# OpenPuff v4.01 Steganography & Watermarking

Data hiding and watermarking made easy, safe and free

**EmbeddedSW © 2018**

Send your suggestions, comments, bug reports, requests to *embedded@embeddedsw.net* – *Skype "embeddedsw.company"*

## OpenPuff HomePage

## Legal remarks

Remember: this program was not written for illegal use. Usage of this program that may violate your country's laws is severely forbidden. The author declines all responsibilities for improper use of this program.

No patented code or format has been added to this program.

**This is a Free Software:**

This software is released under LGPL 3.0

You're free to copy, distribute, remix and make commercial use of this software under the following conditions:

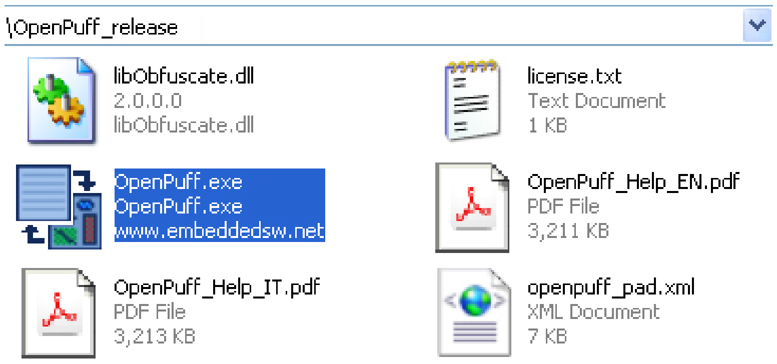
- You have to cite the author (and copyright owner): www.embeddedsw.net
- You have to provide a link to the author's Homepage: www.embeddedsw.net/openpuff.html

Back

## OpenPuff installation: Windows

This program was written to get you maximum privacy and compatibility:

- Portable application, no need to apply any installation procedure
- No dependency on other software/library
- Supported from WinNT up to Win10, 32bit and 64bit architectures


\OpenPuff_release

libObfuscate.dll
2.0.0.0
libObfuscate.dll

license.txt
Text Document
1 KB

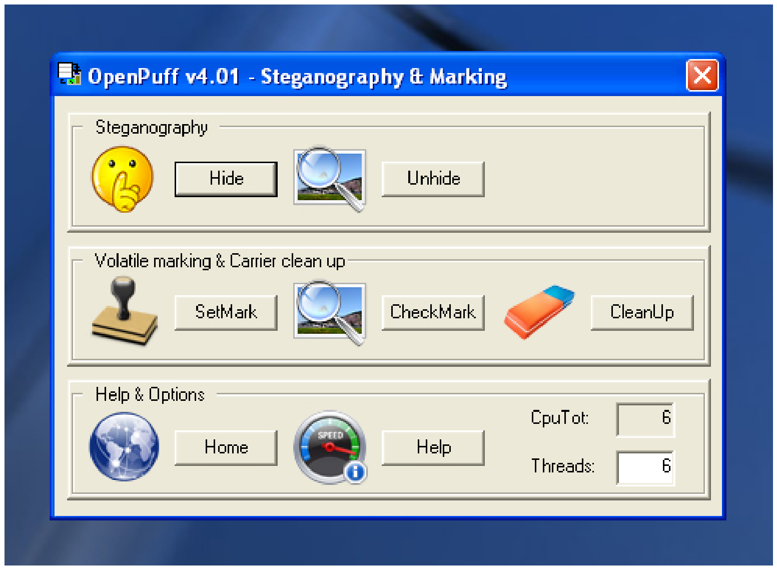
OpenPuff.exe
OpenPuff.exe
www.embeddedsw.net

OpenPuff_Help_EN.pdf
PDF File
3,211 KB

OpenPuff_Help_IT.pdf
PDF File
3,213 KB

openpuff_pad.xml
XML Document
7 KB

*Extract compressed release and run OpenPuff.exe*


OpenPuff v4.01 - Steganography & Marking
Steganography
Hide
Unhide
Volatile marking & Carrier clean up
SetMark
CheckMark
CleanUp
Help & Options
Home
Help
CpuTot: 6
Threads: 6

*Direct access to the main panel*

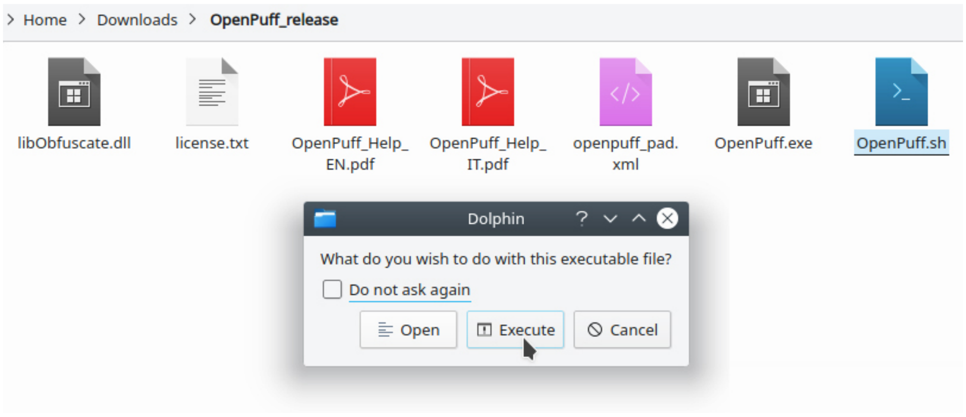
Back

## OpenPuff installation: Linux

This program was written to get you maximum privacy and compatibility:

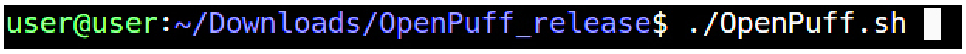
- The only dependency is on Wine
- Automated shell to install/run on Ubuntu provided (*OpenPuff.sh*)
- Automated shell to uninstall/cleanup on Ubuntu provided (*Uninstall.sh*)

### Install/Run:



What do you wish to do with this executable file?

*Extract compressed release and run OpenPuff.sh*

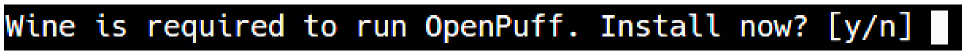


```
user@user:~/Downloads/OpenPuff_release$ ./OpenPuff.sh
```

*You can also run OpenPuff.sh by command line*

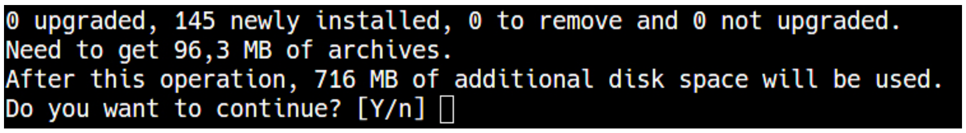
### Wine not installed:

In case Wine is not installed on your system, automated shell will alert you.
Type [y] to confirm you accept to install Wine and continue.



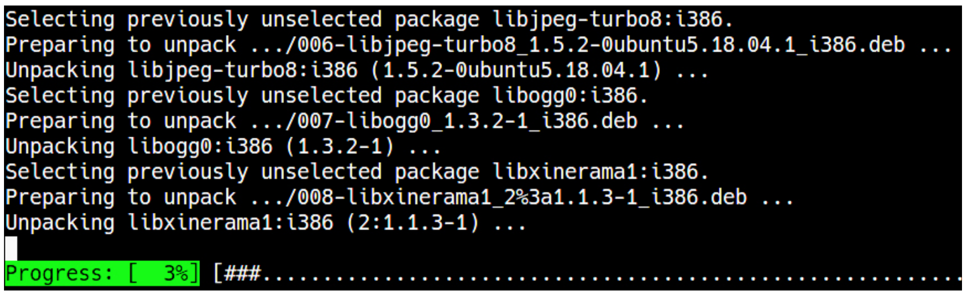
```
Wine is required to run OpenPuff. Install now? [y/n]
```

*Confirm [y] to accept to install Wine and continue*



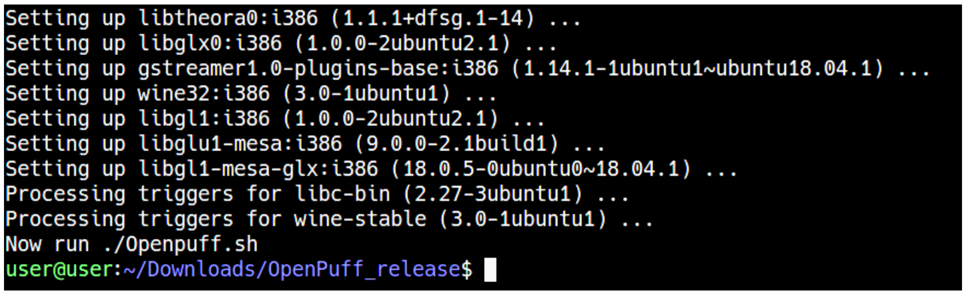
```
0 upgraded, 145 newly installed, 0 to remove and 0 not upgraded.
Need to get 96,3 MB of archives.
After this operation, 716 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

*Confirm [y] to allow linux to download and install requested packages from internet*



```
Selecting previously unselected package libjpeg-turbo8:i386.
Preparing to unpack .../006-libjpeg-turbo8_1.5.2-0ubuntu5.18.04.1_i386.deb ...
Unpacking libjpeg-turbo8:i386 (1.5.2-0ubuntu5.18.04.1) ...
Selecting previously unselected package libogg0:i386.
Preparing to unpack .../007-libogg0_1.3.2-1_i386.deb ...
Unpacking libogg0:i386 (1.3.2-1) ...
Selecting previously unselected package libxinerama1:i386.
Preparing to unpack .../008-libxinerama1_2%3a1.1.3-1_i386.deb ...
Unpacking libxinerama1:i386 (2:1.1.3-1) ...

Progress: [  3%] [###..................................................
```

*Wait for 100%*

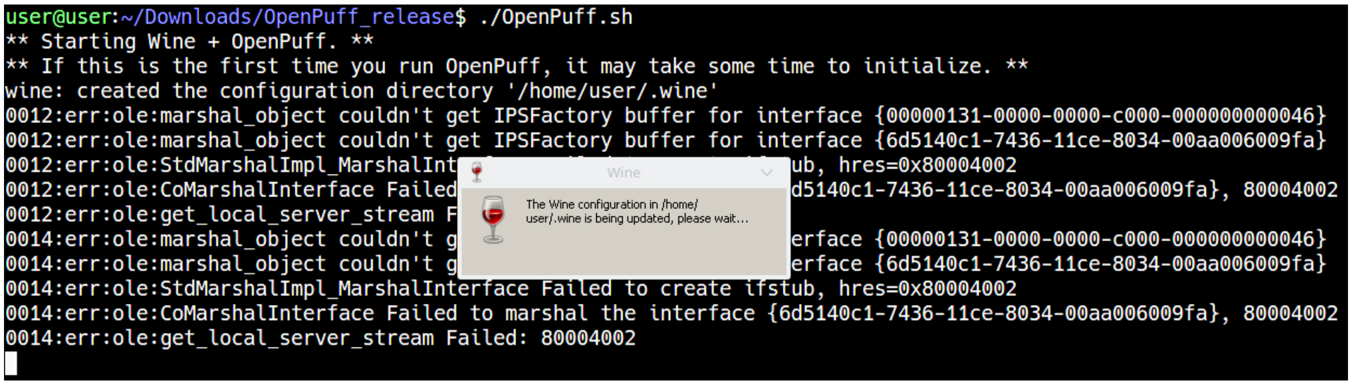


