

Computer & Network Forensics

Week 8 (Lab5)

Steganography

Documentation

Tool 1. OpenPuff (GUI)

Data Embedding

Download: [from here](#), and run on Windows or with `wine` on UNIX (Linux/MacOS)

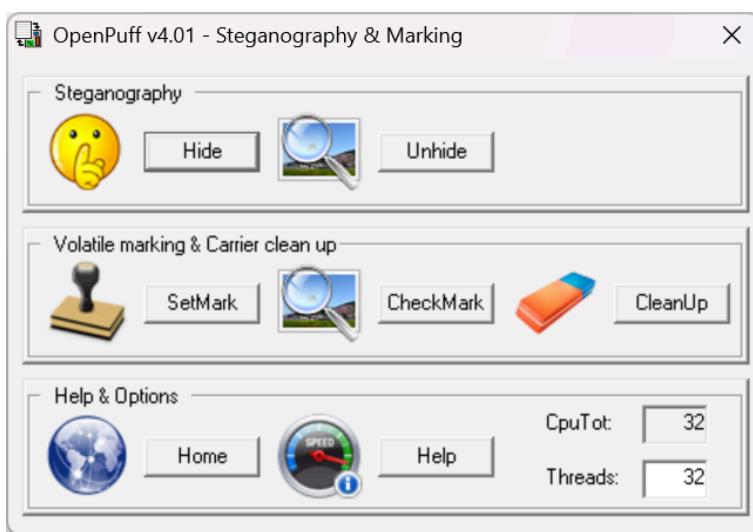
Embed:

- Open OpenPuff → choose “Hide”
- Add carrier (cover.wav), select data (secret.txt), set a password(s) and optional decoy carriers, click “Hide Data!”

Extract:

- Open OpenPuff → choose “Unhide”
- Select stego carrier, enter keys/passwords, click “Unhide!”

Windows (native)



Hide:

OpenPuff v4.01 - Data Hiding

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

Cryptography (A) _____ (B) _____ (C) _____

Scrambling (C) _____ Enable (B) _____ (D) _____

Passwords check: A + B + C

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

(2) Data (Max: 256Mb)

Target: C:\Users\dany\Downloads\secret.b

Size: 10 * name[10] + data[6] bytes

(3) Carrier selection [Order sensitive]

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

Shuffle _____ Clear _____

Name	Bytes	Chain Order
cover.png	3 403	#00000
cover.wav	16 544	Unused

(4) Bit selection options

- 3ogg (Stream)
- Aiff (Audio)
- Bmp (Image)
- Fif (Image)
- Jpeg (Image)
- Mp3 (Audio)
- Mp4 (Stream)
- Mov (Image)
- Pcx (Image)
- Png (Image)
- Svg (Image)
- Tga (Image)
- Vob (Stream)
- Wave (Audio)

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

Add Selected / Total: 25 952 / 26 bytes

Reset Options Add Decoy Hide Data

Hiding Carriers

Now hiding... 0%

Task completed - Warning

1/2 carriers processed

Last carrier hasn't been processed

Don't load unprocessed carriers during Unhiding!

OK

OpenPuff v4.01 - Task Report

*** Begin of Report ***

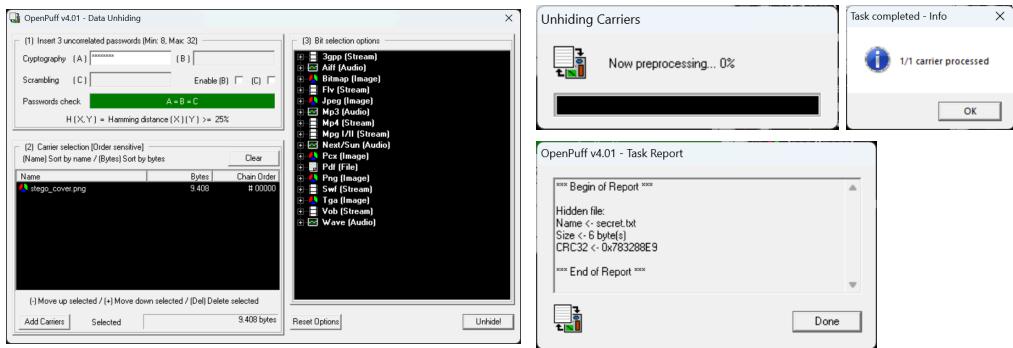
Bit Selection: Png (Image) < 1/5 (20%) - Medium

