

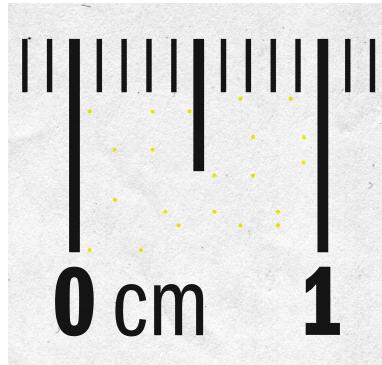
Podstawy steganografii i steganoanalizy

Dominik Lau, Sebastian Kutny, Tomasz Lewandowski, Maciej Krzyżanowski

15 maja 2023

Spis treści

1 Czym jest steganografia? Do czego służy?	2
1.1 Słowniczek	3
1.2 Podział steganografii	3
2 Przykłady rzeczywistych zastosowań steganografii	4
2.1 Historyczne przykłady użycia steganografii	4
2.2 Eurion	5
2.3 Znaki wodne	5
3 Przegląd technik steganografii	6
3.1 Modyfikacja LSB obrazu	6
3.2 Ukrywanie obrazów w spektrogramach	8
3.3 Ukrywanie archiwów w obrazach	10
3.4 Homoglify - Twitter Steganography	11
3.5 Chaffing i winnowing	11
4 Steganoanaliza	12
4.1 Zadania steganoanalizy	13
4.2 Podział steganoanalizy	13
4.3 Metody wykrywania steganografii	13
4.4 Problemy steganoanalizy	14
4.5 Steganoanaliza LSB	15
5 Narzędzia steganograficzne	17
5.1 Outguess	17
5.2 OpenStego	18
6 Narzędzia steganoanalytyczne	18
6.1 strings	18
6.2 binwalk	19
6.3 StegExpose	19
6.4 Sonic Visualizer	19



Rysunek 1: "kropki" zamieszczane przez drukarki

7 Źródła

20

1 Czym jest steganografia? Do czego służy?

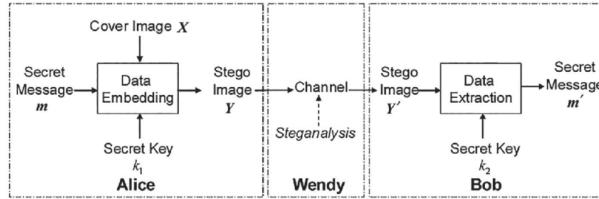
Steganografia polega na ukrywaniu informacji przez ukrywanie komunikacji w innej formie transmisji danych np. w obrazkach, plikach dźwiękowych, tekstowych. Zastosowania steganografii

- omijanie cenzury/szpiegostwo
- umieszczenie znaków wodnych
- ukryta wymiana danych
- dodawanie metadanych do plików (np. znaki sterujące)
- numery seryjne drukarek (za pomocą małych kropek)
- wprowadzanie opóźnień w pakietach sieciowych
- zastosowania w VoIP (steganofonia)
- zabezpieczanie banknotów (np. EURion constellation)

Steganografia może zatem realizować następujące funkcje bezpieczeństwa

- poufność
- autentyczność
- niezaprzeczalność
- integralność

Porównanie kryptografii i steganografii



Rysunek 2: model steganografii

	kryptografia	steganografia
cel	zapewnienie poufności	ukrycie komunikacji
obecność klucza	tak	opcjonalna
widoczność danych	nie	tak
modyfikacja struktury przetwarzanych danych	nie	tak

1.1 Słowniczek

- stegosystem - połączenie metod i narzędzi służących do tworzenia ukrytego kanału do przekazywania informacji
- wiadomość (payload) - przesyłane dane
- kontener/nośnik (carrier) - to wszelkie dane służące do ukrycia tajnej wiadomości
- stegokontener - dane i ukryta w nich tajna wiadomość
- kanał steganograficzny (stegochannel) - kanał transmisji stegokontenera
- klucz (stegokey) - tajny klucz potrzebny do ukrycia stegokontenera

1.2 Podział steganografii

Ze względu na kontener

- w plikach tekstowych
- w plikach audio
- w obrazach
- w ramkach różnych protokołów
- w plikach wykonawczych
- inne...