```
Setting up libtheora0:i386 (1.1.1+dfsg.1-14) ...
Setting up libglx0:i386 (1.0.0-2ubuntu2.1) ...
Setting up gstreamer1.0-plugins-base:i386 (1.14.1-1ubuntu1~ubuntu18.04.1) ...
Setting up wine32:i386 (3.0-1ubuntu1) ...
Setting up libgl1:i386 (1.0.0-2ubuntu2.1) ...
Setting up libglu1-mesa:i386 (9.0.0-2.1build1) ...
Setting up libgl1-mesa-glx:i386 (18.0.5-0ubuntu0~18.04.1) ...
Processing triggers for libc-bin (2.27-3ubuntu1) ...
Processing triggers for wine-stable (3.0-1ubuntu1) ...
Now run ./Openpuff.sh
user@user:~/Downloads/OpenPuff_release$
```

*Wine has been succesfully installed. Run OpenPuff.sh again*

## Wine installed:

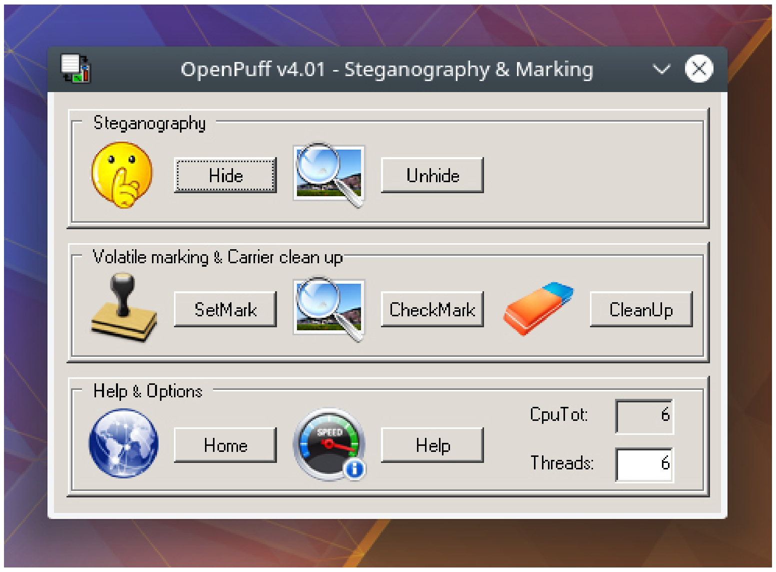
The first time you run Wine + OpenPuff, it may take some time to configure Wine environment.



```
user@user:~/Downloads/OpenPuff_release$ ./OpenPuff.sh
** Starting Wine + OpenPuff. **
** If this is the first time you run OpenPuff, it may take some time to initialize. **
wine: created the configuration directory '/home/user/.wine'
0012:err:ole:marshal_object couldn't get IPSFactory buffer for interface {00000131-0000-0000-c000-000000000046}
0012:err:ole:marshal_object couldn't get IPSFactory buffer for interface {6d5140c1-7436-11ce-8034-00aa006009fa}
0012:err:ole:StdMarshalImpl_MarshalInt                                  ub, hres=0x80004002
0012:err:ole:CoMarshalInterface Failed                                  d5140c1-7436-11ce-8034-00aa006009fa}, 80004002
0012:err:ole:get_local_server_stream F
0014:err:ole:marshal_object couldn't g                                  erface {00000131-0000-0000-c000-000000000046}
0014:err:ole:marshal_object couldn't g                                  erface {6d5140c1-7436-11ce-8034-00aa006009fa}
0014:err:ole:StdMarshalImpl_MarshalInterface Failed to create ifstub, hres=0x80004002
0014:err:ole:CoMarshalInterface Failed to marshal the interface {6d5140c1-7436-11ce-8034-00aa006009fa}, 80004002
0014:err:ole:get_local_server_stream Failed: 80004002
```

Wine
The Wine configuration in /home/user/.wine is being updated, please wait...

*Wine takes some time to setup environment, first time you run OpenPuff.sh*

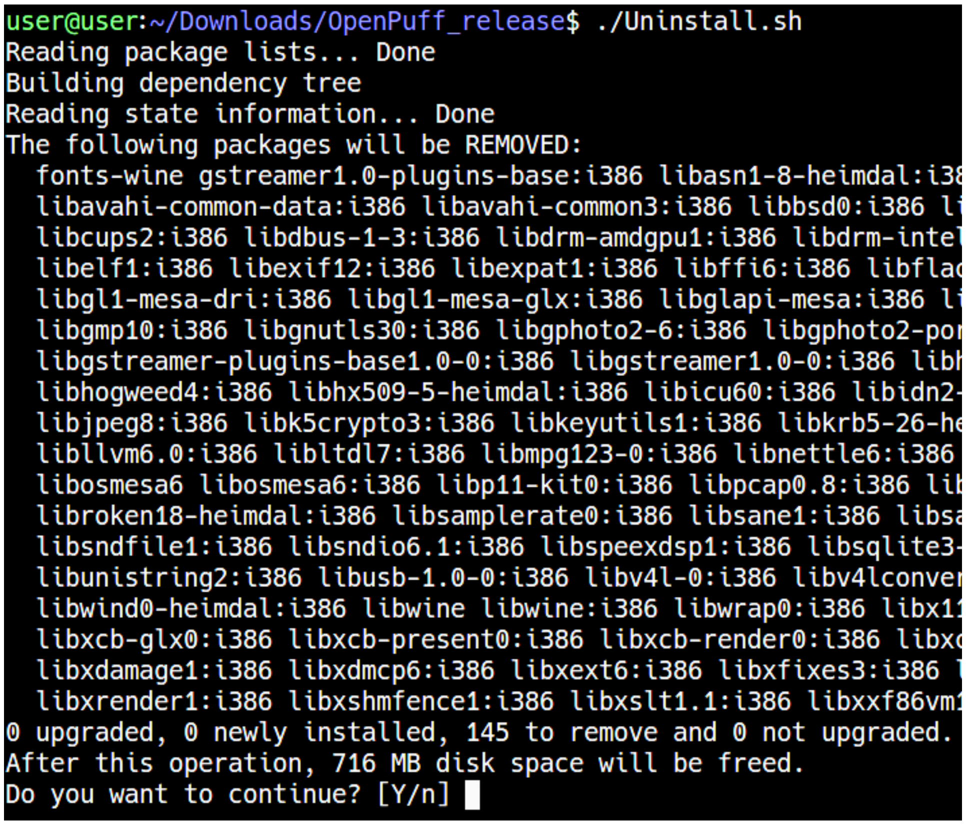


*Direct access to the main panel*

## UNINSTALL/CLEANUP

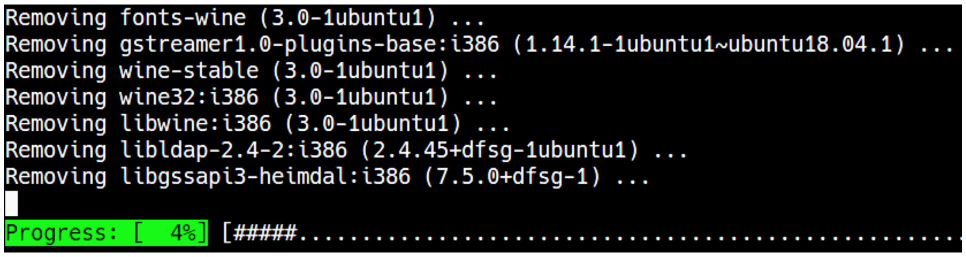
To fully remove this program, be sure to run the automated shell:

- Removing Wine settings (*~/.wine*)
- Uninstalling Wine and dependecy packages



```
user@user:~/Downloads/OpenPuff_release$ ./Uninstall.sh
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages will be REMOVED:
  fonts-wine gstreamer1.0-plugins-base:i386 libasn1-8-heimdal:i38
  libavahi-common-data:i386 libavahi-common3:i386 libbsd0:i386 li
  libcups2:i386 libdbus-1-3:i386 libdrm-amdgpu1:i386 libdrm-inte
  libelf1:i386 libexif12:i386 libexpat1:i386 libffi6:i386 libfla
  libgl1-mesa-dri:i386 libgl1-mesa-glx:i386 libglapi-mesa:i386 li
  libgmp10:i386 libgnutls30:i386 libgphoto2-6:i386 libgphoto2-po
  libgstreamer-plugins-base1.0-0:i386 libgstreamer1.0-0:i386 lib
  libhogweed4:i386 libhx509-5-heimdal:i386 libicu60:i386 libidn2-
  libjpeg8:i386 libk5crypto3:i386 libkeyutils1:i386 libkrb5-26-he
  libllvm6.0:i386 libltdl7:i386 libmpg123-0:i386 libnettle6:i386
  libosmesa6 libosmesa6:i386 libp11-kit0:i386 libpcap0.8:i386 lib
  libroken18-heimdal:i386 libsamplerate0:i386 libsane1:i386 libsa
  libsndfile1:i386 libsndio6.1:i386 libspeexdsp1:i386 libsqlite3-
  libunistring2:i386 libusb-1.0-0:i386 libv4l-0:i386 libv4lconve
  libwind0-heimdal:i386 libwine libwine:i386 libwrap0:i386 libx1
  libxcb-glx0:i386 libxcb-present0:i386 libxcb-render0:i386 libx
  libxdamage1:i386 libxdmcp6:i386 libxext6:i386 libxfixes3:i386
  libxrender1:i386 libxshmfence1:i386 libxslt1.1:i386 libxxf86vm
0 upgraded, 0 newly installed, 145 to remove and 0 not upgraded.
After this operation, 716 MB disk space will be freed.
Do you want to continue? [Y/n]
```

*Run Uninstall.sh and confirm [y] to allow linux to uninstall*



```
Removing fonts-wine (3.0-1ubuntu1) ...
Removing gstreamer1.0-plugins-base:i386 (1.14.1-1ubuntu1~ubuntu18.04.1) ...
Removing wine-stable (3.0-1ubuntu1) ...
Removing wine32:i386 (3.0-1ubuntu1) ...
Removing libwine:i386 (3.0-1ubuntu1) ...
Removing libldap-2.4-2:i386 (2.4.45+dfsg-1ubuntu1) ...
Removing libgssapi3-heimdal:i386 (7.5.0+dfsg-1) ...