Carriers: #00000 <- cover.png

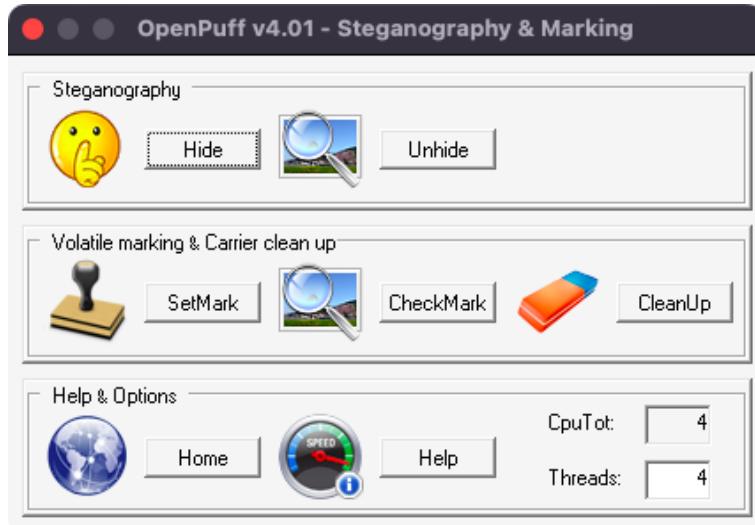
*** End of Report ***

Done

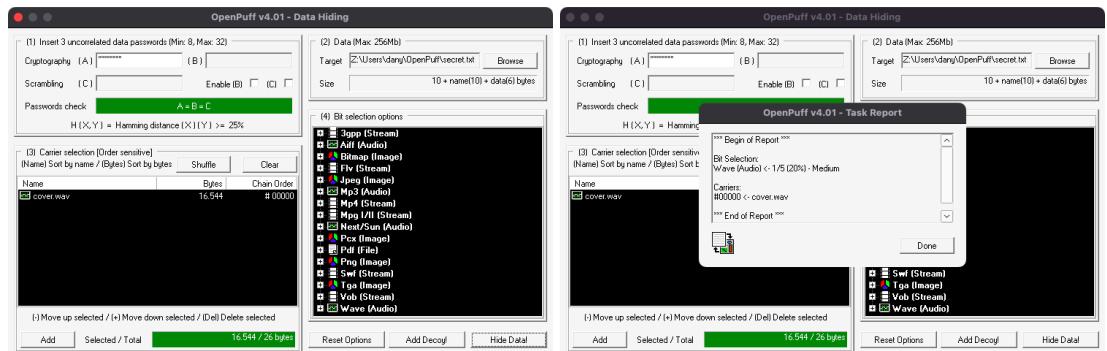
Unhide:



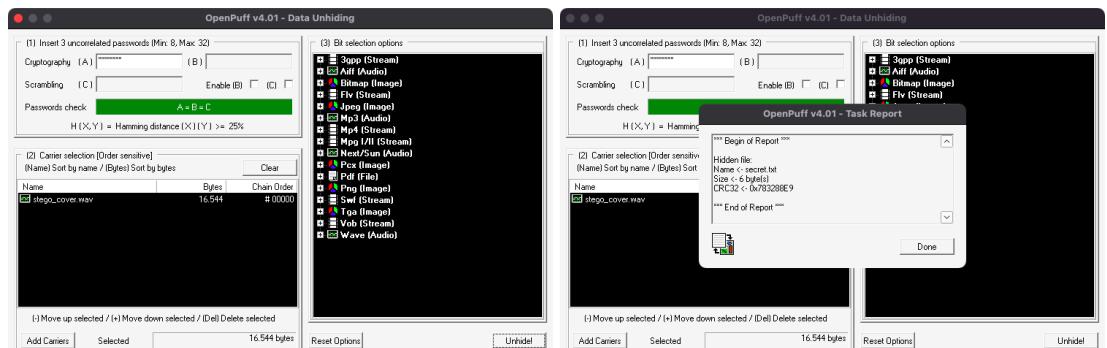
MacOS (with wine)



Hide:



Unhide:



Hash Comparison

Windows (*cmd – certutil*)