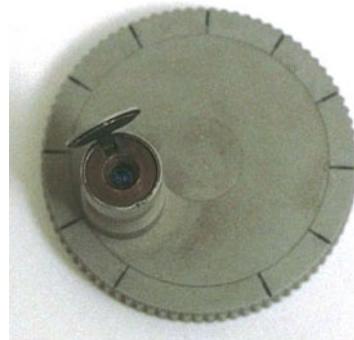
Ze względu na metodę modyfikacji nośnika

- **metody substytucji** - zamiana nadmiarowych danych nośnika
- **metody transformacyjne** - modyfikacja postaci falowej nośnika
- metody statystyczne - modyfikacja właściwości statystycznych nośnika
- metody generacji nośnika - ukrywanie informacji podczas tworzenia samego nośnika
- metody rozproszonego widma - ukrycie poprzez rozpraszanie danych
- metody znieksztalconiowe - wprowadzenie znieksztalczeń do nośnika i pozyskanie informacji poprzez porównanie nośnika oryginalnego i znieksztalonego

2 Przykłady rzeczywistych zastosowań steganografii

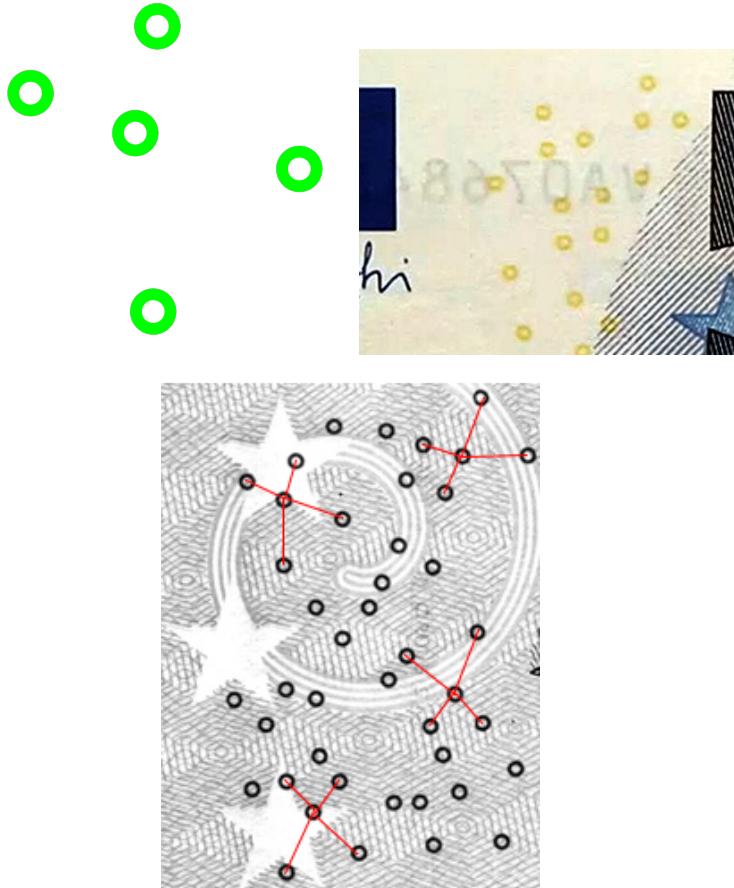
2.1 Historyczne przykłady użycia steganografii

Pierwsze wzmianki o użyciu technik steganograficznych można odnaleźć u Herodota w V wieku p.n.e. Opisuje on sposób tajnego przekazu informacji: tyran Histaios przetrzymywany przez króla perskiego Dariusza postanowił przesłać informację do swego zięcia Arystagorasa z Miletu, tak aby mogła się ona przedostać mimo pilnujących go strażników. Aby tego dokonać na wygolonej głowie swego niewolnika wytatuował przesłanie. Kiedy niewolnikowi odrosły włosy posłał go z oficjalnym, mało istotnym listem. W starożytnym Egipcie i Chinach stosowano atrament sympatyczny, czyli zapis wiadomości bezbarwną substancją (np. sok z cytryny, który zyskuje barwę przy podgrzaniu). Ponadto już podczas wojny francusko-pruskiej w 1871 a także 2 Wojny Światowej Niemcy wykorzystywali technikę mikrokropek wklejanych do tekstu maszynopisu. Na mikrokropkach widoczne były zdjęcia wysokiej rozdzielczości.



Rysunek 3: aparat do wykonywania mikrokropek, skala pomniejszenia ok. 1:300

2.2 Eurion



Rysunek 4: EURion, przykładowy układ na banknocie euro, dollarze

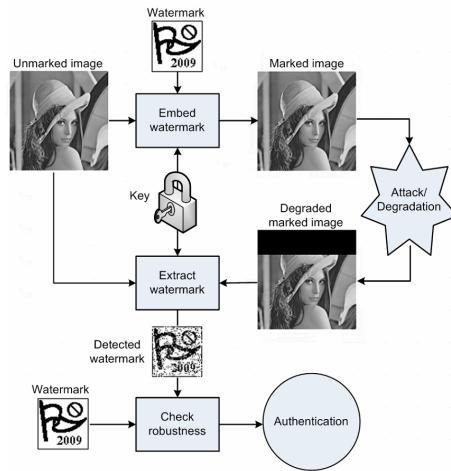
EURion jak i inne podobne zabiegi stanowią metodę przeciwdziałania fałszerstwom. Na banknotach umieszczane są zbiory kropek o różnych średnicach i względnych pozycjach (te parametry są sekretem). Kropki te tworzą fingerprint, który jest wykrywany przez oprogramowanie do skanowania (za pomocą metody detekcji wzorca) i wszelkie próby kopowania banknotów są blokowane.

