# La stéganographie, ou l'art du camouflage

Freetux

*Date de publication : 22 février 2015*
*Dernière modification : 24 février 2015*

## Introduction

C'est en lisant un livre de Franck Thilliez, « Train d'enfer pour ange rouge », que j'ai eu l'idée d'écrire ce billet sur la stéganographie, technique que je connaissais déjà depuis longtemps mais que j'avais un peu oublié depuis.

D'un point de vue général, la stéganographie est une technique de dissimulation : cacher un message dans un autre pour que celui-ci passe inaperçu.

En informatique, cette technique s'est développée au travers de la dissimulation d'un fichier dans un autre, généralement un texte ou une image dans une image ou un fichier audio. L'intégration de l'information dans le fichier « couverture » se fait sur via la modification de bits de poids faible du fichier (certains pixels d'une image par exemple). De ce fait, celui-ci n'est quasiment pas altéré et donc la présence d'une information cachée est très difficilement détectable par l'homme.

Une bonne illustration de la stéganographie, c'est le camouflage de certaines espèces animales, passées maîtres en la matière :

*Un Uroplatus phantasticus (source : 10 amazing animals avec ninja like camouflage).*

 CC-BY

## Installation

Le logiciel, libre bien évidement, que je vais détailler s'appelle **Steghide** et est présent dans toutes les distributions dignes de ce nom.

## La documentation

La documentation de ce logiciel est assez courte. Il existe seulement quelques commandes pour l'utiliser :

- Le premier argument :
  - `info`, `--info` : afficher les informations sur le fichier de couverture ou le fichier stéganographié ;
  - `embed`, `--embed` : cacher des données ;
  - `extract`, `--extract` : extrait les informations cachées à partir de données stéganographiées ;
  - `info`, `--info` : affiche les informations sur un fichier de couverture ;
  - `encinfo`, `--encinfo` : affiche une liste des algorithmes de chiffrement gérés.
- Les options principales de camouflage :
  - `-ef`, `--embedfile` : sélectionne le fichier de données ;
  - `-cf`, `--coverfile` : sélectionne le fichier couverture ;
  - `-sf`, `--stegofile` : sélectionne le fichier stéganographié ;
  - `-e`, `--encryption` : sélectionne les paramètres de chiffrement ;
  - `-e` : précise l'algorithme de chiffrement et/ou le mode
    `-e none` : ne chiffre pas les données avant de les cacher ;
  - `-z`, `--compress` : compresse les données avant inclusion (par défaut) (1, meilleure vitesse... 9, meilleure compression) ;
  - `-Z`, `--dontcompress` : ne pas compresser les données avant inclusion ;
  - `-q`, `--quiet` : supprime les messages d'information ;
  - `-v`, `--verbose` : affiche des informations détaillées.
- Les options d'extraction :
  - `-sf`, `--stegofile` : sélectionne le fichier stéganographié ;
  - `-xf`, `--extractfile` : sélectionne le fichier pour les données extraites.

 CC-BY

Les différents algorithmes de chiffrement et modes possibles sont les suivants :

- cast-128 : cbc cfb ctr ecb ncfb nofb ofb
- gost : cbc cfb ctr ecb ncfb nofb ofb
- rijndael-128 : cbc cfb ctr ecb ncfb nofb ofb
- twofish : cbc cfb ctr ecb ncfb nofb ofb
- arcfour : stream
- cast-256 : cbc cfb ctr ecb ncfb nofb ofb
- loki97 : cbc cfb ctr ecb ncfb nofb ofb
- rijndael-192 : cbc cfb ctr ecb ncfb nofb ofb
- saferplus : cbc cfb ctr ecb ncfb nofb ofb
- wake : stream
- des : cbc cfb ctr ecb ncfb nofb ofb
- rijndael-256 : cbc cfb ctr ecb ncfb nofb ofb
- serpent : cbc cfb ctr ecb ncfb nofb ofb
- xtea : cbc cfb ctr ecb ncfb nofb ofb
- blowfish : cbc cfb ctr ecb ncfb nofb ofb
- enigma : stream
- rc2 : cbc cfb ctr ecb ncfb nofb ofb
- tripledes : cbc cfb ctr ecb ncfb nofb ofb

La liste est trop longue pour présenter tous les algorithmes de chiffrement ici, du coup j'ai mis un lien vers les pages Wikipédia correspondantes. Pour les modes de chiffrement je vous invite à lire cette page. Pour faire simple, je vous conseille d'utiliser le mode « cbc » qui a fait ses preuves.