Progress: [  4%] [#####.............................................
```

*Wait for 100%*

BACK

## Features: why is this steganography tool different from the others?

OpenPuff is a professional steganography tool, with unique features you won't find among any other free or commercial software. OpenPuff is 100% free and suitable for highly sensitive data covert transmission.
What is steganography?

Let's take a look at its features

- [CARRIERS CHAINS]
  Data is split among many carriers. Only the correct carrier sequence enables unhiding. Moreover, up to 256Mb can be hidden, if you have enough carriers at disposal. Last carrier will be filled with random bits in order to make it undistinguishable from others.

- [SUPPORTED FORMATS]
  Images, audios, videos, flash, adobe.
  Supported formats in detail

- [LAYERS OF SECURITY]
  Data, before carrier injection, is encrypted (1), scrambled (2), whitened (3) and encoded (4).
  Features: program architecture

  - [LAYER 1 - MODERN MULTI-CRIPTOGRAPHY]
    A set of 16 modern 256bit open-source cryptography algorithms has been joined into a double-password multi-cryptography algorithm (256bit+256bit).

  - [LAYER 2 - CSPRNG BASED SCRAMBLING]
    Encrypted data is always scrambled to break any remaining stream pattern. A new cryptographically secure pseudo random number generator (CSPRNG) is seeded with a third password (256bit) and data is globally shuffled with random indexes.

  - [LAYER 3 - CSPRNG BASED WHITENING]
    Scrambled data is always mixed with a high amount of noise, taken from an independent CSPRNG seeded with hardware entropy.
    Options: Bits selection level

  - [LAYER 4 - ADAPTIVE NON-LINEAR ENCODING]
    Whitened data is always encoded using a non-linear function that takes also original carrier bits as input. Modified carriers will need much less change and deceive many steganalysis tests (e.g.: $\chi^2$ test).
    Features: adaptive encoding and steganalysis resistance

  - [EXTRA SECURITY - DENIABLE STEGANOGRAPHY]
    Top secret data can be protected using less secret data as a decoy.
    What is deniable steganography?

- [SOURCE CODE]
  This program relies on the libObfuscate system-independent open-source library. Users and developers are absolutely free to link to the core library (100% of the cryptography & obfuscation code), read it and modify it.

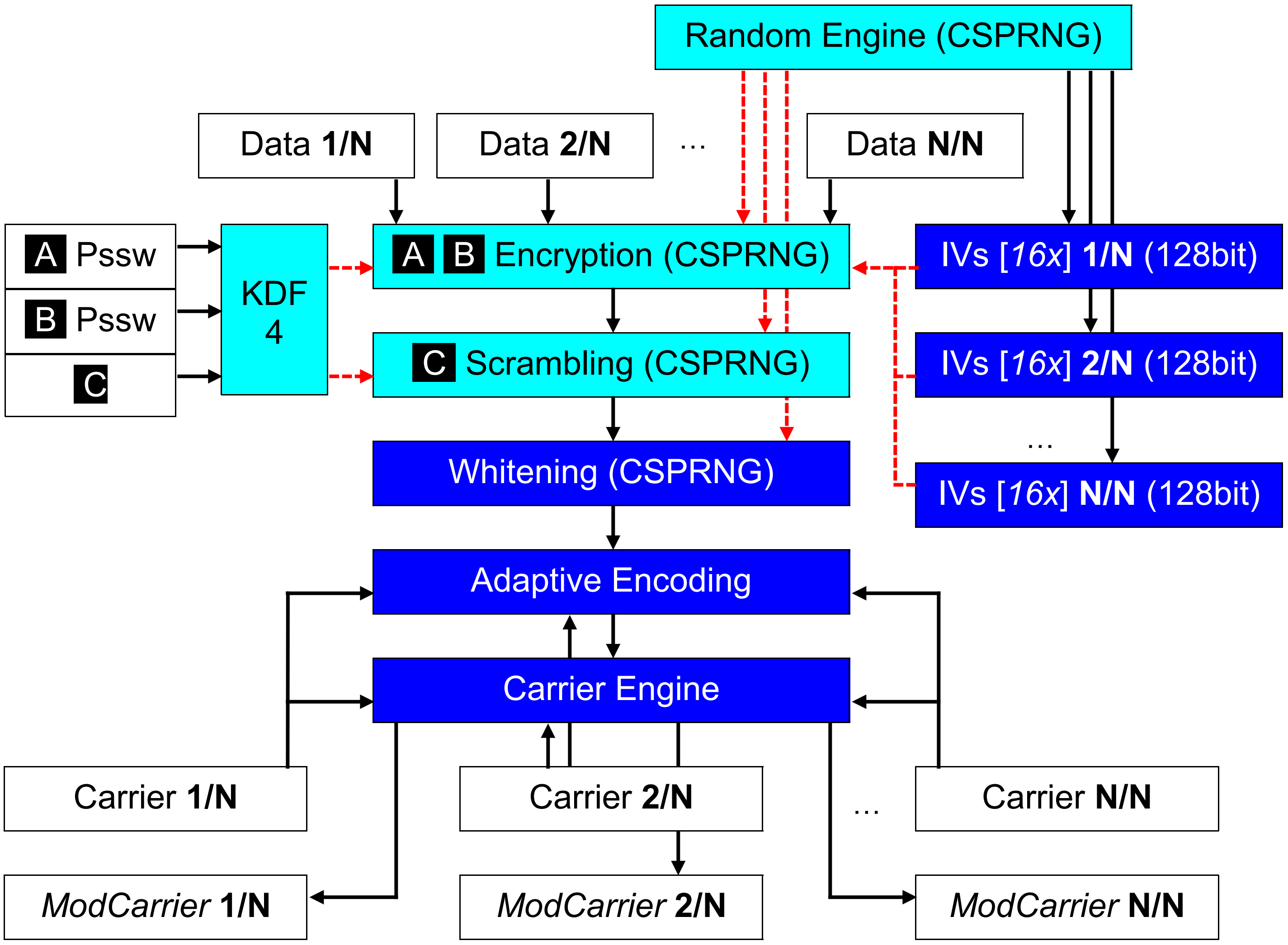
  *You're kindly asked to send any libObfuscate porting/upgrade/customizing/derived sw, in order to analyze them and add them to the project homepage. A central updated official repository will avoid sparseness and unreachability of the project derived code.*

Back

## FEATURES: PROGRAM ARCHITECTURE

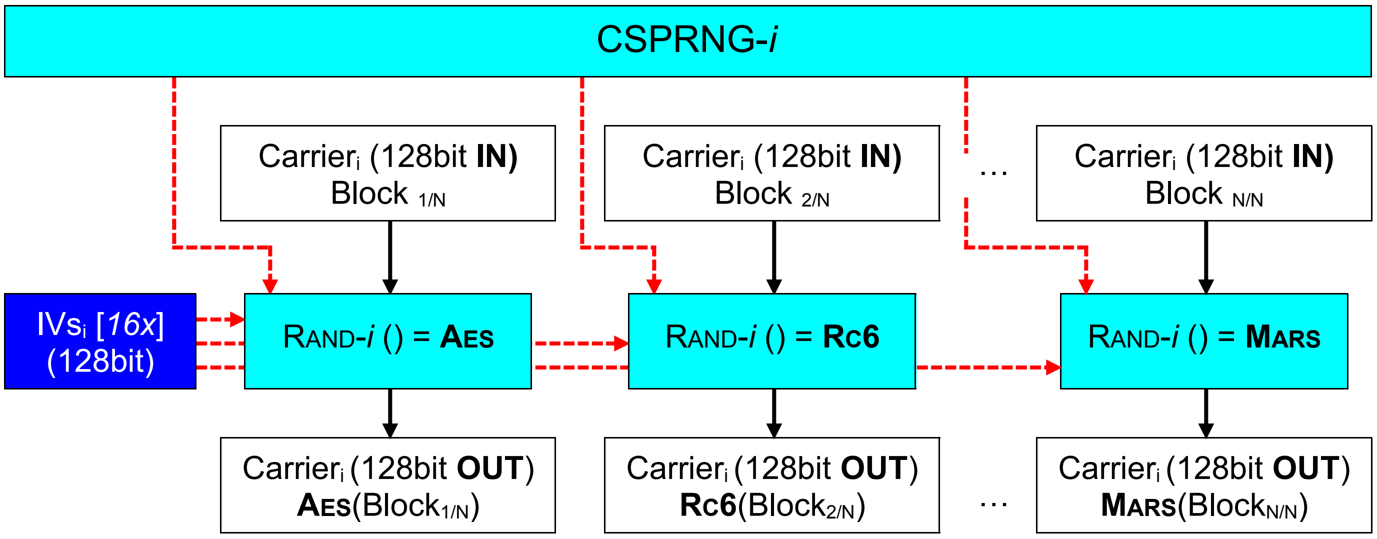
A high-level global description of OpenPuff's architecture

- data is split among carriers
- each carrier is associated to a random initialization vector array (IVs)
- text passwords (32 characters = 256bit) are associated (KDF4) to hexadecimal passwords
- data is first encrypted with two 256bit KEYS (**A**) (**B**), using multi-cryptography
- encrypted data is then scrambled, with a third key (**C**), to break any remaining stream pattern
- scrambled data is then whitened (= mixed with random noise)
- whitened data is then encoded using a function that takes also original carrier bits as input
- modified carriers receive the processed stream



Cryptography is a multi step process

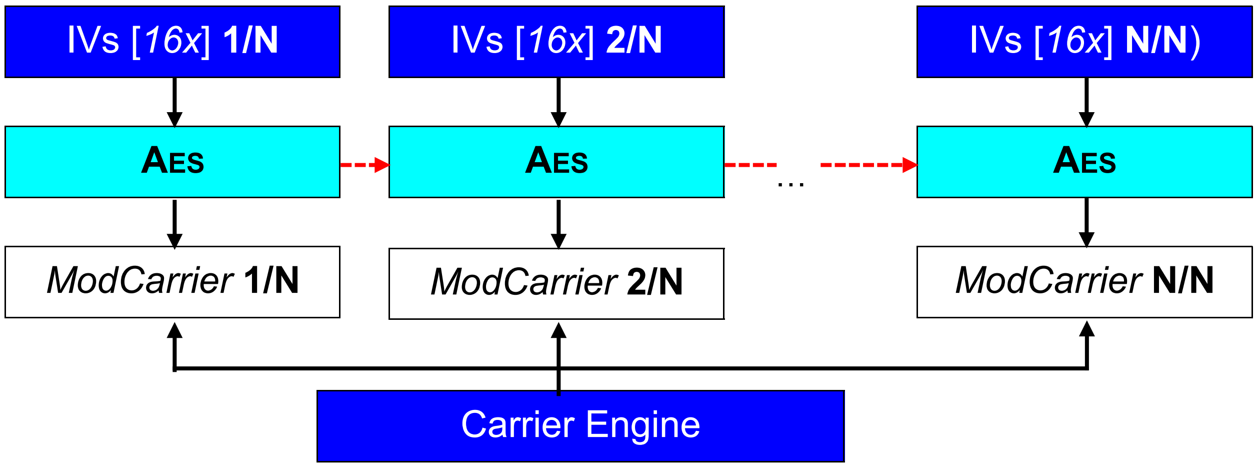
- each carrier gets an independent setup
  $CarrierSetup_i = \{ IVs_i , CSPRNG_i , Keys_i \}$
- each cipher gets an independent setup
  $Cipher_j = \{ IV_j , Key_j \}$
- each data block is processed with a different cipher, selected using the CSPRNG
  $Carrier_i\ CryptedBlock_k = r \leftarrow Rand\text{-}i\ () ;\ Cipher_r ( IV_r , Key_r , Carrier_i\ Block_k )$



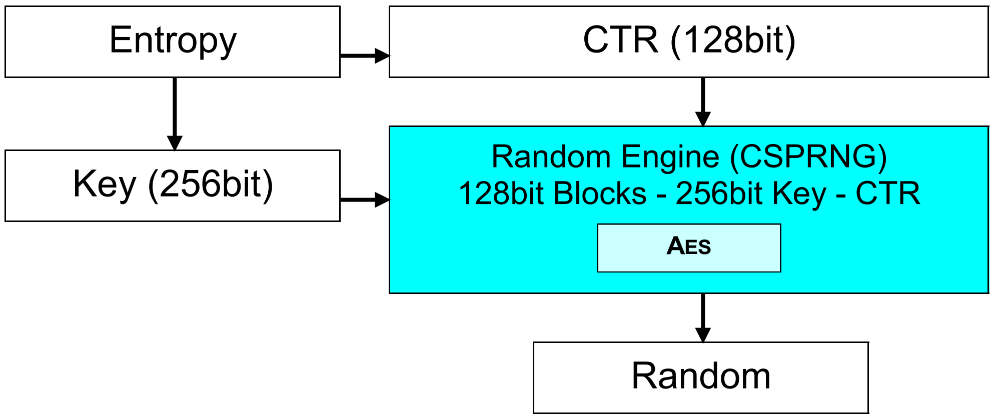
CSPRNG-i
Carrieri (128bit IN) Block 1/N
Carrieri (128bit IN) Block 2/N
Carrieri (128bit IN) Block N/N
IVsi [16x] (128bit)
RAND-i () = AES
RAND-i () = RC6
RAND-i () = MARS
Carrieri (128bit OUT) AES(Block1/N)
Carrieri (128bit OUT) RC6(Block2/N)
Carrieri (128bit OUT) MARS(BlockN/N)

Modified carriers receive

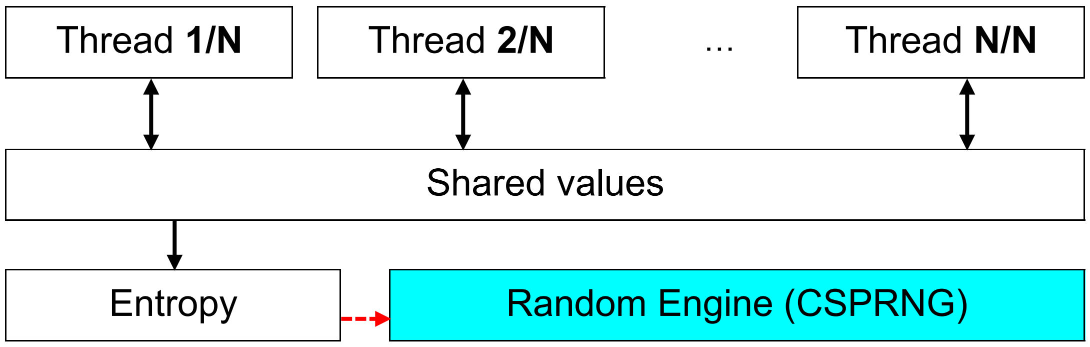
- an encrypted copy of (AES) its initialization vector array
  $CryptedIVs_n = Crypt ( IVs_n , CryptedIVs_{n-1} )$
- processed data



OpenPuff implements a cryptographically secure pseudo random number generator (CSPRNG) using AES-256 encryption. Block-based secure algorithms running in Counter-Mode (CTR) behave, by construction, as a random engine.



A good hardware source of starting entropy has been provided, not depending on any third-party library or system-API. Threads are always scheduled by the OS in an unpredictable sequence (due to an unavoidable lack of timing accuracy), easily allowing to get a significant amount of EXECUTION RACE CONDITION. ***N*** threads run in parallel, incrementing and decrementing shared values that, after a while, turn into random values.



Testing has been performed on the statistical resistance of the CSPRNG and the multi-wrapper, using the well known PSEUDORANDOM NUMBER SEQUENCE TEST PROGRAM - ENT.

Provided results are taken from 64Kb, 128Kb, ... 256Mb samples:

- bit entropy test resistance:

| | |
|---|---|
| >7.9999xx / 8.000000 | *reference: >7.9* |

- compression test resistance (size reduction after compression):

| | |
|---|---|
| 0% | *reference: <1%* |

- chi-squared distribution test resistance:

| | |
|---|---|
| 20% < deviazione < 80% | *reference: >10%, <90%* |

- mean value test resistance:

| | |
|---|---|
| 127.4x / 127.5 | *reference: >127, <128* |

- Monte Carlo test resistance:

| | |
|---|---|
| errore < 0.01% | *reference: < 1%* |

- serial correlation test resistance:

| | |
|---|---|
| < 0.0001 | *reference: < 0.01* |

BACK

## FEATURES: ADAPTIVE ENCODING AND STEGANALYSIS RESISTANCE

Security, performance and steganalysis resistance are conflicting trade-offs.

[Security vs. Performance]: Whitening
- Pro: ensures higher data security
- Pro: allows deniable steganography
- **Con1**: *requires a lot of extra carrier bits*

[Security vs. Steganalysis]: Cryptography + Whitening
- Pro: ensure higher data security
- **Con2**: *their random-like statistical response marks carriers as more "suspicious"*

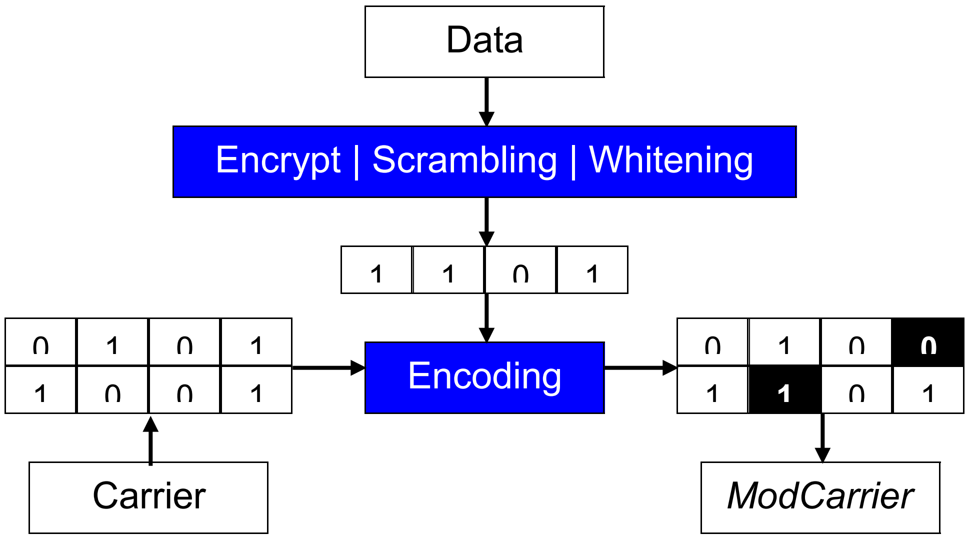
Should we then be concerned about OpenPuff's STEGANALYSIS RESISTANCE? Data, before carrier injection, is encrypted (1), scrambled (2) and whitened (3). Do these 3 steps turn a small amount of hidden data into a big chunk of suspicious data?

A new security layer has been added at the bottom of the data process. Whitened data is always encoded using a non-linear function that takes also original carrier bits as input. Modified carriers will need much less change (**Con1**) and, lowering their random-like statistical response, deceive many steganalysis tests (**Con2**).

"DEFENDING AGAINST STATISTICAL STEGANALYSIS" (Niels Provos)

"CONSTRUCTING GOOD COVERING CODES FOR APPLICATIONS IN STEGANOGRAPHY" (Jessica Fridrich)

The provided coding implementation is a novel unpublished function (built from scratch) that ensures