2.3 Znaki wodne

Umieszczanie znaków wodnych w plikach ma na celu zamieszczenie informacji o właściwym prawie autorskim. Wykorzystujemy różnych metod steganografii w celu zapewnienia:

- trudności w usunięciu
- odporności na transformacje (**robustness**)
- niedostrzegalności (**perceptibility**)
- przepustowość

z czego najważniejszą cechą jest odporność na transformacje



Rysunek 5: schemat zamieszczania znaku wodnego



Rysunek 6: CAP(Coded Anti Piracy) - przykład znaku wodnego zamieszczanego w filmach do identyfikacji źródła nielegalnych kopii

3 Przegląd technik steganografii

3.1 Modyfikacja LSB obrazu

Jest to klasyczny algorytm steganografii, którego główną wadą jest łatwość w wykryciu/zniszczeniu wiadomości (np. przez wyzerowanie najmłodszych bitów). Przed nieuchcianym odczytem wiadomości możemy zapobiec poprzez zastosowanie kryptografii. Zasada działania algorytmu jest prosta:

1. wybierz, w którym kanale zapisać bity wiadomości (r,g,b, a, może obraz czarnobiały?)
2. zaszyfruj wiadomość wybranym algorytmem kryptograficznym
3. zastąp stare wartości najmłodszych bitów określonego kanału obrazu kolejnymi bitami zaszyfrowanej wiadomości

Przy odbieraniu wiadomości wyciągamy daną liczbę bitów ukrytych w pliku oraz deszyfrujemy tak skonstruowany szyfr, co daje nam wiadomość wynikową. Algorytm skutecznie ukrywa wiadomość w obrazie, ponieważ zamiana najmłodszych bitów pliku nie powoduje widocznej dla człowieka zmiany w jego odbiorze. Przy próbie wykonania tej samej operacji z MSB może okazać się, że zmiana jest na tyle drastyczna, że wcale nie ukrywa naszych szyfrowanych danych.

Analogiczna metoda jest możliwa na plikach dźwiękowych, tylko tam zmieniamy LSB próbki.

lsb_hide.py

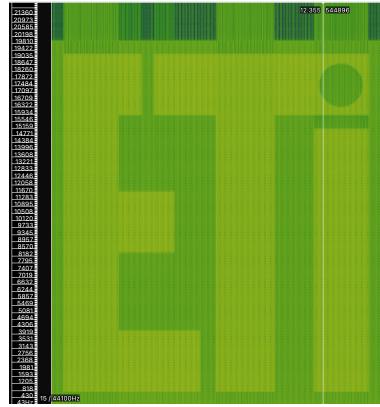
lsb_hide.py to nasza implementacja metody omówionej w poprzednim podpunkcie.

Skrypt ukrywa w wartościach RGB odpowiedniej ilości pikseli naszą zaszyfrowaną szyfrem AES w trybie licznikowym wiadomość. Na wyjściu otrzymujemy wygenerowany dla nas *nonce* oraz *key*. Argumentami potrzebnymi do wywołania są kolejno: ścieżka do pliku obrazu na którym chcemy ukryć wiadomość, ścieżka do pliku obrazu który zostanie zapisany jako wynik działania algorytmu oraz wiadomość którą chcemy ukryć.

Przykładowe użycie skryptu

```
python3 lsb_hide.py "your_file_path" "your_result_file_path"  
"your_message"
```

3.2 Ukrywanie obrazów w spektrogramach



Rysunek 7: Obraz zamieniony na dźwięk, bez żadnego "ukrywającego" pliku dźwiękowego, dźwięk ten jest odbierany jako szum

Inną z technik jest ukrywanie danych w określonych częstotliwościach pliku dźwiękowego. Jak to działa? Zaczynamy od konwersji obrazu na odcień szarości. Następnie korzystamy ze wzoru (IDFT - inverse discrete fourier transform)

$$x = \sum_{y=0}^{H-1} I[x, y] \sin\left(\frac{2\pi f i}{S}\right)$$

gdzie:

H to wysokość obrazu,

x to próbka odpowiadająca x -tej kolumnie obrazu,

$I[x, y]$ to jasność piksela o współrzędnych x, y ,

S to częstotliwość próbkowania,

i to numer próbki w bloku (odwrotna transformacja Fouriera operuje na blokach stałego rozmiaru, które łączy w wartość wielomianu w punkcie x)