→ MD5

```
C:\Users\Dan\OpenPuff>certutil -hashfile cover.png MD5
MD5 hash of cover.png:
02e0a3c2fa58b8a0d5e4b2a
CertUtil: -hashfile command completed successfully.

C:\Users\Dan\OpenPuff>certutil -hashfile stego_cover.png MD5
MD5 hash of stego_cover.png:
02e0a3c2fa58b8a0d5e4b2a
CertUtil: -hashfile command completed successfully.

C:\Users\Dan\OpenPuff>
```

→ SHA1

```
C:\Users\Dan\OpenPuff>certutil -hashfile cover.png SHA1
SHA1 hash of cover.png:
D5E5A1F000000000000000000000000000000000
CertUtil: -hashfile command completed successfully.

C:\Users\Dan\OpenPuff>certutil -hashfile stego_cover.png SHA1
SHA1 hash of stego_cover.png:
D5E5A1F000000000000000000000000000000000
CertUtil: -hashfile command completed successfully.

C:\Users\Dan\OpenPuff>
```

→ SHA256

```
C:\Users\Dan\OpenPuff>certutil -hashfile cover.png SHA256
SHA256 hash of cover.png:
D5E5A1F000000000000000000000000000000000
CertUtil: -hashfile command completed successfully.

C:\Users\Dan\OpenPuff>certutil -hashfile stego_cover.png SHA256
SHA256 hash of stego_cover.png:
D5E5A1F000000000000000000000000000000000
CertUtil: -hashfile command completed successfully.

C:\Users\Dan\OpenPuff>
```

MacOS (*zsh/bash – md5 & shasum*)

→ MD5

```
OpenPuff -- zsh -- 80x24
dany@Dan: OpenPuff % md5 cover.wav stego_cover.wav
MD5 (cover.wav) = edef2d6a3c2fa58b8a0d5e4b2a
MD5 (stego_cover.wav) = 1765c2e4e0bc5c5e795/a4a1e68dd0
dany@Dan: OpenPuff %
```

→ SHA1

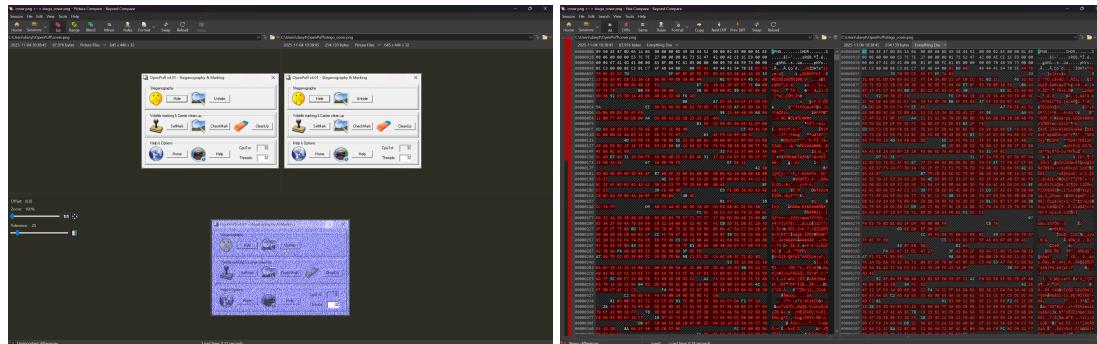
```
OpenPuff -- zsh -- 80x24
dany@Dan: OpenPuff % shasum -a 1 cover.wav stego_cover.wav
99d7d1295edf7fb83e221f77267a947d2c949d2 cover.wav
9775477f487d27e51e582ed79c2489a3d94b11 stego_cover.wav
dany@Dan: OpenPuff %
```

→ SHA256

```
OpenPuff -- zsh -- 80x24
dany@Dan: OpenPuff % shasum -a 256 cover.wav stego_cover.wav
67571ba4abcbf86f6cdeddf5fdfe941c1b0dd32ed856f963b5a373d4994aad
cover.wav
sc92ab5b05fc072446513128cb3a2e007f424cc6a5b767071/a4e8d2e929208
stego_cover.wav
dany@Dan: OpenPuff %
```

Binary/Hex Comparison

Beyond Compare ([ScooterSoftware](#)) – Windows



Tool 2. Steghide (CLI)

Data Embedding

Linux (*kali, in my case*)