- output password dependence
- high (50%) embedding efficiency
- low (<20%) change rate



BACK

## Features: multi-cryptography & data obfuscation

### FAQ 1: Why didn't you simply implement a standard AES-256 or RSA-1024?

Modern open-source cryptography

- has been thoroughly investigated and reviewed by the scientific community
- it's widely accepted as the safest way to secure your data
- fulfills almost every *standard* need of security

OpenPuff doesn't support any Conspiracy theory against our privacy (Secret cracking backdoors, intentionally weak cryptography designs, …). There's really no reason not to trust standard modern publicly available cryptography (although some old ciphers have been already Cracked).

Steganography users, however, are very likely to be hiding very sensitive data, with an *unusually high* need of security. Their secrets need to go through a deep process of data Obfuscation in order to be able to *longer* survive forensic investigation and hardware aided brute force attacks.

### FAQ 2: Is multi-cryptography similar to multiple-encryption?

Multi-cryptography is something really different from Multiple-encryption (encrypting more than once). There's really no common agreement about multiple-encryption's reliability. It's thought to be:

- better than single encryption
- weak as the weakest cipher in the encryption queue/process
- worse than single encryption

OpenPuff supports the last thesis (worse) and never encrypts already encrypted data.

### FAQ 3: Is multi-cryptography similar to random/polymorpihc-cryptography?

Random-cryptography, a.k.a. polymorphic cryptography, is a well-known Snake-oil cryptography. Multi-cryptography is something completely different and never aims to build some better, random or on-the-fly cipher.

OpenPuff only relies on stable modern open-source cryptography.

Features: program architecture

Back

## What is steganography?

It's a Smart way to hide data into other files, called **carriers**. Modified carriers will look like the original ones, without perceptible changes. Best carriers are videos, images and audio files, since everybody can send/receive/download them without suspects.

The steganography process is highly selective and adaptive, with a minimum payload for each carrier. Carriers with a maximum hidden data amount less than the minimum payload will be discarded.

| | | |
|---|---|---|
| +256B | → | IV |
| +16B | → | a cryptography block |

Features: program architecture

There's no CARRIER bytes threshold during the marking process.
What is marking?

### Why should I need this technique?

You **don't need** this technique if your data

- can be explicitly sent or stored in altered suspicious format.

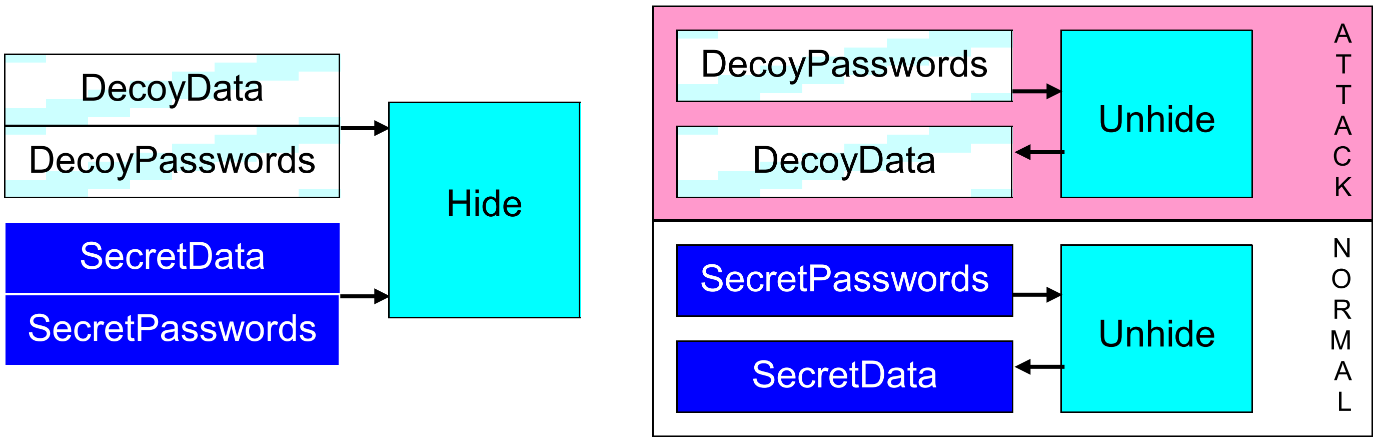
You **may be** interested in this technique if your data

- needs hiding without turning into suspicious format.
- have to be easily accessible by everyone, but retrievable only by people acquainted with your secret intent.

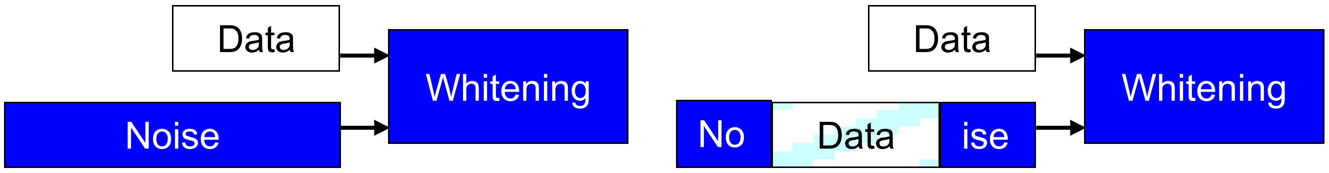
Back

## What is deniable steganography?

Deniable encryption/steganography is a decoy based technique that allows you to convincingly deny the fact that you're hiding **sensitive data**, even if attackers are able to state that you're hiding some data. You only have to provide some expendable decoy data that you would **plausibly** want to keep confidential. It will be revealed to the attacker, claiming that this is all there is.



How is it possible? Encrypted and scrambled data, before carrier injection, is whitened (Features: program architecture) with a high amount of noise (Options: bits selection level). Decoy data can replace some of this noise without loosing final properties of Cryptanalysis resistance.

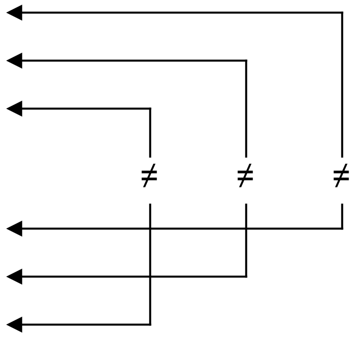


Sensitive data and decoy data are encrypted using different passwords. You have to choose two different sets of different passwords.

Example:

| | | |
|---|---|---|
| Sensibile data: | Password (A) | "*FirstDataPssw1*" |
| | Password (B) | "*SecondDataPssw2*" |
| | Password (C) | "*AnotherDataPssw3*" |

(A ∩ B) 70%, (A ∩ C) 67%, (B ∩ C) 68%, Hamming distance ≥ 25%



| | | |
|---|---|---|
| Decoy data: | Password (A') | "*FirstDecoyPssw1*" |
| | Password (B') | "*SecondDecoyPssw2*" |
| | Password (C') | "*AnotherDecoyPssw3*" |

(A' ∩ B') 72%, (A' ∩ C') 60%, (B' ∩ C') 70%, Hamming distance ≥ 25%

Each password has to be different (at bit level) and at least 8 characters long.

Example: “DataPssw1” (A) “DataPssw2” (B) “DataPssw3” (C)

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| (A) | 01000100 | 01100001 | 01110100 | 01100001 | 01010000 | 01110011 | 01110011 | 01110111 | 001100**01** | … |
| (B) | 01000100 | 01100001 | 01110100 | 01100001 | 01010000 | 01110011 | 01110011 | 01110111 | 001100**10** | … |
| (C) | 01000100 | 01100001 | 01110100 | 01100001 | 01010000 | 01110011 | 01110011 | 01110111 | 001100**11** | … |

(A ∩ B) 98%, (A ∩ C) 99%, (B ∩ C) 99%, Hamming distance < 25% = **KO**

Example: “FirstDataPssw1” (A) “SecondDataPssw2” (B) “AnotherDataPssw3” (C)

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| (A) | 0100**0110** | 0110**1001** | 011**10010** | 011**10011** | 011**10100** | 01**0**00100 | 01**100001** | 01**110100** | 01**100001** | … |
| (B) | 010**10011** | 0110**0101** | 011**00011** | 011**01111** | 011**01110** | 01**1**00100 | 01**000100** | 01**100001** | 01**110100** | … |
| (C) | 010**00001** | 0110**1110** | 011**01111** | 011**10100** | 011**01000** | 01**1**00101 | 01**110010** | 01**000100** | 01**100001** | … |

(A ∩ B) 70%, (A ∩ C) 67%, (B ∩ C) 68%, Hamming distance ≥ 25% = **OK**

You will be asked for

- two **different** sets of different passwords
- a stream of sensitive data
- a stream of decoy data **compatible** (by size) with sensitive data
  $\sum_{k \in \{1, N-1\}} used\_carrier\_bytes( carr_k ) < Sizeof( Decoy ) \le \sum_{k \in \{1, N\}} used\_carrier\_bytes( carr_k )$

Example:

| Carriers | Carrier bytes | SensibleData | DecoyData |
|---|---|---|---|
| +Carr (1/N) | 32 | X | Used |
| … | 2688 | X | Used |
| +Carr (N-1/N) | 48 | X | Used |
| **+Carr (N/N)** | **64** | | **Not used** |
| | Total = 2832 | Total = 2795 | 2720 < Size ≤ 2768 |

BACK

## WHAT IS MARKING?

Marking is here stated as the action of signing a file with your copyright mark (best known as WATERMARKING). This program does it in a steganographic way, applied to videos, images and audio files. Your copyright mark will be invisible, but accessible by everyone (using this program), since it won't be password protected.

### WHY SHOULD I NEED THIS TECHNIQUE?

You **don't need** this technique if your copyright mark
- needs to be clearly visible
- has to be independent from graphic/audio data, therefore capable of surviving editing operations

You **may be** interested in this technique if your copyright mark
- needs to be invisible
- has to be dependent from graphic/audio data, therefore incapable of surviving editing operations
- has to be accessible by everyone (using this program)