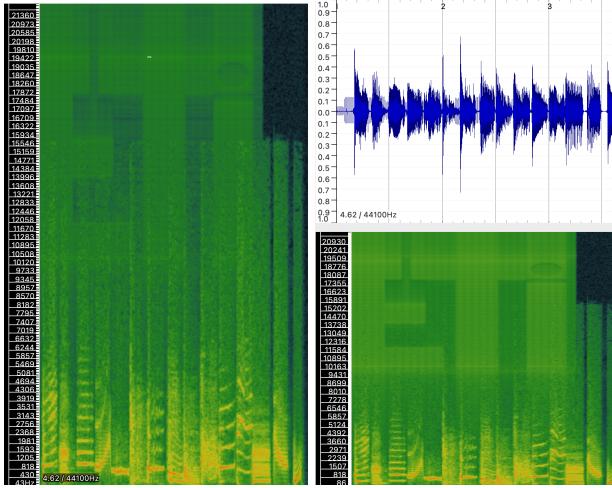
rozmiar bloku definiuje szerokość umieszczonego obrazu (w sensie ilości próbek)

$$f = y * \frac{f_{max} - f_{min}}{H} + f_{min}$$

f_{min} to częstotliwość, od której zaczyna się dolna krawędź obrazu, f_{max} to górną krawędź (np. $f_{min} = 20$ Hz, $f_{max} = 20$ kHz). Chcąc dodać ukryty sygnał do istniejącego pliku dźwiękowego dodajemy sygnały

$$I' = I + \beta x$$

β to współczynnik tłumienia mający na celu ukrycie "brzęczenia" ukrytego obrazu



Rysunek 8: Z lewej: przytłumiony sygnał obrazu umieszczony w spektrogramie istniejącego pliku dźwiękowego, z prawej: ilustracja szumów przy braku tłumienia obrazu - szum tworzy gołyim okiem wąsy

audio.py

audio.py to nasza implementacja metody omówionej w poprzednim podpunkcie

```
usage: in ~/dev/stego on master ①②③ python3 audio.py --help
      [-h, --help]           show this help message and exit
      -i INPUT, --input INPUT
                            Name of the image to be converted.
      -o OUTPUT, --output OUTPUT
                            Name of the output wav file.).
      -c CARRIER, --carrier CARRIER
                            Name of the input wav file to be used as stegocarrier.
      -p PIXEL, --pixel PIXEL
                            'pixel size', samples per pixel
      -s STARTT, --starttt STARTT
                            second, from which to embed image into carrier
      -d DAMP, --damp DAMP specify damp i.e. the mute coefficient on signal
      -b BFREQ, --bfreq BFREQ
                            specify bottom frequency i.e. bottom edge of an image
      -t TFREQ, --tfreq TFREQ
                            specify top frequency i.e. top edge of an image
      -r SAMPLERATE, --samplerate SAMPLERATE
                            specify samplerate of an audio file
```

Rysunek 9: wywołanie instrukcji help, flaga -d to omówiony współczynnik tłumienia, -p to ilość próbek na piksel

Przykładowe użycia skryptu

```
python3 audio.py --input plik.png --output plik.wav
python3 audio.py --input plik.png --output plik.wav --carrier carrier.wav
```

domyślne wartości nieobowiązkowych parametrów można konfigurować w pierwszych linijkach skryptu

3.3 Ukrywanie archiwów w obrazach

Kolejny bardzo prosty sposób na ukrycie danych w obrazach polega na sklejeniu dwóch plików ze sobą, uzyskując tym samym plik polyglot. Wymaga to jednak, żeby formaty akceptowały "śmieci" przed nagłówkiem. Przykładami takich formatów są pdf, rar, zip. Metodę tą można łatwo wykryć na przykład za pomocą strings/binwalk.

Zapisywanie:

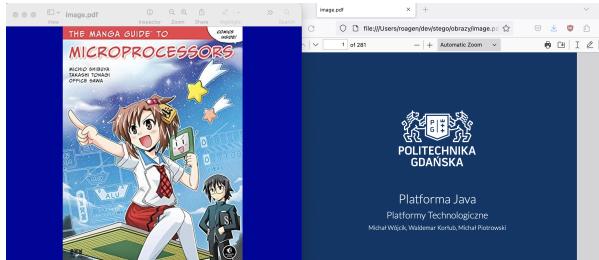
```
$ cat microprocessors.jpg JAVA_slajdy.pdf > image.jpg
```

W ten sposób ukrywać możemy zwykłe pliki tekstowe

```
$ cat microprocessors.jpg haslo.txt > image.jpg
```

wówczas

```
$ strings image.jpg
...
ri]P
:9w;
!`{?
haslo
```



Rysunek 10: Przykład pliku "poligloty", który może być jednocześnie interpretowany jako jpg i .pdf

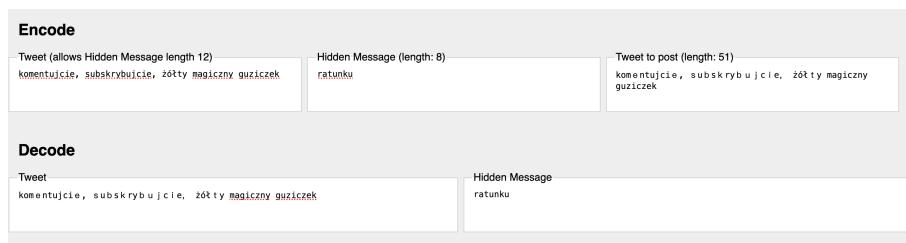
Szczególnym przypadkiem złączonych plików są pliki **GIFAR** - gif + jar. Przy nieodpowiednio zabezpieczonej stronie umożliwiającej umieszczanie gifów atakujący może uruchomić kod z pliku jar będącego częścią zamieszczonego gifa. Jary podobnie jak wszystkie formaty bazujące na formacie zip umożliwiają umieszczanie dodatkowych bajtów przed nagłówkiem.

