UNIWERSYTET MARII CURIE-SKŁODOWSKIEJ W LUBLINIE
Wydział Fizyki Matematyki i Informatyki

Kierunek: **Informatyka**

Sebastian Dadej
nr albumu: 243584

Algorytmy tworzenia i wykrywania steganografii

Algorithms to create and detect steganography

Praca licencjacka
napisana w Zakładzie Technologii Informatycznych
pod kierunkiem dr Marcina Denkowskiego

Lublin rok 2014

1. Wstęp

Celem niniejszej pracy jest omówienie części współcześnie istniejących algorytmów mających zastosowanie przy tworzeniu i demaskowaniu ukrytego kanału komunikacji opartego o metody steganograficzne.

Zakres tematyczny obejmuje algorytmy steganograficzne mające zastosowanie przy wykorzystaniu, w roli nośników ukrytych danych, dokumentów tekstowych, zarówno drukowanych jak i cyfrowych, plików graficznych z kompresją bezstratną BMP, PNG, oraz stratną JPEG, a także steganografię w plikach dźwiękowych z kompresją bezstratną WAV – PCM. Zostaną także omówione metody steganoanalityczne oraz prostsze metody detekcji ukrytej zawartości.

W pierwszym rozdziale pracy przedstawię idee steganografii, które później rozwinę w rozdziale drugim omawiając najważniejsze fakty związane ze steganografią cyfrową. W rozdziale trzecim nakreślę charakterystykę nośników danych wykorzystywanych do osadzania danych. W rozdziale czwartym przedstawię podstawowe algorytmy i techniki steganograficzne z rozróżnieniem na te stosowane w plikach bez kompresji (z kompresją bezstratną) oraz te z kompresją stratną. W rozdziale piątym omówię sposoby detekcji ukrytych kanałów komunikacji oraz ataki na takie kanały.

1.1. Krótko o steganografii

Celem steganografii jest ukrywanie faktu komunikowania się. W ogólnej postaci sprowadza się to do ukrywania informacji w na pozór niewinnych nośnikach, takich jak tekst, czy plik cyfrowy. Słowo steganografia zostało zaczerpnięte z języka greckiego i znaczy dosłownie "ukryte pismo" od słów *steganos* - ukryty i *graphos* - pismo.

W celu ukrycia informacji poufnej wewnątrz innej informacji jawnej wykorzystywane są przeróżne metody. Kiedyś robiono to za pomocą tatuaży na głowie zakrywanych przez odrastające włosy, wyryte litery na drewnianych deskach pokrywanych woskiem i wypisaną na wosku mało znacząca wiadomością, czy stosując, tak jak to robiono za czasów 2 wojny światowej, mikrokropki lub atrament sympatyczny. Dzisiaj informacje ukrywane są m.in. w dźwięku poprzez modyfikowanie próbek cyfrowych lub w obrazie poprzez modyfikację pikseli. Nic nie stoi na przeszkodzie, aby poufne dane były chowane w przetransformowanej postaci nośnka, albo transportowane przy wykorzystaniu właściwości statystycznych nośnika. Dzisiaj najbardziej skomplikowane algorytmy są wstanie skutecznie rozproszyć informację w obrębie całego nośnika, odpowiednio dobrać informację nośną lub obszary dodawania danych, a nawet zaadaptować statystyki do przewidywanych wartości. Znane są skuteczne próby kodowania ukrytych informacji w DNA [18]. Dąży się do rozwinięcia coraz to nowszych i coraz to bardziej wyrafinowanych algorytmów ukrywania danych opartych np. o sztuczną inteligencję i algorytm genetyczny.