A possible usage of this technique could be: insertion of an invisible copyright mark into registered files that have to be publicly shared. Illegally manipulated copies will maybe look like original ones, but will partially/totally loose the invisible copyright mark.

BACK

## SUPPORTED FORMATS IN DETAIL

| | |
|---|---|
| Images: | BMP, JPG, PCX, PNG, TGA |
| Audios: | AIFF, MP3, NEXT/SUN, WAV |
| Videos: | 3GP, FLV, MP4, MPG, SWF, VOB |
| Flash-Adobe: | PDF |

Carriers will keep their format

- [in: 32 bits per plane TGA, out: 32 bits per plane TGA]
- [in: Stereo WAV, out: Stereo WAV]
- [in: RGB+Alpha BMP, out: RGB+Alpha BMP]

etc…

Additional tags/chunks and extra bytes will be entirely copied unchanged.
Don't perform any further operation to modified carriers. Their carrier bits would very probably be altered.

BACK

## BMP Images (Microsoft)

- Known extensions: **.BMP, *.DIB*
- 24/32 bits per pixel
- Mono/RGB/RGB+Alpha
- Up to version 5

Back

## JPG Images (Joint Photographic Experts Group)

- Known extensions: **.JPG, *.JPE, *.JPEG, *.JFIF*
- 8 bits per plane
- 1-4 planes per pixel, i.e.: Mono/RGB/YCbCr/YCbCrK/CMY/CMYK
- Baseline lossy DCT-jfif with Huffman compression
- h2v2 (4:4), h1v2 (4:2), h2v1 (4:2), h1v1 (4:1) planes independent alignment

Back

## PCX Images (ZSoft)

- Known extensions: **.PCX*
- 24 bits per pixel Mono/RGB
- Compressed/Uncompressed

Back

## PNG Images (Portable Network Graphics)

- Known extensions: **.PNG*
- 8/16 bits per plan
- Mono/RGB/Mono+Alpha/RGB+Alpha
- Interlaced/Linear

Back

## TGA Images (Targa Truevision)

- Known extensions: **.TGA, *.VDA, *.ICB, *.VST*
- Mono-8 bits per pixel or RGB/RGB+Alpha-24/32 bits per pixel
- Compressed/Uncompressed

Back

## AIFF Audios (Audio Interchange File Format)

- Known extensions: **.AIF, *.AIFF*
- 16 bits per sample
- Mono/Stereo/Multi channels
- Linear, uncompressed

Back

## MP3 Audios (Fraunhofer Institut)

- Known extensions: **.MP3*
- MPG 1/MPG 2/MPG 2.5 Layer III
- Fixed/Variable Bitrate
- Mono/Dual Channel/Joint Stereo/Stereo
- ID Tagged

Back

## NEXT/SUN Audios (Sun & Next)

- Known extensions: **.AU, *.SND*
- 16 bits per sample
- Mono/Stereo/Multi channels
- Linear, uncompressed

Back

## WAV Audios (Microsoft)

- Known extensions: **.WAV, *.WAVE*
- 16 bits per sample
- Mono/Stereo/Multi channels
- PCM, uncompressed

Back

## 3GP Videos (3rd Generation Partnership Program)

- Known extensions: **.3GP, *.3GPP, *.3G2, *.3GP2*
- Up to version 10
- Codec independent support
- Up to 32 tracks

Back

## Adobe FLV Videos (Flash Video)

- Known extensions: **.FLV*, **.F4V*, **.F4P*, **.F4A*, **.F4B*
- Up to version 10
- Codec independent support
- Audio MP3 tracks analysis

Back

## MP4 Videos (Motion Picture Experts Group)

- Known extensions: **.MP4*, **.MPG4*, **.MPEG4*, **.M4A*, **.M4V*, **.MP4A*, **.MP4V*
- Up to specification ISO/IEC 14496-12:2008
- Codec independent support
- Up to 32 tracks

Back

## MPG Videos (Motion Picture Experts Group)

- Known extensions: **.MPG*, **.MPEG*, **.MPA*, **.MPV*, **.MP1*, **.MPG1*, **.M1A*, **.M1V*, **.MP1A*, **.MP1V*, **.MP2*, **.MPG2*, **.M2A*, **.M2V*, **.MP2A*, **.MP2V*
- Mpeg I Systems - up to specification ISO/IEC 11172-1:1999
- Mpeg II Systems - up to specification ISO/IEC 13818-1:2007
- Codec independent support

Back

## Adobe SWF Videos (Shockwave Flash)

- Known extensions: **.SWF*
- Up to version 10
- Codec independent support
- Audio MP3 tracks analysis

Back

## VOB Videos (DVD - Video Object)

- Known extensions: **.VOB*
- Mpeg II Systems - up to specification ISO/IEC 13818-1:2007
- Codec independent support

Back

## Adobe PDF files (Portable Document Format)

- Known extensions: **.PDF*
- Up to specification ISO/IEC 32000-1:2008
- Revision independent support

Back

## <u>Suggestions for better results</u>

### Carrier chains:

Hide your data into single/multiple carrier chains, adding carriers in unexpected order. Unhiding attempts by unallowed curious people will grow in complexity.

Single carrier example: (Simple, Fast unhiding time, Unsafe)
*+MyData >> John.mp3*

Single chain example: (Medium complexity, Medium unhiding time, Safe)
*+MyData >> Bear.jpg | Zoo.tga | Arrow.png | John.bmp | …*

Multiple chains example: (Maximum complexity, Slow unhiding time, Safer)
*+MyData (1/n) >> Bear.jpg | Arrow.png | John.bmp | …*
*…*
*+MyData (n/n) >> Zoo.tga | Arrow.png | Beep.wav | …*

### Password:

Make use of long (>16 chars) easy to remember passwords, changing them every time.

### Carrier bits selection level:

Make always use of different levels for each hiding process. Unhiding attempts by unallowed curious people will grow in complexity.

Example:
*Hiding process 1:*
- *Aiff: Low*
- *BMP: Very low*
- *JPG: Maximum*

*…*
*Hiding process 2:*
- *AIFF: Medium*
- *BMP: Low*
- *JPG: Minimum*

*…*

## A Full system could be…

- Hiding your data into many complex chains (hundreds of carriers, with non alphabetical random order), each one with a completely different set of 32-chars-passwords
- Saving all settings inside an “index” single carrier

Example:

| | |
|---|---|
| *+MyData (1/n)* | *[carrier1 \| … \| carrier100]*<br>*[VeryLongPasswords1]*<br>*[BitsSelectionLevel1]* |

…

| | |
|---|---|
| *+MyData (n/n)* | *[carrier1 \| … \| carrier100]*<br>*[VeryLongPasswordsN]*<br>*[BitsSelectionLevelN]* |

A very unsuspicious “index” carrier (fixed password + fixed bits selection level) holding a text file that summaries

- carriers name and order
- passwords
- bit selection levels

Back

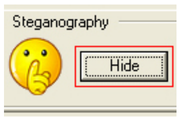
## Options: bits selection level

| | |
|---|---|
| (*Minimum*) | 1/8 data, 7/8 whitening. |
| (*Very Low*) | 1/7 data, 6/7 whitening. |
| (*Low*) | 1/6 data, 5/6 whitening. |
| (*Medium*) | 1/5 data, 4/5 whitening. |
| (*High*) | 1/4 data, 3/4 whitening. |
| (*Very High*) | 1/3 data, 2/3 whitening. |
| (*Maximum*) | 1/2 data, 1/2 whitening. |

Back

## Data Hiding step by step

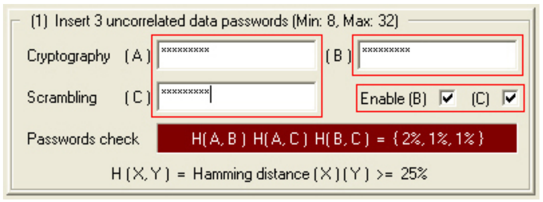
### Begin:



| | |
|---|---|
| (*Hide*) | Go to hiding panel |

Select *Hide*.

### Step 1 – Choose password(s):

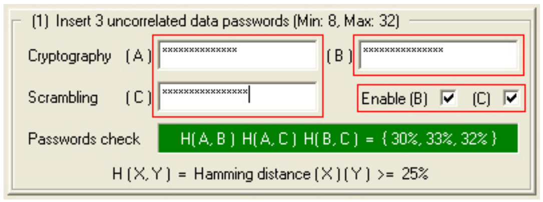

(1) Insert 3 uncorrelated data passwords (Min: 8, Max: 32)

Cryptography (A) ********* (B) *********

Scrambling (C) ********* Enable (B) ☑ (C) ☑

Passwords check H( A, B ) H( A, C ) H( B, C ) = { 2%, 1%, 1% }

H ( X, Y ) = Hamming distance ( X ) ( Y ) >= 25%


(1) Insert 3 uncorrelated data passwords (Min: 8, Max: 32)

Cryptography (A) ************** (B) **************

Scrambling (C) *************** Enable (B) ☑ (C) ☑

Passwords check H( A, B ) H( A, C ) H( B, C ) = { 30%, 33%, 32% }

H ( X, Y ) = Hamming distance ( X ) ( Y ) >= 25%

| | |
|---|---|
| (*Cryptography A*) | First password (cryptography keys) |
| (*Cryptography B*) | Second password (cryptography CSPRNG) |
| (*Scrambling C*) | Third password (scrambling CSPRNG) |
| (*Enable B*) | Second password enable/disable |
| (*Enable C*) | Third password enable/disable |

Insert three separate passwords. Each password has to be different (at bit level) and at least 8 characters long. Password type and number can be easily customized disabling the second (B) and/or the third (C) password. Disabled passwords will be set as the first (A) password.

Example: “DataPssw1” (A) “DataPssw2” (B) “DataPssw3” (C)