3.4 Homoglify - Twitter Steganography

Homoglify to znaki, których kształty mogą być interpretowane na różne sposoby. Chcąc przy ich użyciu ukryć wiadomość musimy:

1. zdefiniować alfabet wiadomości
 - ile bitów na znak? np. 6
 - jak wygląda alfabet np. _abcdefghijklmnopqrstuvwxyz123456789
 - dla powyższego alfabetu znak a będzie miał kod 000001 a np. 1001100
2. ustalić tekst wiadomości, która będzie kontenerem
3. ustalić tekst ukrytej wiadomości i przekodować go na alfabet
4. dla każdego znaku kontenera
 - (a) sprawdzamy ile ma homoglifów h
 - (b) liczbę różnych homoglifów danego znaku możemy zakodować na $\lceil \log_2 h \rceil = b$ bitach
 - (c) bierzemy b bitów ukrywanej wiadomości i na ich podstawie wybieramy, który z homoglifów zapiszemy do tekstu wyjściowego
 - (d) czyli na przykład, jeżeli pierwsza litera kontenera to A, które ma 4 homoglify, to bierzemy 2 bity wiadomości, jeżeli jest to 00 to znaku nie zmieniamy, 01 - wybieramy pierwszy homoglif itd...

Żeby zdekodować wiadomość musimy znać alfabet i ilość bitów na znak.



Rysunek 11: przykład wiadomości zakodowanej za pomocą Twitter Steganography

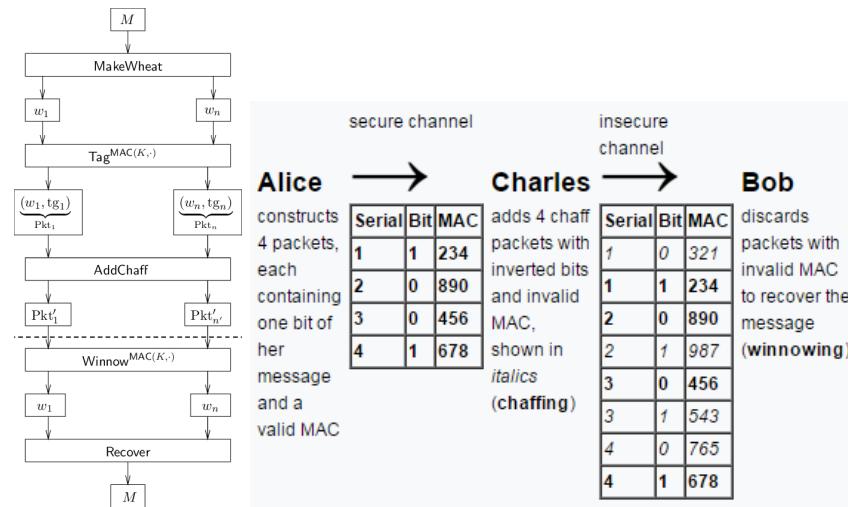
3.5 Chaffing i winnowing

Chaffing i winnowing to metoda z pogranicza kryptografii jak i steganografii. Technikę tą wymyślił Ron Rivest. Założymy, że Alicja chce wysłać wiadomość do Bogdana oraz wymienili między sobą klucz, który będą wykorzystywali w algorytmie MAC. Ponadto pakiety będą wysyłane bit po bicie, bity będą ponumerowane (żeby zachować ich kolejność).

1. Alicja wysyła pakiety wiadomości razem z tagiem MAC tych wiadomości
2. między pakietami losowo wysyła również zanegowane wartości bitów z losowymi wartościami tagu MAC (chaffing)
3. Bogdan odbiera pakiety i rozważa tylko te, dla których zgadzia się MAC (winnowing)

Metoda ta gwarantuje

1. poufność, podsłuchujący nie wie, który pakiet ma poprawny MAC
2. autentyczność, ponieważ poprawne pakiety są zabezpieczone za pomocą MAC



Rysunek 12: schemat działania metody i przykład

4 Steganoanaliza

Steganoanaliza (Steganalysis) jest tym samym dla steganografii, czym kryptoanaliza dla kryptografii - sztuką detekcji ukrytych wiadomości.