Install: `sudo apt install steghide`

Embed: `steghide embed -cf cover.jpg -ef secret.txt -sf stego_cover.jpg`

Extract: `steghide extract -sf stego_cover.jpg`

The terminal window shows the following sequence of commands and outputs:

```
# steghide secret -ef secret.txt
# steghide embed -cf cover.jpg -ef secret.txt -sf stego_cover.jpg
# Enter password:
# steghide embed -cf cover.jpg -ef secret.txt -sf stego_cover.jpg
writing stego file "stego_cover.jpg"...
done
# steghide extract -sf stego_cover.jpg
# Enter password:
# Extracted data to "secret.txt".
# cat secret.txt
secret
```

Hash Comparison

Linux (*bash – md5sum & sha1sum, sha256sum*)

md5sum → `md5sum cover.jpg stego_cover.jpg`

sha1sum → `sha1sum cover.jpg stego_cover.jpg`

sha256sum → `sha256sum cover.jpg stego_cover.jpg`

The terminal window shows the following command and output:

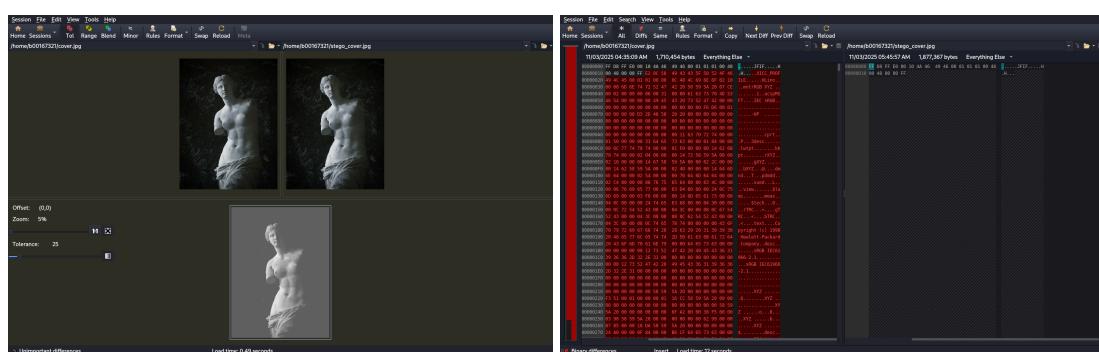
```
# md5sum cover.jpg stego_cover.jpg
# sha1sum cover.jpg stego_cover.jpg
# sha256sum cover.jpg stego_cover.jpg
```

Outputs:

```
000010732141:kali:~$ md5sum cover.jpg stego_cover.jpg
000010732141:kali:~$ sha1sum cover.jpg stego_cover.jpg
000010732141:kali:~$ sha256sum cover.jpg stego_cover.jpg
```

Binary/Hex Comparison

Beyond Compare ([ScooterSoftware](#)) – Linux



```
cmp -l (list differing bytes) | xxd & diff -u (hex comparison)
```

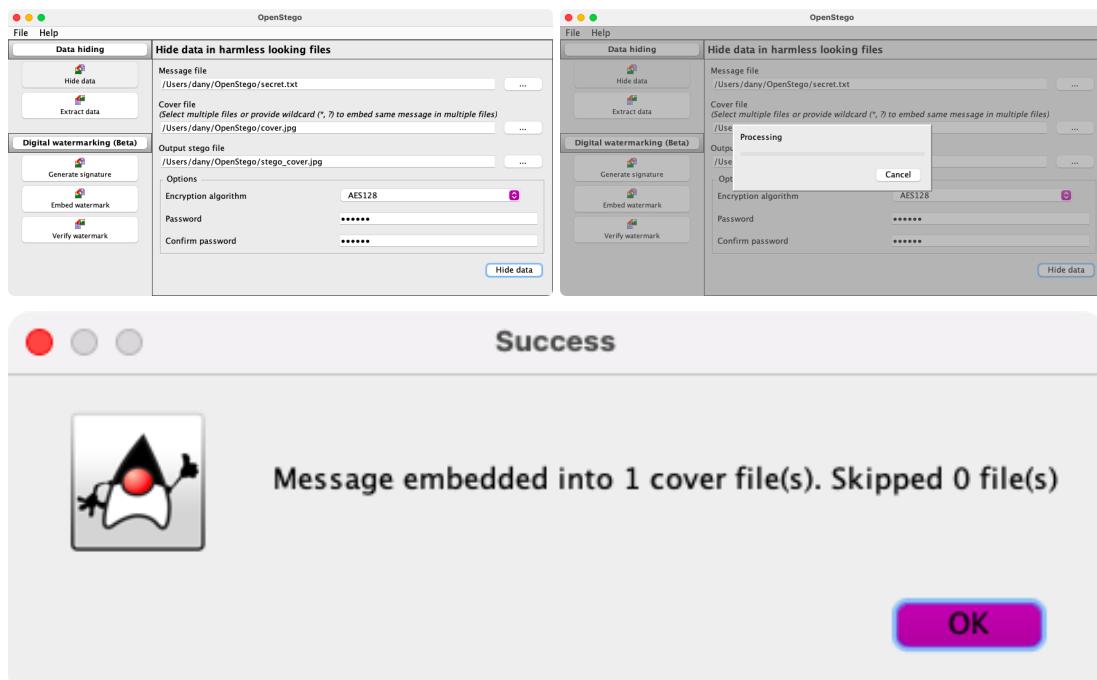
Tool 3. OpenStego (Java GUI)

