a Documentation

a

Apr 30, 2020

Contents:

CHAPTER 1

Cryptosteganography

A python steganography module to store messages or files protected with AES-256 encryption inside an image.

Steganography is the art of concealing information within different types of media objects such as images or audio files, in such a way that no one, apart from the sender and intended recipient, suspects the existence of the message. By default steganography is a type of security through obscurity.

Additionally this module also enhance the security of the steganography through data encryption. The data concealed is encrypted using AES 256 encryption, a popular algorithm used in symmetric key cryptography.

1.1 Prerequisites

Python 3+

pip3

(Most Linux systems comes with python 3 installed by default).

1.2 Dependencies Installation (Ubuntu)

```
$ sudo apt-get install python3-pip
```

1.3 Dependencies Installation (MacOS)

To install Python3 I recommend use Homebrew package manager

The script will explain what changes it will make and prompt you before the installation begins.

```
$ ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/
↪install)"
```

Edit your ~/.profile to include (if it is not already there)

```
export PATH=/usr/local/bin:/usr/local/sbin:$PATH
```

To install Python 3:

```
$ brew install python3
```

1.4 Installation

To install the package just run

```
$ pip3 install cryptosteganography
```

1.5 Usage

1.5.1 Use as a library in a python program

Store a message string inside an image

```
from cryptosteganography import CryptoSteganography

crypto_steganography = CryptoSteganography('My secret password key')

# Save the encrypted file inside the image
crypto_steganography.hide('input_image_name.jpg', 'output_image_file.png', 'My secret␣
↪message')

secret = crypto_steganography.retrieve('output_image_file.png')

print(secret)
# My secret message
```

Store a binary file inside an image

Note: This only works if the concealed file size is smaller than the input image

```
from cryptosteganography import CryptoSteganography

message = None
with open('sample.mp3', "rb") as f:
    message = f.read()
```

(continues on next page)

(continued from previous page)

```
crypto_steganography = CryptoSteganography('My secret password key')

# Save the encrypted file inside the image
crypto_steganography.hide('input_image_name.jpg', 'output_image_file.png', message)

# Retrieve the file ( the previous crypto_steganography instance could be used but I␣
↪instantiate a brand new object
# with the same password key just to demonstrate that can it can be used to decrypt)
crypto_steganography = CryptoSteganography('My secret password key')
decrypted_bin = crypto_steganography.retrieve('output_image_file.png')

# Save the data to a new file
with open('decrypted_sample.mp3', 'wb') as f:
    f.write(secret_bin)
```

1.5.2 Use as a python program

Check help at command line prompt to learn how to use it.

```
$ cryptosteganography -h
usage: cryptosteganography [-h] {save,retrieve} ...

A python steganography script that save/retrieve a text/file (AES 256
encrypted) inside an image.

positional arguments:
  {save,retrieve}  sub-command help
    save           save help
    retrieve       retrieve help

optional arguments:
  -h, --help       show this help message and exit
```

Save sub command help

```
$ cryptosteganography save -h
usage: cryptosteganography save [-h] -i INPUT_IMAGE_FILE
                                (-m MESSAGE | -f MESSAGE_FILE) -o
                                OUTPUT_IMAGE_FILE

optional arguments:
  -h, --help            show this help message and exit
  -i INPUT_IMAGE_FILE, --input INPUT_IMAGE_FILE
                        Input image file.
  -m MESSAGE, --message MESSAGE
                        Your secret message to hide (non binary).
  -f MESSAGE_FILE, --file MESSAGE_FILE
                        Your secret to hide (Text or any binary file).
  -o OUTPUT_IMAGE_FILE, --output OUTPUT_IMAGE_FILE
                        Output image containing the secret.
```

Retrieve sub command help

```
$ cryptosteganography retrieve -h
usage: cryptosteganography retrieve [-h] -i INPUT_IMAGE_FILE [-o RETRIEVED_FILE]
```

(continues on next page)

(continued from previous page)

```
optional arguments:
  -h, --help            show this help message and exit
  -i INPUT_IMAGE_FILE, --input INPUT_IMAGE_FILE
                        Input image file.
  -o RETRIEVED_FILE, --output RETRIEVED_FILE
                        Output for the binary secret file (Text or any binary
                        file).
```

Save message example

```
$ cryptosteganography save -i 4824157.png -m "My secret message..." -o output.png
Enter the key password:
Output image output.png saved with success
```

Retrieve message example

```
$ cryptosteganography retrieve -i output.png
Enter the key password:
My secret message...
```

Save file example

```
$ cryptosteganography save -i input_image_name.jpg -f duck_logo.pem -o output_file.png
Enter the key password:
Output image output_file.png saved with success
```

Retrieve file example

```
$ cryptosteganography retrieve -i output.png -o decrypted_file
Enter the key password:
decrypted_file saved with success
```

1.6 License

This project is licensed under the MIT License - see the LICENSE file for details

1.7 Authors

Vin Busquet file for details

1.8 Limitations

- Only works with python 3
- It does not work if the conceived file is greater than original input file
- Ouput image is limited to PNG format only.
- I did not tested with all conceived file types. Feel free to report any bug you find

1.9 Contributing

For details, check out CONTRIBUTING.md.

1.10 Changelog

For details, check out CHANGELOG.md.

1.11 Acknowledgments

- PyCryptodome
- Stéganô

CHAPTER 2

Indices and tables

- genindex
- modindex
- search