010) à 123 (01111011), cela sera imperceptible à un humain (surtout

avec le bleu). On peut donc utiliser ce principe en modifiant les bits de poids faible de tout ou une partie des pixels de l'image.

2 l'énoncé

L'objectif est de retrouver un message crypté grâce à l'algorithme RSA, puis caché par stéganographie dans une image couleur. Le principe de cryptage est le suivant :

- soit le message msg composé caractères ASCII. Chaque caractère correspond à un entier sur 8 bits,
- soit `bit(`msg`)` tous les bits associés à chaque caractère, dans l'ordre d'apparition.
- soit $m =$ `val(bit(`msg`))` la valeur entière représentée par ces bits.
- soit n le produit de deux nombres premiers p et q.
- soit pub la clé publique et $priv$ le clé privée, obtenues grâce à p et q.
- alors m_{crypt} est le **cryptage** de m (et donc de msg) grâce au couple (n, pub).
- alors le **décryptage** de m_{crypt} grâce au couple $(n, priv)$ permet d'obtenir m.

Par exemple, prenons $msg =$ `abc`. D'après la table ASCII, `'a'` vaut 97, `'b'` vaut 98 et `'a'` vaut 99, ce qui se traduit en binaire par respectivement `01100001`, `01100010` et `01100011`.

La concaténation dans l'ordre donne : `bit(`msg`) = 011000010110001001100011` donc $m = 6513249$. Bien entendu, le cryptage par le couple (n, pub) va donner un chiffre absolument énorme, qui une fois écrit en binaire, va comporter plus de 1000 bits (NB : pour voir comment est fait ce cryptage, référez-vous au canevas de code fourni).

La partie stéganographie est faite comme suivant :

- soit une image de taille $w \times h$, avec w la largeur et h la hauteur. Son contenu est une suite de triplets `(R,V,B)`, chaque composante étant stockée sur un octet. Il faut donc $3 \times w \times h$ octets pour stocker l'image.
- soit $n^2 =$ `bit(`n`)` la valeur binaire de n, recadrée sur $w \times h$ bits,
- soit $priv^2 =$ `bit(`$priv$`)` la valeur binaire de $priv$, recadrée sur $w \times h$ bits,
- soit $m^2_{crypt} =$ `bit(`m_{crypt}`)` la valeur binaire de m_{crypt}, recadrée sur $w \times h$ bits
- alors pour un pixel j représenté par un triplet `(R,V,B)`, et un entier i :
  - le i^{eme} bit de n^2 va être stocké dans le bit de poids faible de `R`,
  - le i^{eme} bit de $priv^2$ va être stocké dans le bit de poids faible de `V`,
  - le i^{eme} bit de m^2_{crypt} va être stocké dans le bit de poids faible de `B`.

Le résultat est ensuite enregistré à la place de l'image originale.

Concernant le recadrage, voici une précision utile : n^2, $priv^2$ et m^2_{crypt} ont à priori des tailles différentes. Le recadrage consiste à compléter les bits de poids forts avec des 0 jusqu'à ce que leur taille recadrée soit $w \times h$. Par exemple, si $n^2 =$ `1011` et que la taille de l'image est 3×2, alors la valeur recadrée est `001011`. On obtient ainsi des valeurs recadrées ayant toutes la même taille en binaire.

En revanche, ce que l'énoncé ne précise pas, c'est la correspondance entre j et i, c'est-à-dire quel est le pixel j qui va être modifié avec le i^{eme} bit des valeurs recadrées. A vous de trouver cette correspondance, sachant qu'elle n'est pas très compliquée.

Pour retrouver le message caché, votre programme doit donc :

1. lire une image au format PPM sur l'entrée standard (NB : déjà fait dans le canevas de code),
2. récupérer n^2, $priv^2$ et m_{crypt}^2 dans cette image et en déduire n, $priv$ et m_{crypt},
3. décrypter m_{crypt} grâce au couple $(n, priv)$, pour obtenir m
4. obtenir msg à partir de m et l'afficher sur la sortie standard.

Conseil : le canevas fourni est en python car ce langage permet une manipulation facile de très grands nombres, ce qui n'est pas le cas du Java ou C. Il est donc très fortement conseillé d'utiliser ce langage.

3 les ressources

Pour vous aider dans la réalisation du programme, vous trouverez sur
`http://cours-info.iut-bm.univ-fcomte.fr`
un article dans la section `hackaton` → `édition 2017`, portant le même titre que l'exercice. Il contient un lien permettant de télécharger un canevas de code en python, ainsi que deux fichiers images pour tester, avec le résultat attendu.

Bien entendu, vous êtes libres d'utiliser ou non ce canevas, mais c'est un gain de temps que de s'en servir comme base.