Par défaut, Steghide utilise l'algorithme rijndael-128 en mode cbc, qui correspond au standard AES qui fait office de valeur sûre à l'heure actuelle.

Après, si vous êtes intéressés par d'autres algorithmes, il vous est possible de les utiliser avec Steghide. :-)

## Dans la pratique

Comme la stéganographie est une technique « destructive » dans le sens où l'information dissimulée va dégrader le support, il n'est donc pas possible de stocker autant de données que l'on souhaite dedans sans dégrader visuellement le support. Si vous souhaitez ajouter un texte ou une image dans une image, vous

 CC-BY

devez alors connaître le volume maximum que celle-ci peut stocker en tapant cette commande :

```
$ steghide --info image.jpg
```

Le retour de la commande est celui-ci (tapez n à la question posée pour le moment) :

```
"image.jpg":
  format: jpeg
  capacité: 9,1 KB
Essayer d'obtenir des informations à propos des données incorporées ?
(o/n) n
```

Une fois que nous connaissons le volume maximal intégrable dans cette image (ici 9.1 KB), on peut y dissimuler l'information. La commande basique est la suivante :

```
$ steghide embed -cf image.jpg -ef texte.txt
```

Une *passphrase* vous sera alors demandée (deux fois), et si tout s'est bien passé le message suivant apparaîtra :

```
embedding "texte.txt" in "image.jpg"... done
```

Si vous regardez la taille de l'image avant et après le embed, vous verrez que celle-ci est désormais un peu plus volumineuse. En retapant la commande `$ steghide --info image.jpg`, il est possible d'obtenir des informations sur les données incorporées. Cette fois-ci, tapez o pour avoir la suite de la réponse :

```
"image.jpg":
  format: jpeg
  capacité: 9,1 KB
Essayer d'obtenir des informations à propos des données incorporées ?
(o/n) o
Entrez la passphrase:
  fichier à inclure "texte.txt":
    taille: 3,5 KB
    cryptage: rijndael-128, cbc
    compression: oui
```

Enfin, pour extraire le fichier texte incorporé dans l'image, il suffit de lancer cette commande :

```
$ steghide extract -sf image.jpg
```

 CC-BY

Une fois la *passphrase* donnée, vous retrouverez votre fichier texte initial.

Pour plus de personnalisation, vous pouvez utiliser les différentes options détaillées ci-dessus. Par exemple, pour utiliser le chiffrement enigma sans compression des données à cacher dans l'image c'est cette commande :

```
$ steghide embed -cf image.jpg -ef texte.txt -e enigma stream -Z
```

## Quelques limites de la stéganographie

Bien que cette technique de dissimulation soit intéressante, elle a cependant une limite d'un point de vue sécurité : c'est une méthode symétrique. Par conséquent, si vous souhaitez envoyer des données secrètes via cette méthode, il vous faudra trouver une autre solution pour envoyer la *passphrase* à votre destinataire. Une solution est le chiffrement asymétrique comme le PGP.

Vous me direz, « mais du coup quel intérêt d'utiliser la stéganographie si on peut envoyer des messages chiffrés par PGP ? ». Indépendamment du côté « marrant » de cette techno, il est possible de faire passer des messages de façon quasi-invisible par d'autres personnes (des services de renseignements un peu trop curieux par exemple) contrairement à des courriels chiffrés. En effet, la stéganographie peut-être utilisée sur une page Web publiques ou encore dans des pièces jointes anodines alors qu'un courriel chiffré peut facilement être détecté, la seule protection reste l'algorithme de chiffrement utilisé.

Une autre application possible de ce logiciel est l'enregistrement de certaines informations relatives à des photos publiées sur le Net, une licence ou l'auteur par exemple. À une époque j'utilisais cette technique pour mes propres photos.

## Petit jeu !

Un message est dissimulé dans l'image du Uroplatus phantasticus par l'intermédiaire de Steghide. Sauriez-vous le retrouver ? ;-)

## Source

- Page Wikipédia de la stéganographie : https://fr.wikipedia.org/wiki/Steganographie ;
- Site officiel de Steghide : http://steghide.sourceforge.net.

 CC-BY