Zastosowanie metod steganograficznych precyzyjnie zobrazował Simmons wykorzystując w tym celu problem dwóch więźniów. Mamy dwóch więźniów: Alice i Boba. Ulokowani są w dwóch oddzielnych celach, bez możliwości przeprowadzenia

prywatnej rozmowy. Chcą wzajemnie uciec z wiezienia, jednak muszą wcześniej uzgodnić dokładny plan działania. W tym celu nie pozostaje im nic innego jak wykorzystać otwarty kanał komunikacyjny, którego rolę spełniają listy. Te jednak podlegają cenzurze. Strażnik Wendy cenzurujący listy nie dopuści by żadna zaszyfrowana wiadomość, lub wyglądająca na niebezpieczną została przekazana dalej. W tym przypadku, gdy mamy dostępny tylko jeden i to silnie monitorowany kanał komunikacji klasyczna kryptografia się załamuje, stając bezużyteczną. Pewne jest, że wiadomość musi zostać przemycona, aby sprawne oko strażnika nic nie wykryło. Tu swoją role odegra steganografia umożliwiając więźniom swobodną wymianę informacji [2].

2. Wstęp do Steganografii Cyfrowej

2.1. Najważniejsze fakty

Ciąg bitów stanowiący cyfrową reprezentacje wiadomości można osadzić praktycznie w dowolnym formacie plików, lub innym zbiorze danych binarnych, których przykładem może być nagłówek pakietu TCP/IP, jedynym warunkiem jest występowanie nadmiarowości w danych nośnych. Algorytm stosowany do osadzenia danych uzależniony jest od wybranej metody steganograficznej oraz wybranego nośnika, jednak w każdym przypadku jego filozofia działania jest podobna. Dokonuje on osadzenia poszczególnych bitów wiadomości w odpowiednich miejscach nośnika, wykorzystując istniejącą w nim nadmiarowość oraz charakterystyczne jego cechy; jednocześnie zachowuje pełną jego wewnętrzną integralność.

Steganografię komputerową ze względu na wykorzystywany algorytm i miejsce osadzania bitów wiadomości można podzielić na sześć grup:

1. Metody podstawieniowe – polegające na zastępowaniu nieistotnych/nadmiarowych elementów kontenera bitami tajnej wiadomości (np. BMP).
2. Metody transformacyjne – w których osadzenie informacji odbywa się w zmodyfikowanej przestrzeni sygnału poddanego odpowiedniej transformacji , np. do dziedziny częstotliwości (np. JPEG). Modyfikacji poddaje się uzyskane współczynniki transformaty.
3. Metody rozpraszania widma – polegające na rozszerzaniu widma sygnału. Stosowane w przypadku plików dźwiękowych (np. WAV, MP3).
4. Metody statystyczne – polegające na modyfikowaniu cech statystycznych kontenera umożliwiających odpowiednie rozmieszczenie bitów wiadomości,
5. Metody zniekształcające – działające poprzez wprowadzanie zniekształceń do sygnału kontenera. W celu odczytania osadzonych danych konieczne jest porównanie z oryginałem. Ich przykładem może być dodanie do cyfrowej reprezentacji dźwięku zawartej w kontenerze nieco białego szumu w którym umieszcza się bity tajnej wiadomości.
6. Generowanie nośnika – gdzie kontener jest tworzony specjalnie dla konkretnej wiadomości i do niej dostosowywany [5].

W celu silnego zabezpieczenia utworzonego kanału utajonej komunikacji należy odpowiednio dobrać zawartość kontenera, bowiem jego głównym przeznaczeniem jest maskowanie naszej wiadomości. Nawet jeśli dotychczas nasz kanał komunikacji nie został zdemaskowany, to jednorazowe zastosowanie kontenera z nieodpowiednią treścią może doprowadzić do niepotrzebnego wzbudzenia podejrzeń u osób trzecich. Jako przykład złego podejścia może posłużyć stosowanie w roli kontenera obrazów o jednolitych kolorach, bardzo niskim kontraście, czy małej różnorodności barw, a w przypadku plików audio spokojnych studyjnych nagrań o płaskiej linii melodycznej (posiadających bardzo płaską amplitudę dźwięku). W skrajnie pesymistycznym