Data Embedding

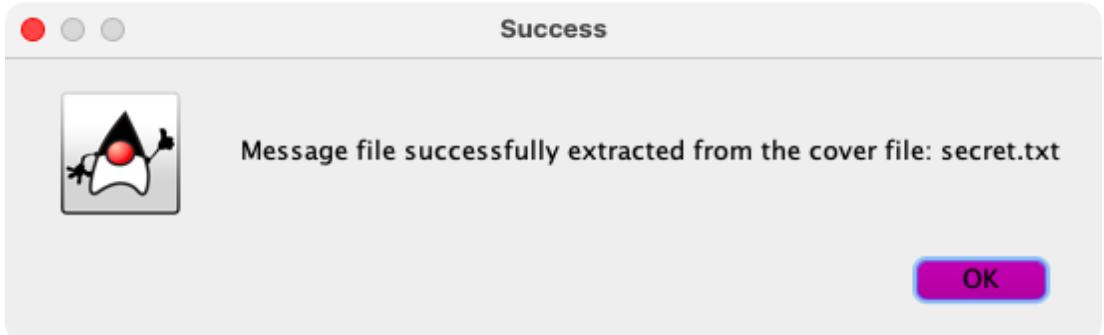
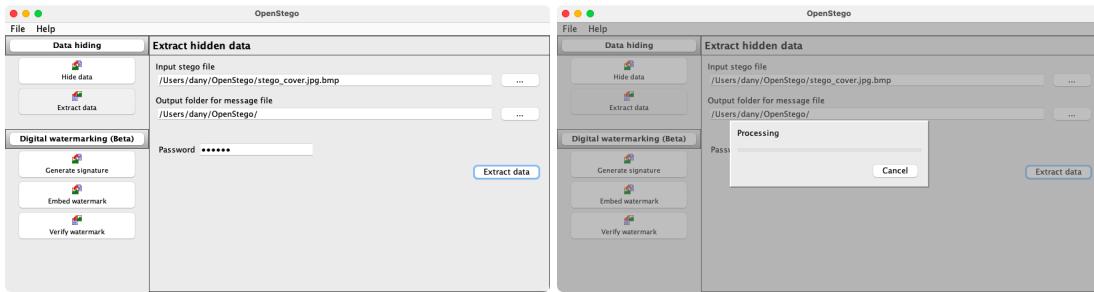
Download: [from here \(github – openstego/releases\)](#), and run jar file
(java need to be installed)

MacOS (jar file application – can be executed on any machine with java installed)

Hide data:



Extract data:



Hash Comparison

MacOS (zsh/bash – md5 & shasum)

→ MD5

```
OpenStego = zsh - 80x24
dany@Dany:~$ md5 cover.jpg stego_cover.jpg.bmp
MD5 (cover.jpg) = e85f1932818d1ba52618391e6a397c01
MD5 (stego_cover.jpg.bmp) = 6f1c77ef286851ba86f97988f3da87a1
dany@Dany:~$ OpenStego %
```

