Compiling and using the software

The software was developed in the Rust language for command-line use. The source code is available at: https://github.com/yugi386/steganography. To compile, simply use the command:

```
cargo build --release
```

To perform the encryption, you will need a folder containing at least one image. The command to encrypt it is:

```
./target/release/esteganografia hide   --input-dir "[IMAGE_FOLDER]"   --file "[CLEARTEXT]"   --output-dir "[OUTPUT_FOLDER]"   --password "[PASSWORD]"
```

The output at the terminal will look similar to this:

```
--- HIDE MODE (INFINITE RECYCLING) ---
Source pool size: 95 images.
Total payload size to hide: 108192299 bytes
[0] Saved '7JUZRrBL.png' (Seq: 0). Points to -> 'a64aiX14.png'
[1] Saved 'a64aiX14.png' (Seq: 1). Points to -> 'GKwaRdy2.png'
[2] Saved 'GKwaRdy2.png' (Seq: 2). Points to -> 'af4ErlMO.png'
…
[203] Saved 'iQ3rKMAD.png' (Seq: 203). Points to -> 'uZJtigFD.png'
[204] Saved 'uZJtigFD.png' (Seq: 204). Points to -> 'fUfRiZqi.png'
[205] Saved 'fUfRiZqi.png' (Seq: 205). Points to -> 'END'

SUCCESS! Chain created with 206 fragments.
--- SCRAMBLING FILE CREATION ORDER ---
Re-creating 206 files in random order to reset 'Birth Time'...
Creation order successfully scrambled.
--- SANITIZING METADATA (TIMESTAMPS) ---
Shuffling timestamps of 206 files...
Timestamps successfully altered. Chronological order destroyed.
```

To perform the decryption, use:

```
./target/release/esteganografia scan   --dir "[OUTPUT_FOLDER]"   --password "[PASSWORD]"
```

The terminal output will look something like this:

```
--- SCANNER MODE ---
Searching for chain start in '[OUTPUT_FOLDER]'...
.....................
[SUCCESS] Potential start found: "7JUZRrBL.png"
 -> Next file pointed to: 'a64aiX14.png'
 -> Starting automatic recovery...

--- RECOVERY MODE ---
Processing: "7JUZRrBL.png"
Processing: "a64aiX14.png"
Processing: "GKwaRdy2.png"
Processing: "af4ErlMO.png"
Processing: "Y7s3gl6u.png"
Processing: "9znlJh6z.png"
Processing: "jsWJFftQ.png"
Processing: "QJOx1gQG.png"
…
Processing: "uZJtigFD.png"
Processing: "fUfRiZqi.png"
End of chain detected.
Decompressing...
SUCCESS! File saved as: RECOVERED_programa
```

To increase security (but also size), you can include "decoys" using the following command:

```
./target/release/esteganografia decoys   --input-dir [IMAGE_FOLDER]"  --output-dir "[OUTPUT_FOLDER]"
```

The terminal output will look something like this:

```
--- GENERATING DECOYS (RECYCLED SOURCE) ---
Generating 95 new decoys based on random source picks...
Decoy created: UuhNlRQ2.png
Decoy created: Ggv2fjQy.png
Decoy created: DlhEwjkL.png
Decoy created: FDwbLpZX.png
Decoy created: RlaodYhT.png
Decoy created: Khy7sl0H.png
…
Decoy created: KtZhfgbs.png
Finished! 95 decoys added.
--- SCRAMBLING FILE CREATION ORDER ---
Re-creating 301 files in random order to reset 'Birth Time'...
Creation order successfully scrambled.
--- SANITIZING METADATA (TIMESTAMPS) ---
Shuffling timestamps of 301 files...
Timestamps successfully altered. Chronological order destroyed.
```

By inserting "decoys" we increase the security of the cryptogram, but as a side effect we have a significant increase in its size. Ideally, security should be balanced with data volume to make it easier to use. The increase in the size of the cryptogram does not have a significant effect on the decryption time.