Rysunek 13: obraz z "edge ringing" i bez - jest to przewidywalne zniekształcenie; proste algorytmy steganografii mogą mieć problem z dobrym odwzorowaniem artefaktów o wysokim prawdopodobieństwie wystąpienia

4.1 Zadania steganoanalizy

- detekcja istnienia kanału steganograficznego (steganoanaliza pasywna)
- zniszczenie wiadomości w stegokontenerze
- ekstrakcja wiadomości ze stegokontenera

4.2 Podział steganoanalizy

Podział w zależności od posiadanych informacji

- atak skierowany na stegoobiekt - atakujący ma tylko podejrzany obiekt
- atak ze znajomością nośnika - atakujący ma dostęp do czystego nośnika
- atak ze znajomością wiadomości
- atak z wybranym stegoobiektem - atakujący zna algorytm maskowania
- atak z wybraną wiadomością
- atak ze znanym stegoobiektem - atakujący zna algorytm, czysty nośnik i stegoobiekt

4.3 Metody wykrywania steganografii

1. Metoda Analizy Statycznej

Polega na wykrywaniu nieprawidłowości lub nieoczekiwanych wzorców w danych w ich nośnikach, które mogą wskazywać obecność ukrytych danych. Analiza Statyczna może wykorzystywać przy różnego rodzaju plikach, przykładowo:

- rozkład wartości pikseli
- histogramy
- rozkład częstotliwości

Można wykorzystać również inne parametry statyczne pliku. Na przykład przy wykrywaniu obrazów może wykorzystywać analizę średnich średnich kolorów czy kontrastu.

2. Metoda analizy bitów najmniej znaczących
inaczej LSB, która została opisana bardzo dokładnie, razem z przykładem w dalszej części dokumentacji.

3. Metoda analizy spektralnej

Polega na analizie widma częstotliwościowego sygnału lub obrazu w celu wykrycia nieprawidłowości. Podczas analizy spektralnej zastosowana jest na przykład transformacja Fouriera. Transformacja Fouriera jest to rozkład sygnału w dziedzinie czasu na jego składowe częstotliwości. W ten sposób otrzymujemy informacje o występowaniu różnych częstotliwości w sygnale. Kiedy porównamy otrzymany informacje z wzorcami dla danego nośnika danych możemy wykryć nieprawidłowości, które sugerują obecność ukrytych danych.

4. Metoda analizy entropii

Entropia jest miarą nieprzewidywalności lub nieokreśloności, więc analiza entropii polega na ocenie stopnia losowości lub regularności danych w celu wykrycia nieprawidłowości, które będą sugerować ukryte dane.

5. Metoda analizy różnicowej

Tę metodę możemy przedstawić w kilku krokach:

- Posiadanie oryginalnego nośnika danych, niestety tu możemy mieć problem
- Porównanie z podejrzany nośnikiem
- Obliczanie różnic między oryginalnym nośnikiem, a podejrzany (np. na poziomie pikseli w obrazie czy próbek dźwięku)
- Analizujemy wzorce i anomalie, wyniki mogą nam pomóc w analizie innych plików już bez oryginalnego.

Ta metoda jest dosyć żmudna i zależy od dokładnej analizy wyników jakie dostajemy.

4.4 Problemy steganoanalizy

- wiele szyfrów ma taką właściwość, że produkuje szyfrogramy przypominające szum biały (o kompletnie płaskim widmie),
- barrage noise - bombardowanie stegokontenerami z losowymi/bezwartościowymi informacjami

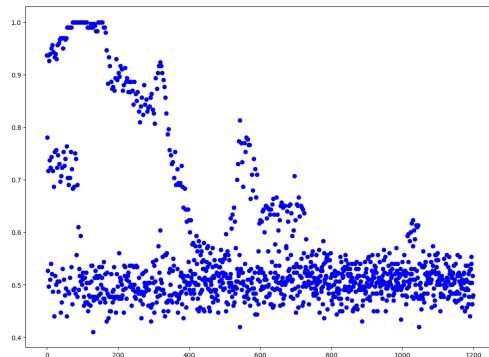
4.5 Steganoanaliza LSB

Prosta metoda wykrycia, czy obraz zawiera zakodowaną wiadomość

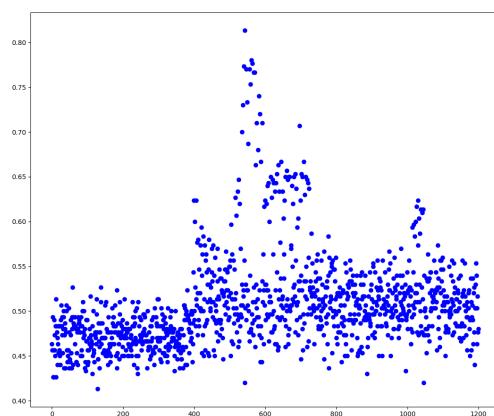
1. dzielimy piksele na bloki
2. dla każdego bloku liczymy wartość średnią LSB

Jeżeli w obrazie ukryto **zaszyfrowaną** wiadomość, to niektóre regiony bloków powinny mieć średnią ilość jedynek ≈ 0.5 (ze względu na losowy charakter szyfrów). Możliwe są oczywiście false positives, gdy piksele w niezmodyfikowanym obrazie mają taką charakterystykę.