→ SHA1

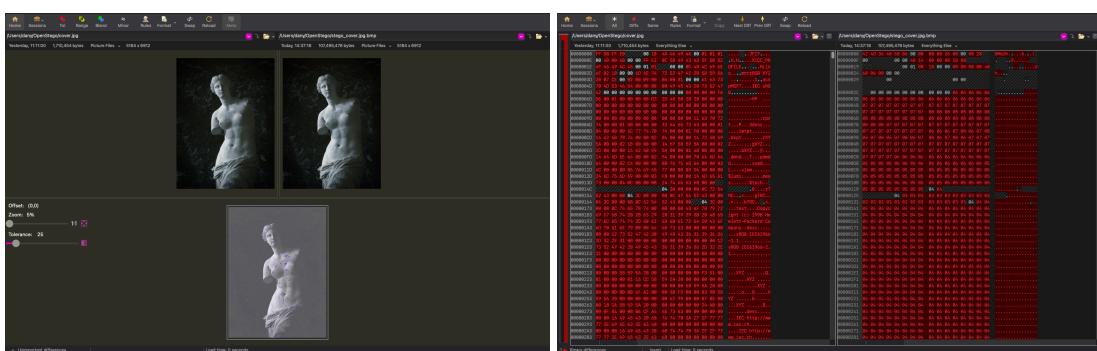
```
OpenStego = zsh - 80x24
dany@Dany:~$ shasum -a 1 cover.jpg stego_cover.jpg.bmp
18dd8b13a931cb1a19b7f3880804384972328ca497  cover.jpg
c25bc98409ec291dec115b6b401d128ae78935b01  stego_cover.jpg.bmp
dany@Dany:~$ OpenStego %
```

→ SHA256

```
OpenStego = zsh - 80x24
dany@Dany:~$ shasum -a 256 cover.jpg stego_cover.jpg.bmp
ba57729588c11fd7838c691a1de1a890938a85cc7c8a3d1ccb58327a8e4f4  cover.jpg
9eddrf41vabf1df616025a441850f1cd2b2b15db29c8b19b81ee826aeb7b5  stego_cover.jpg.bmp
dany@Dany:~$ OpenStego %
```

Binary/Hex Comparison

Beyond Compare ([ScooterSoftware](#)) – MacOS

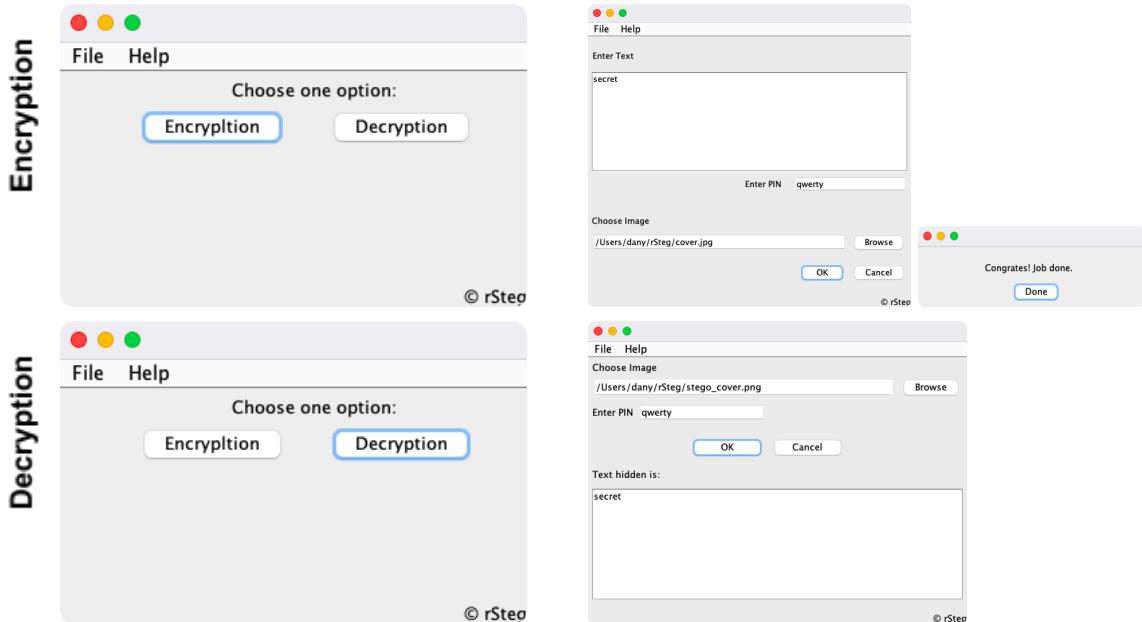


Tool 4.rSteg

Data Embedding

[Download: from here \(softpedia.com\)](#), and run jar file (java need to be installed)