```
(A)  01000100  01100001  01110100  01100001  01010000  01110011  01110011  01110111  00110001  …
(B)  01000100  01100001  01110100  01100001  01010000  01110011  01110011  01110111  00110010  …
(C)  01000100  01100001  01110100  01100001  01010000  01110011  01110011  01110111  00110011  …
```

(A ∩ B) 98%, (A ∩ C) 99%, (B ∩ C) 99%, Hamming distance < 25% = **KO**

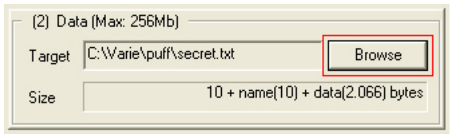
Example: “FirstDataPssw1” (A) “SecondDataPssw2” (B) “AnotherDataPssw3” (C)

```
(A)  01000110  01101001  01110010  01110011  01110100  01000100  01100001  01110100  01100001  …
(B)  01010011  01100101  01100011  01101111  01101110  01100100  01000100  01100001  01110100  …
(C)  01000001  01101110  01101111  01110100  01101000  01100101  01110010  01000100  01100001  …
```

(A ∩ B) 70%, (A ∩ C) 67%, (B ∩ C) 68%, Hamming distance ≥ 25% = **OK**

Suggestions for better results
What is deniable steganography?

## Step 2 – Choose data to be hidden:


(2) Data (Max: 256Mb)

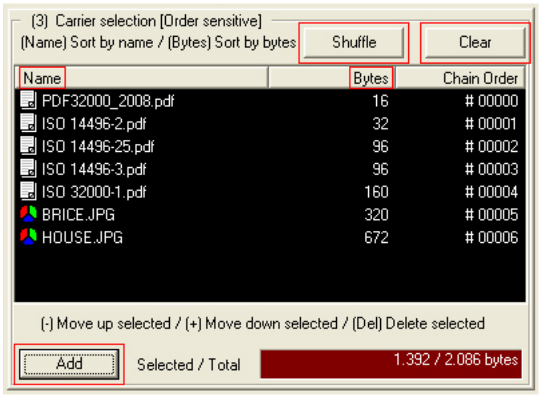
Target C:\Varie\puff\secret.txt Browse

Size 10 + name(10) + data(2.066) bytes

| | |
|---|---|
| (*Browse*) | Select a file |

Choose the secret data you want to hide (typically a zip/rar/… archive).

## Step 3 – Choose carrier(s):

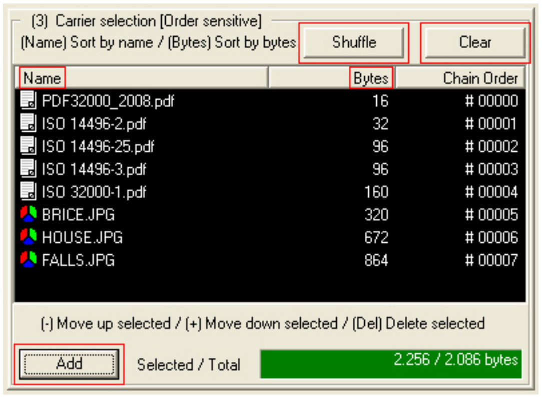

(3) Carrier selection [Order sensitive]

(Name) Sort by name / (Bytes) Sort by bytes Shuffle Clear

| Name | Bytes | Chain Order |
|---|---|---|
| PDF32000_2008.pdf | 16 | # 00000 |
| ISO 14496-2.pdf | 32 | # 00001 |
| ISO 14496-25.pdf | 96 | # 00002 |
| ISO 14496-3.pdf | 96 | # 00003 |
| ISO 32000-1.pdf | 160 | # 00004 |
| BRICE.JPG | 320 | # 00005 |
| HOUSE.JPG | 672 | # 00006 |

(-) Move up selected / (+) Move down selected / (Del) Delete selected

Add Selected / Total 1.392 / 2.086 bytes


(3) Carrier selection [Order sensitive]

(Name) Sort by name / (Bytes) Sort by bytes Shuffle Clear

| Name | Bytes | Chain Order |
|---|---|---|
| PDF32000_2008.pdf | 16 | # 00000 |
| ISO 14496-2.pdf | 32 | # 00001 |
| ISO 14496-25.pdf | 96 | # 00002 |
| ISO 14496-3.pdf | 96 | # 00003 |
| ISO 32000-1.pdf | 160 | # 00004 |
| BRICE.JPG | 320 | # 00005 |
| HOUSE.JPG | 672 | # 00006 |
| FALLS.JPG | 864 | # 00007 |

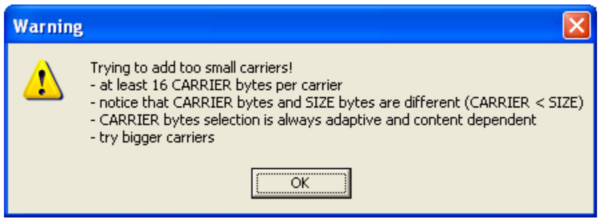
(-) Move up selected / (+) Move down selected / (Del) Delete selected

Add Selected / Total 2.256 / 2.086 bytes

| | |
|---|---|
| (*Shuffle*) | Random shuffle all carriers |
| (*Clear*) | Discard all carriers |
| (*Add*) | Add new carriers to the list |
| (*Name*)/ (*Bits*) | Sort carriers by name/bits |
| (+)/(-) | Move selected carriers up/down |
| (*Del*) | Delete selected carriers |

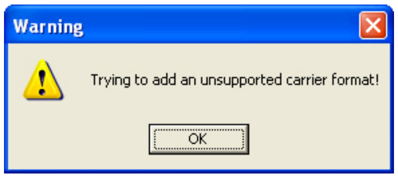
Until *selected bytes* < *total bytes* try

- adding new carriers
- increasing bit selection level

(I) 
Warning

Trying to add too small carriers!
- at least 16 CARRIER bytes per carrier
- notice that CARRIER bytes and SIZE bytes are different (CARRIER < SIZE)
- CARRIER bytes selection is always adaptive and content dependent
- try bigger carriers

OK

(II) 
Warning

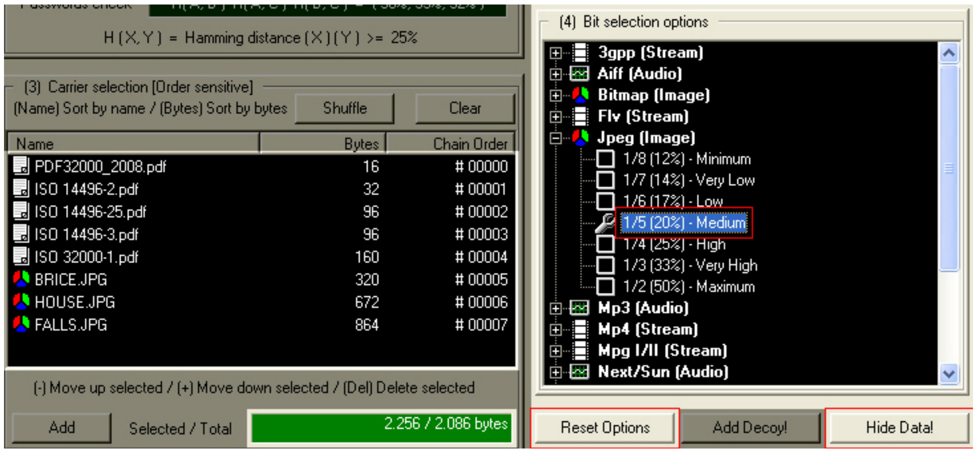
Trying to add an unsupported carrier format!

OK

Some carriers will not be added because of steganography-process constraints

- (I) not enough carrier bytes (carrier bytes $\subset$ carrier size)
  What is steganography?
- (II) unsupported format
  Supported formats in detail

## Step 4 – Choose bits selection level:



H ( X, Y ) = Hamming distance ( X ) ( Y ) >= 25%

(3) Carrier selection [Order sensitive]
(Name) Sort by name / (Bytes) Sort by bytes — Shuffle — Clear

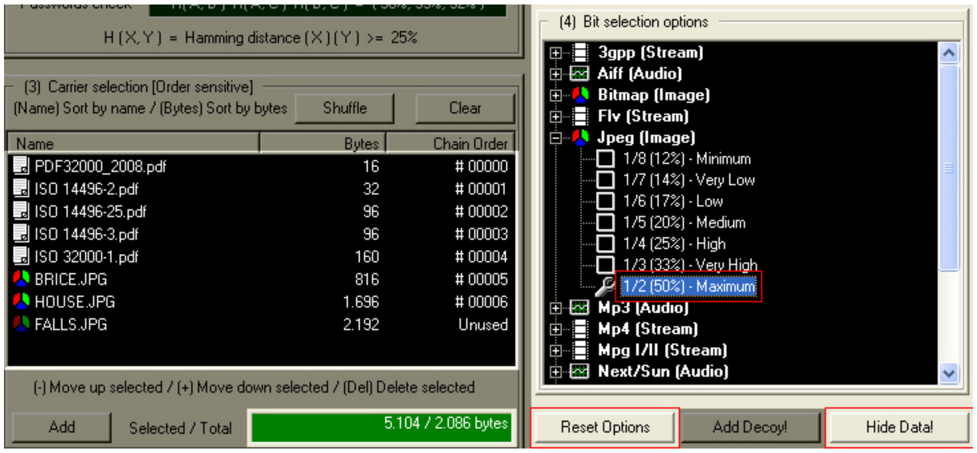
| Name | Bytes | Chain Order |
|---|---|---|
| PDF32000_2008.pdf | 16 | # 00000 |
| ISO 14496-2.pdf | 32 | # 00001 |
| ISO 14496-25.pdf | 96 | # 00002 |
| ISO 14496-3.pdf | 96 | # 00003 |
| ISO 32000-1.pdf | 160 | # 00004 |
| BRICE.JPG | 320 | # 00005 |
| HOUSE.JPG | 672 | # 00006 |
| FALLS.JPG | 864 | # 00007 |

(-) Move up selected / (+) Move down selected / (Del) Delete selected

Add — Selected / Total — 2.256 / 2.086 bytes

(4) Bit selection options
- 3gpp (Stream)
- Aiff (Audio)
- Bitmap (Image)
- Flv (Stream)
- Jpeg (Image)
  - 1/8 (12%) - Minimum
  - 1/7 (14%) - Very Low
  - 1/6 (17%) - Low
  - 1/5 (20%) - Medium
  - 1/4 (25%) - High
  - 1/3 (33%) - Very High
  - 1/2 (50%) - Maximum
- Mp3 (Audio)
- Mp4 (Stream)
- Mpg I/II (Stream)
- Next/Sun (Audio)

Reset Options — Add Decoy! — Hide Data!



H ( X, Y ) = Hamming distance ( X ) ( Y ) >= 25%

(3) Carrier selection [Order sensitive]
(Name) Sort by name / (Bytes) Sort by bytes — Shuffle — Clear

| Name | Bytes | Chain Order |
|---|---|---|
| PDF32000_2008.pdf | 16 | # 00000 |
| ISO 14496-2.pdf | 32 | # 00001 |
| ISO 14496-25.pdf | 96 | # 00002 |
| ISO 14496-3.pdf | 96 | # 00003 |
| ISO 32000-1.pdf | 160 | # 00004 |
| BRICE.JPG | 816 | # 00005 |
| HOUSE.JPG | 1.696 | # 00006 |
| FALLS.JPG | 2.192 | Unused |

(-) Move up selected / (+) Move down selected / (Del) Delete selected

Add — Selected / Total — 5.104 / 2.086 bytes

(4) Bit selection options
- 3gpp (Stream)
- Aiff (Audio)
- Bitmap (Image)
- Flv (Stream)
- Jpeg (Image)
  - 1/8 (12%) - Minimum
  - 1/7 (14%) - Very Low
  - 1/6 (17%) - Low
  - 1/5 (20%) - Medium
  - 1/4 (25%) - High
  - 1/3 (33%) - Very High
  - 1/2 (50%) - Maximum
- Mp3 (Audio)
- Mp4 (Stream)
- Mpg I/II (Stream)
- Next/Sun (Audio)

Reset Options — Add Decoy! — Hide Data!

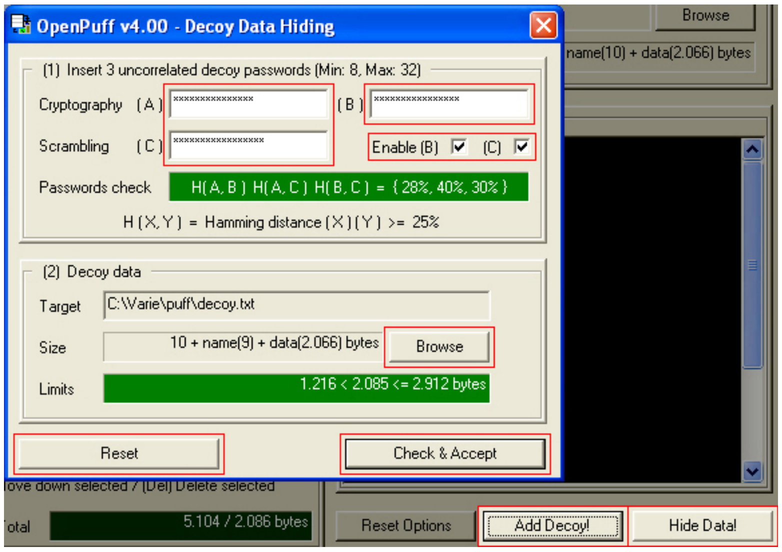
| | |
|---|---|
| (*Reset Options*) | Reset all bits selection level to normal |
| (*Add Decoy!*) | Add a decoy (deniable steganography) |
| (*Hide!*) | Start hiding |

After
- typing twice the same password, at least 8 chars
- selecting a non-empty file to hide
- adding enough carrier bits
- adding a decoy (optional)

start the hiding task

Options: bits selection level

## Step 5 (optional) – Choose password(s) & decoy data:



OpenPuff v4.00 - Decoy Data Hiding

(1) Insert 3 uncorrelated decoy passwords (Min: 8, Max: 32)

Cryptography (A) *************** (B) ***************

Scrambling (C) ****************

Enable (B) ✓ (C) ✓

Passwords check H(A, B) H(A, C) H(B, C) = {28%, 40%, 30%}

H(X, Y) = Hamming distance (X)(Y) >= 25%

(2) Decoy data

Target C:\Varie\puff\decoy.txt

Size 10 + name(9) + data(2.066) bytes

Limits 1.216 < 2.085 <= 2.912 bytes

name(10) + data(2.066) bytes

ove down selected / [Del] Delete selected

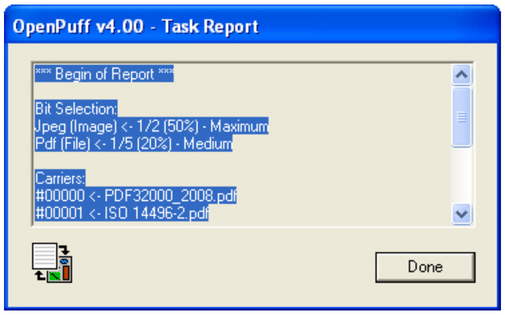
otal 5.104 / 2.086 bytes

| | |
|---|---|
| (*Cryptography A*) | First password (cryptography keys) |
| (*Cryptography B*) | Second password (cryptography CSPRNG) |
| (*Scrambling C*) | Third password (scrambling CSPRNG) |
| (*Enable B*) | Second password enable/disable |
| (*Enable C*) | Third password enable/disable |
| (*Browse*) | Select a file |
| (*Reset*) | Reset password and file |
| (*Check & Accept*) | Check password correlation and file size |

- decoy passwords have to be each other **different**, and different from data passwords
- decoy password type and number can be customized like data passwords
- decoy data has to be **compatible** (by size) with sensitive data
  $\sum_{k \in \{1, N-1\}} used\_carrier\_bytes( carr_k ) < Sizeof( Decoy ) \leq \sum_{k \in \{1, N\}} used\_carrier\_bytes( carr_k )$

What is deniable steganography?

## Task report:



OpenPuff v4.00 - Task Report

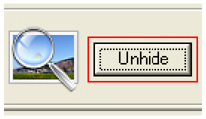
```
*** Begin of Report ***