Blokи podejrzane o zawieranie ukrytej wiadomości dobrze widać na porównaniu wykresu oryginalnego pliku z plikiem zmodyfikowanym.



Rysunek 14: Przykład obrazu oryginalnego wraz z jego histogramem LSB



Rysunek 15: Przykład obrazu zmodyfikowanego wraz z jego histogramem LSB

Przy deszyfrowaniu wiadomości potrzebna jest nam znajomość klucza. Sprawdzamy podejrzany blok tworząc wiadomość z bitów LSB bloku oraz próbujemy zdeszyfrować dane wybranym algorytmem poprzez rozpoczęcie prób deszyfrowania od jednego bajtu do całego bloku dodając po kolei po jednym bajcie więcej bloku. Tym sposobem powinniśmy w którymś momencie działania algorytmu otrzymać poprawną odszyfrowaną wiadomość.

lsb_find.py i lsb_decrypt.py

lsb_find.py i lsb_decrypt.py to nasza implementacja metody omówionej w poprzednim podpunkcie.

Skrypt lsb_find.py przeszukuje plik obrazu pod względem podejrzanych bloków o zawieranie ukrytej wiadomości. Po znalezieniu takiego, otrzymujemy jego koordynaty na zdjęciu.

Skrypt lsb_decrypt.py podejmuje próbę odszyfrowania wiadomości na podstawie koordynatów podejrzanych o zawieranie ukrytej wiadomości.

Argumentami potrzebnymi do wywołania skryptu lsb_find.py są kolejno: ścieżka do pliku obrazu, który podejrzymy o zawieranie ukrytej wiadomości, wielkość bloku, który będzie analizowany pod kątem zawierania ukrytej wiadomości oraz próg odchyłki od prawdopodobieństwa 0.5 uznawanego za charakterystykę szyfru.

Argumentami potrzebnymi do wywołania skryptu lsb_decrypt.py są kolejno: ścieżka do pliku obrazu, który podejrzymy o zawieranie ukrytej wiadomości, wielkość bloku, który według nas zawiera ukrytą wiadomość, koordynaty ukrytej wiadomości w obrazie (kolejno: szerekość i wysokość), nonce użyty w zaszyfrowaniu wiadomości oraz klucz użyty w zaszyfrowaniu wiadomości.

Przykładowe użycie skryptu lsb_find.py

```
python3 lsb_find.py "your_file_path" block_size threshold
```

Przykładowe użycie skryptu lsb_decrypt.py

```
python3 lsb_decrypt.py "your_file_path" block_size width height nonce key
```

5 Narzędzia steganograficzne

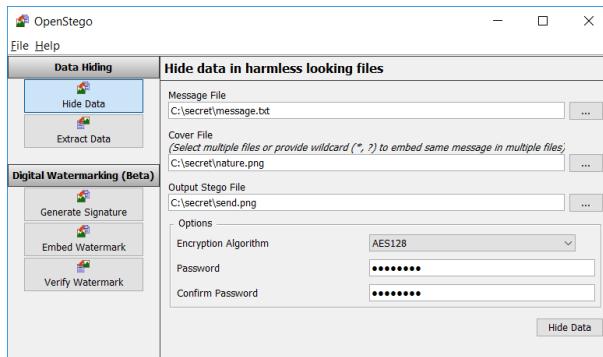
Wymienione narzędzia mogą zarówno ukrywać jak i wykrywać pliki, motadami jakie wykorzystuje dany program.

5.1 Outguess

Jest to oprogramowanie steganograficzne do ukrywania i wykrywania danych w obrazach. Metoda działania OutGuess opiera się na podziale obrazu na bloki pikseli, a następnie analizie każdego bloku w celu ukrycia lub wykrycia danych. Podobnie jak wyżej opisana steganografia LSB. W trakcie ukrywania danych przeprowadza analizę statyczną pikseli w blokach obrazu i określa czy blok jest odpowiednim miejscem na ukrycie danych. Jeśli tak OutGuess zamienia najmniej znaczące bity pikseli (LSB) w celu zakodowania ukrytych danych. Taka metoda powoduje, że ukryte dane są odporne na chi-square attack, który opiera się na analizie statystyk pierwszego rzędu.

5.2 OpenStego

Narzędzie do steganografii, które umożliwia ukrywanie danych w plikach multimedialnych. Wykorzystuje technikę LSB, czyli modyfikuje bity w sposób by nie zostały wykryte ludzkim okiem. OpenStego umożliwia szyfrowanie danych przed ich ukryciem co daje dodatkowe zabezpieczenie przed ich znalezieniem. Posiada również funkcję odkrycia zaszyfrowanych danych przez osoby do tego uprawnione.