MacOS (*jar file application – can be executed on any machine with java installed*)



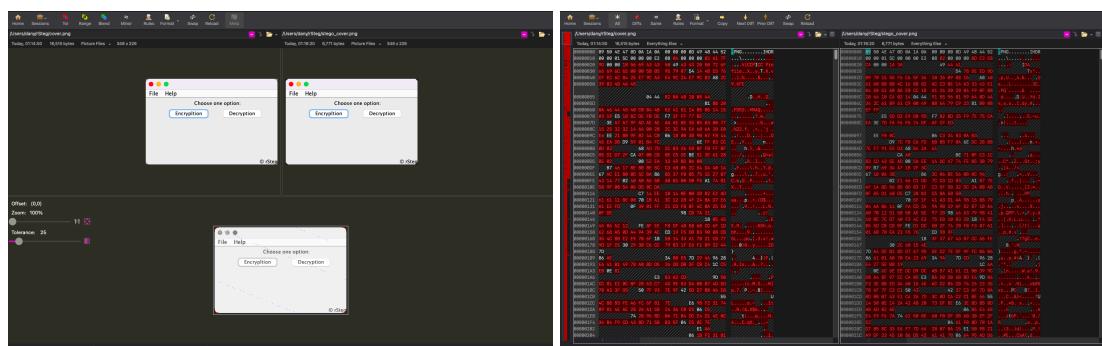
Hash Comparison

MacOS (*zsh/bash – md5 & shasum*)



Binary/Hex Comparison

[Beyond Compare \(ScooterSoftware\)](#) – MacOS



Questions

1. How would you Investigate Hidden Information?
 - Compute file hashes (MD5, SHA-1, SHA-256) and compare with known originals to detect any change.
 - Use metadata tools (`exiftool`) to identify suspicious metadata.
 - Use steganalysis tools such as `stegdetect`, `zsteg`, `stegseek`, and `binwalk` to check for signatures or anomalies.
 - Perform entropy analysis to detect unnatural uniformity or noise bumps.
 - Do a binary/hex comparison (Beyond Compare or `xxd + diff`) to identify changed byte offsets.
 - Inspect image bitplanes and color channels (with `stegsolve`) and check LSB planes visually.
 - Attempt extraction with known stego tools (OpenPuff, Steghide, OpenStego) using guessed passwords if necessary.
2. Compare and Contrast Steganography and Encryption:
 - **Purpose:** Encryption hides content (makes it unreadable), steganography hides existence of a message.
 - **Visibility:** Encrypted data is visible but unintelligible; steganographic data is concealed inside a carrier to avoid detection.
 - **Use together:** Best practice often combines both — encrypt payload first, then hide it (defense-in-depth).
 - **Detection:** Encrypted files usually show high entropy and are clearly different; steganography aims to minimize detectable changes so carriers look normal.
 - **Legal/forensic implications:** Stego can bypass cursory inspections, while encryption draws attention (may trigger suspicion or legal obligations in some jurisdictions).
3. List Data Breaches Involving Steganography:
 - **Stegoloader (Gatak) Malware (2015)** – Malware used steganography to conceal its code within PNG images downloaded from the web to evade antivirus detection.
 - **Turla APT Campaign (2019)** – The Turla hacking group embedded encrypted commands in Instagram image comments to control malware remotely.
 - **Vawtrak Banking Trojan (2016)** – Used steganography to hide configuration data inside favicons, making detection more difficult.
4. List 4 Authoritative Figures in Steganography:
 - **Jessica Fridrich** – Leading researcher in digital image steganography and steganalysis, known for developing modern detection techniques.
 - **Neil F. Johnson** – Early pioneer who co-authored foundational papers on steganography and steganalysis in the 1990s.
 - **Niels Provos** – Creator of the OutGuess tool and researcher on statistical defenses against steganalysis.
 - **Sushil Jajodia** – Academic expert in information hiding and computer security; co-authored seminal works on steganography theory.