Bit Selection:
Jpeg (Image) <- 1/2 (50%) - Maximum
Pdf (File) <- 1/5 (20%) - Medium

Carriers:
#00000 <- PDF32000_2008.pdf
#00001 <- ISO 14496-2.pdf
```

End report summarizes all information needed for a successful unhiding.

Back

## DATA UNHIDING STEP BY STEP

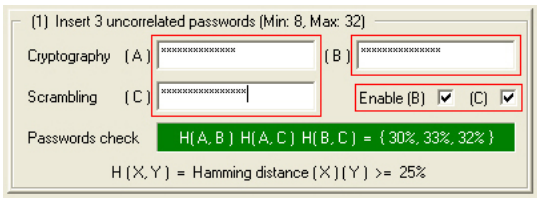
**BEGIN:**



| | |
|---|---|
| (*Unhide*) | Go to unhiding panel |

Select *Unhide*.

**STEP 1 – CHOOSE PASSWORD(S):**



(1) Insert 3 uncorrelated passwords (Min: 8, Max: 32)

Cryptography (A) ************** (B) ***************

Scrambling (C) ***************

Enable (B) ✓ (C) ✓

Passwords check H(A,B) H(A,C) H(B,C) = {30%, 33%, 32%}

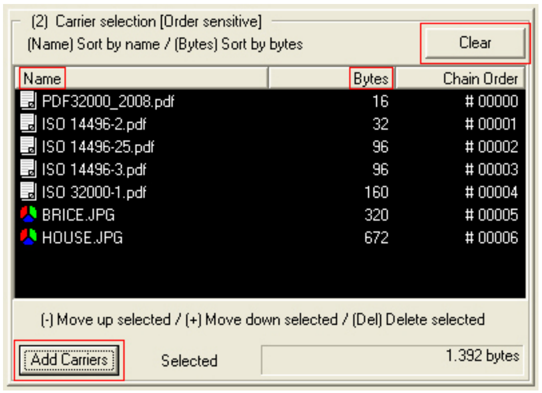
H(X,Y) = Hamming distance (X)(Y) >= 25%

| | |
|---|---|
| (*Cryptography A*) | First password (cryptography keys) |
| (*Cryptography B*) | Second password (cryptography CSPRNG) |
| (*Scrambling C*) | Third password (scrambling CSPRNG) |
| (*Enable B*) | Second password enable/disable |
| (*Enable C*) | Third password enable/disable |

Insert your passwords (secret to get secret data, decoy to get decoy data), enabling only those used at hiding time.

SUGGESTIONS FOR BETTER RESULTS
WHAT IS DENIABLE STEGANOGRAPHY?

**STEP 2 – CHOOSE CARRIER(S):**



(2) Carrier selection [Order sensitive]
(Name) Sort by name / (Bytes) Sort by bytes

Clear

| Name | Bytes | Chain Order |
|---|---|---|
| PDF32000_2008.pdf | 16 | # 00000 |
| ISO 14496-2.pdf | 32 | # 00001 |
| ISO 14496-25.pdf | 96 | # 00002 |
| ISO 14496-3.pdf | 96 | # 00003 |
| ISO 32000-1.pdf | 160 | # 00004 |
| BRICE.JPG | 320 | # 00005 |
| HOUSE.JPG | 672 | # 00006 |

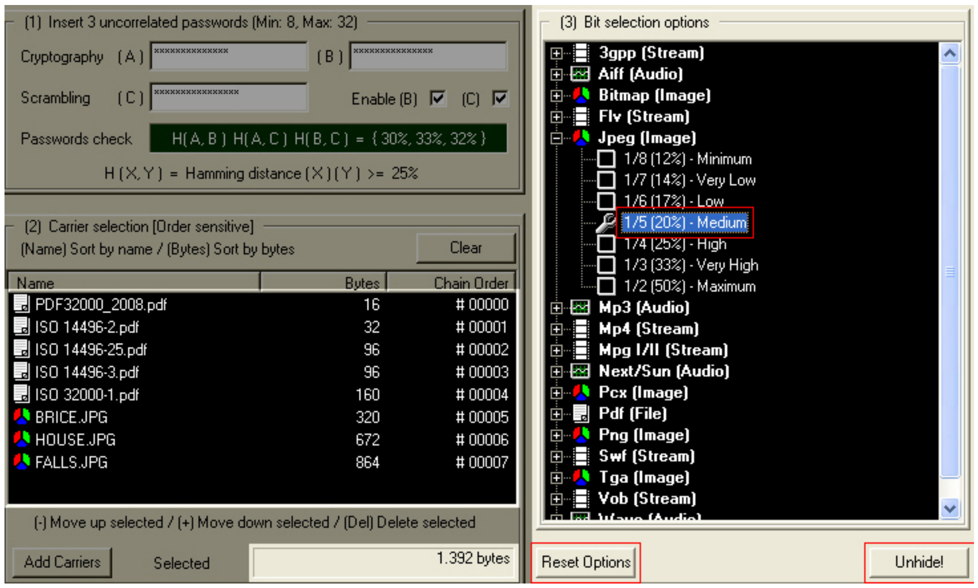
(-) Move up selected / (+) Move down selected / (Del) Delete selected

Add Carriers    Selected    1.392 bytes

| | |
|---|---|
| (*Clear*) | Discard all carriers |
| (*Add*) | Add new carriers to the list |
| (*Name*)/ (*Bits*) | Sort carriers by name/bits |
| (+)/(-) | Move selected carriers up/down |
| (*Del*) | Delete selected carriers |

Add all the carriers that have been processed during the hide task.
SUPPORTED FORMATS IN DETAIL

## Step 3 – Choose bits selection level:



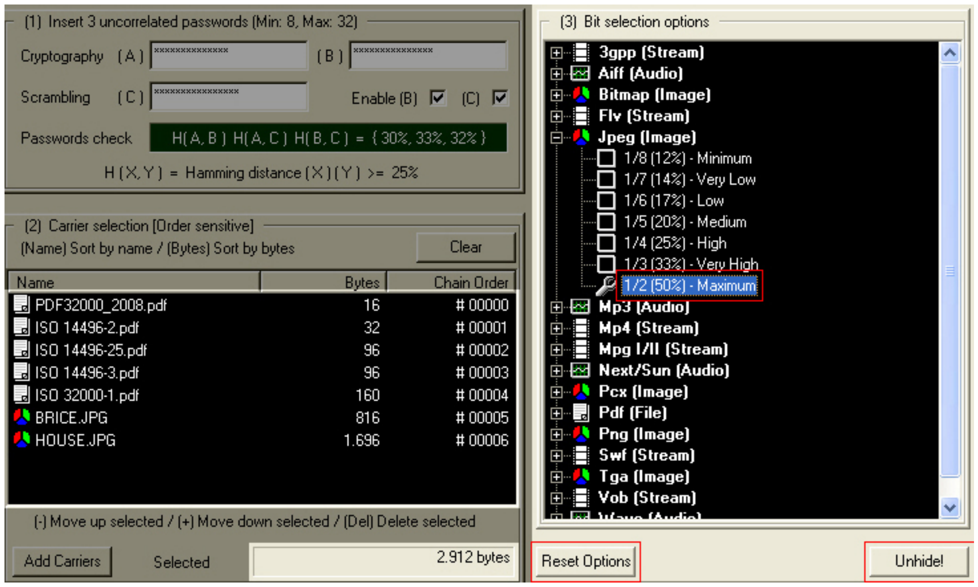
(1) Insert 3 uncorrelated passwords (Min: 8, Max: 32)
Cryptography (A) ************** (B) ***************
Scrambling (C) ***************  Enable (B) ☑ (C) ☑
Passwords check H(A,B) H(A,C) H(B,C) = (30%, 33%, 32%)
H(X,Y) = Hamming distance (X)(Y) >= 25%

(2) Carrier selection (Order sensitive)
(Name) Sort by name / (Bytes) Sort by bytes