6 Narzędzia steganoanalityczne

6.1 strings

strings to linuxowa komenda wypisująca łańcuchy znaków z danego pliku. Dzięki temu możemy pozyskać informacje o metadanych ukrytych np. w plikach wykonawczych

```
[roagen in ~/dev/rvtest λ strings a.out
riscv
rv32i2p0_m2p0
_boot
.symtab
.strtab
.shstrtab
.text
.data
.bss
.riscv.attributes
```

Rysunek 16: przykład użycia strings na pliku wykonawczym w architekturze rv32

6.2 binwalk

Metoda działania Binwalk polega na skanowaniu plików binarnych w poszukiwaniu charakterystycznych sygnatur dla różnych typów danych. Przy pomocy bazy jaką posiada może wykrywać dane takie jak pliki, obrazy, podpisów cyfrowych czy kodów wykonawczych. Program umożliwia także analizę entropii pliku, porównywanie różnic binarnych oraz rozpakować znalezione archiwa rekurencyjne.

DECIMAL	HEXADECIMAL	DESCRIPTION
0	0x0	JPEG image data, JFIF standard 1.01
59235	0xE763	PDF document, version: "1.4"
107288	0x1A318	Zlib compressed data, default compression

Rysunek 17: przykład użycia binwalk na pliku poliglocie jpg + pdf

6.3 StegExpose

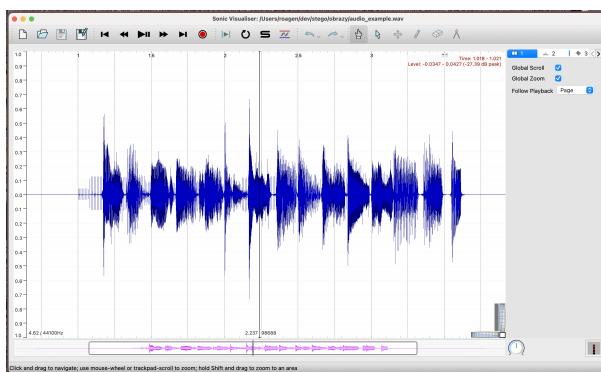
Narzędzie do wykrywania steganografii w obrazach, które jest wyspecjalizowane w wykrywaniu steganografii LSB w obrazach bezstratnych jakimi są PNG czy BMP. Steg wywodzi się z metod steganalizy opartych na pikselach takich jak Sample Pairs, RS Analysis, Chi-Square attack czy Primary Sets. Oprócz wykrywania obecności steganografii, StegExpose oferuje również steganoanalizę ilościową (określanie długości ukrytej wiadomości).

```
[root@openenclave ~]# ./StepExploit.py -o /root/stegeo -r /root/stegeo -t fast  
stegeo_6666458261_e45bd2625_b.png is suspicious. Approximate amount of hidden data is 114785 bytes.  
stegeo_6672108499_88c5827a79.png is suspicious. Approximate amount of hidden data is 137847 bytes.  
stegeo_6672542281_532f70bffe.png is suspicious. Approximate amount of hidden data is 67144 bytes.
```

Rysunek 18: przykład użycia StegExpose

6.4 Sonic Visualizer

Sonic Visualizer to program do wizualizacji plików dźwiękowych, który może być użyteczny do znajdowania podejrzanych szumów itp.



Rysunek 19: ilustracja wizualizacji sygnału z podejrzany szumem

7 Źródła

- https://pl.wikipedia.org/wiki/Steganografia_drukarkowa
- <https://royalprice.ru/pl/setting/steganografiya-i-stegoanaliz-obzor-sushchestvuyushchih-programm-i-algoritmov/>
- https://www.researchgate.net/figure/The-model-of-steganography-and-steganalysis_fig1_333772050
- <http://datagenetics.com/blog/september12015/index.html>
- <https://holloway.nz/steg/>
- [https://en.wikipedia.org/wiki/Polyglot_\(computing\)](https://en.wikipedia.org/wiki/Polyglot_(computing))
- <https://github.com/livz/cloacked-pixel>
- <https://github.com/b3dk7/StegExpose>
- https://link.springer.com/chapter/10.1007/11424826_54
- <https://github.com/fallais/tweg>
- <http://datagenetics.com/blog/september12015/index.html>
- <https://carlmastrangelo.com/blog/gamma-steganography>
- <https://pl.wikipedia.org/wiki/Mikrokropka>
- <https://pl.wikipedia.org/wiki/Steganografia>
- https://en.wikipedia.org/wiki/Chaffing_and_winning
- https://www.researchgate.net/figure/Chaff-and-wining-based-encryption_fig1_2360410
- <https://klinikadanych.pl/artykuly/steganologia-metody-ukrywania-informacji>
- <https://en.wikipedia.org/wiki/OutGuess>
- <https://github.com/syvaidya/openstego>
- <https://www.openstego.com/>
- <https://github.com/b3dk7/StegExpose>
- <https://www.youtube.com/watch?v=heprU4URpgc>
- <https://github.com/ReFirmLabs/binwalk>