```
Name                 Bytes   Chain Order
PDF32000_2008.pdf       16   # 00000
ISO 14496-2.pdf         32   # 00001
ISO 14496-25.pdf        96   # 00002
ISO 14496-3.pdf         96   # 00003
ISO 32000-1.pdf        160   # 00004
BRICE.JPG              320   # 00005
HOUSE.JPG              672   # 00006
FALLS.JPG              864   # 00007
```

(-) Move up selected / (+) Move down selected / (Del) Delete selected
Selected 1.392 bytes

(3) Bit selection options
3gpp (Stream), Aiff (Audio), Bitmap (Image), Flv (Stream), Jpeg (Image): 1/8 (12%) - Minimum, 1/7 (14%) - Very Low, 1/6 (17%) - Low, [✓ 1/5 (20%) - Medium], 1/4 (25%) - High, 1/3 (33%) - Very High, 1/2 (50%) - Maximum; Mp3 (Audio), Mp4 (Stream), Mpg I/II (Stream), Next/Sun (Audio), Pcx (Image), Pdf (File), Png (Image), Swf (Stream), Tga (Image), Vob (Stream), Wave (Audio)



(1) Insert 3 uncorrelated passwords (Min: 8, Max: 32)
Cryptography (A) ************** (B) ***************
Scrambling (C) ***************  Enable (B) ☑ (C) ☑
Passwords check H(A,B) H(A,C) H(B,C) = (30%, 33%, 32%)
H(X,Y) = Hamming distance (X)(Y) >= 25%

(2) Carrier selection (Order sensitive)
(Name) Sort by name / (Bytes) Sort by bytes

```
Name                 Bytes   Chain Order
PDF32000_2008.pdf       16   # 00000
ISO 14496-2.pdf         32   # 00001
ISO 14496-25.pdf        96   # 00002
ISO 14496-3.pdf         96   # 00003
ISO 32000-1.pdf        160   # 00004
BRICE.JPG              816   # 00005
HOUSE.JPG            1.696   # 00006
```

(-) Move up selected / (+) Move down selected / (Del) Delete selected
Selected 2.912 bytes

(3) Bit selection options
3gpp (Stream), Aiff (Audio), Bitmap (Image), Flv (Stream), Jpeg (Image): 1/8 (12%) - Minimum, 1/7 (14%) - Very Low, 1/6 (17%) - Low, 1/5 (20%) - Medium, 1/4 (25%) - High, 1/3 (33%) - Very High, [✓ 1/2 (50%) - Maximum]; Mp3 (Audio), Mp4 (Stream), Mpg I/II (Stream), Next/Sun (Audio), Pcx (Image), Pdf (File), Png (Image), Swf (Stream), Tga (Image), Vob (Stream), Wave (Audio)

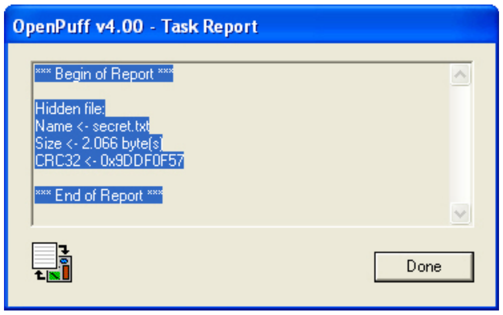
| | |
|---|---|
| (*Reset Options*) | Reset all bits selection level |
| (*Unhide!*) | Start unhiding |

After
- typing twice the same password
- adding all the carriers, in the right order
- setting all bits selection levels to the original value

start the unhiding task

Options: bits selection level

## Task report:



```
*** Begin of Report ***

Hidden file:
Name <- secret.txt
Size <- 2.066 byte(s)
CRC32 <- 0x9DDF0F57

*** End of Report ***
```

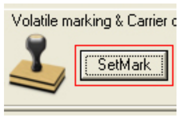
If carriers have been added in the right order, with the original bits selection levels, OpenPuff will be able to reconstruct the original data. For better security, data will be reconstructed only after a successful CRC check.

*Even the slightest change in one of the carrier could damage the data and prevent every unhiding try.*

Back

## MARK SETTING STEP BY STEP

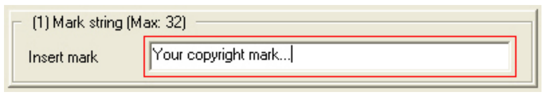
### BEGIN:



|  |  |
|---|---|
| (*Set Mark*) | Go to mark setting panel |

Select *Set Mark*.

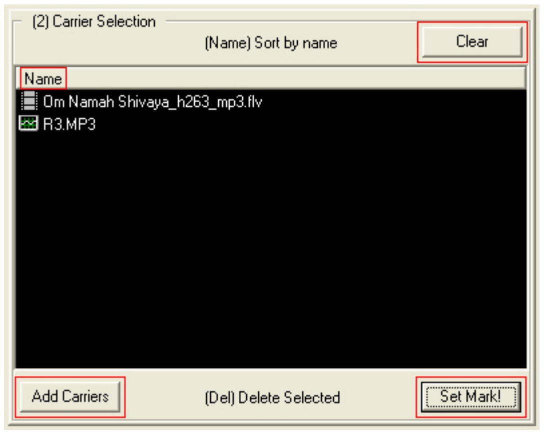
### STEP 1 – CHOOSE MARK:



|  |  |
|---|---|
| (*Insert mark*) | Your mark |

Type once your mark.

### STEP 2 – CHOOSE CARRIER(S):



|  |  |
|---|---|
| (*Clear*) | Discard all carriers |
| (*Add*) | Add new carriers to the list |
| (*Name*) | Sort carriers by name |
| (*Del*) | Delete selected carriers |
| (*Set Mark!*) | Start mark setting |

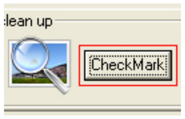
Add all the carriers that need to be marked.
Start the setting task.

SUPPORTED FORMATS IN DETAIL

BACK

## Mark Checking step by step

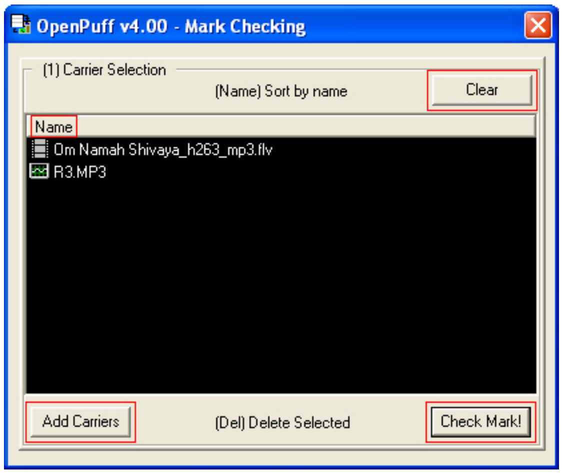
### Begin:



| | |
|---|---|
| (*Check Mark*) | Go to mark checking panel |

Select *Check Mark*.

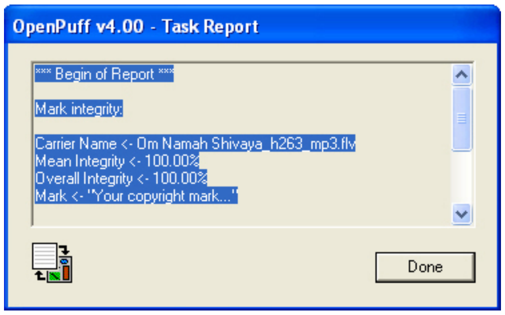
### Step 1 – Choose carrier(s):



| | |
|---|---|
| (*Clear*) | Discard all carriers |
| (*Add*) | Add new carriers to the list |
| (*Name*) | Sort carriers by name |
| (*Del*) | Delete selected carriers |
| (*Set Mark!*) | Start mark checking |

Add all the carriers that need to be checked. Start the checking task.
Supported formats in detail

### Task report:



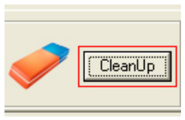
```
*** Begin of Report ***

Mark integrity:

Carrier Name <- Om Namah Shivaya_h263_mp3.flv
Mean Integrity <- 100.00%
Overall Integrity <- 100.00%
Mark <- "Your copyright mark..."
```

End report summarizes, for each carrier, integrity and mean integrity information.

Back

## Data & Mark Erasing step by step

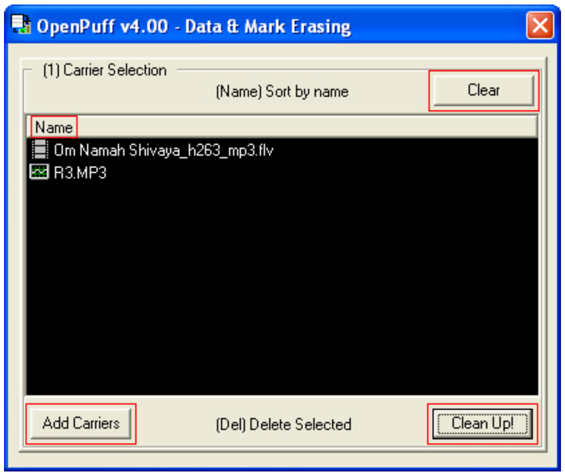
**Begin:**



| | |
|---|---|
| (*Clean Up*) | Go to data & mark erasing panel |

Select *Clean Up*.

**Step 1 – Choose carrier(s):**



| | |
|---|---|
| (*Clear*) | Discard all carriers |
| (*Add*) | Add new carriers to the list |
| (*Name*) | Sort carriers by name |
| (*Del*) | Delete selected carriers |
| (*Clean Up!*) | Start data & mark erasing |

Add all the carriers that need to be cleaned and start the cleaning task.

Supported formats in detail

Back