przypadku dobór złego kontenera może doprowadzić do sytuacji w której drobna modyfikacja danych nośnych, nawet najbardziej wyrafinowanym algorytmem spowoduje powstanie albo widocznych gołym okiem artefaktów, albo słyszalnych szumów, a tego muśmy stanowczo unikać, dla tego należy stosować nośniki o jak najbardziej zróżnicowanej zawartości(sygnale). Przykładami takich kontenerów mogą być nagrania muzyki heavymetalowej, studyjne jak i na żywo, fotografie pozbawione licznych ostrych konturów, przedstawiające bardzo kolorowe i zróżnicowane kompozycje. Dzięki stosowaniu kontenerów o takiej charakterystyce szum jaki wprowadzamy osadzając bity danych bardzo mocno maskowany jest przez „szum" naturalnie występujący w nośniku, a tym samym mniej widoczny. Dodatkowo unikając modyfikacji istotnych elementów sygnału zawartego w kontenerze utrudniamy przeprowadzenie efektywnej steganoanalizy, zmniejszając prawdopodobieństwo wykrycia osadzonej informacji. Najdoskonalszym rozwiązaniem jest stosowanie kontenera, którego zawartość została stworzona, odpowiednio spreparowana na podstawie innego pliku, z intencją ukrycia w niej konkretnej porcji danych oraz nie istnieje, żadna jego kopia, oprócz tej posiadanej przez nas. Brak innych kopi zabezpiecza nasz kanał komunikacji przed atakami typu Know-Cover-Attack, czyli ataku polegającym na wykrywaniu różnic miedzy oryginalnym kontenerem, a kontenerem z osadzoną wiadomością. Należy pamiętać, aby przygotowany kontener był na tyle duży, aby możliwa była za jego pomocą sprawna wymiana danych. Dobrą praktyką jest jednorazowe wykorzystywanie kontenera, tak, aby niepotrzebnie nie wzbudzać podejrzeń oraz, aby ewentualni agresorzy nie mogli gromadzić materiałów, które później mogły by być wykorzystane do ataku typu Steganography-Only-Attack.

Bezpieczeństwo wiadomości w dużej mierze zależy do utrzymania w sekrecie faktu jej istnienia oraz sposobu jej ekstrakcji z nośnika, dla tego główną zasadą, którą kierują się cyfrowi steganografowie, jest modyfikowanie nośnika w jak najmniejszym stopniu, tak, aby pierwotna informacja w nim zawarta nie uległa zakłamaniu i aby dokonane zmiany nie były zauważalne. Osiąga się to poprzez modyfikowanie jedynie nieistotnych porcji danych, a także poprzez sprawienie, aby jak największa ilość odpowiednich bitów ukrywanych danych oraz bitów kontenera miały wstępnie tą samą wartość. Sprzyja to zmniejszeniu ilości bitów kontenera wymajających modyfikacji, a tym samym utrudnia przeprowadzanie ataków bazujących na analizie statystycznej oraz opóźnia podjęcie ataków steganoanalitycznych. Umożliwia to nam lepsze utrzymanie w tajemnicy faktu istnienia ukrytej zawartości i zmniejsza prawdopodobieństwo jej wykrycia [2].

W celu sprawdzenia jak dużych zmian dokona się podczas osadzania danych można wykorzystać *funkcję podobieństwa*.

„C" stanowi nie pusty zbiór, zaś funkcja *Pod*: $C^2 \rightarrow (-\infty, 1]$, będzie funkcją podobieństwa dla „C", to jeżeli dla x, y należącego do liczb całkowitych zachodzi:

Pod(x,y) = 1 ⇔ x=y

Np. dla x <> y Pod(x,y) < 1

Zaś w przypadku systemów steganograficznych:
Pod(C, E(m,c)) powinno dążyć do 1, dla każdego $c \in C \; i \; m \in M$,

Gdzie:

- E(m,c) to funkcja osadzającą wiadomość „m" w kontenerze „c",
- C to zbiór wszystkich możliwych kontenerów,
- M to zbiór wiadomości do osadzenia [5, 7].

Nie nadają się do wykorzystania algorytmy, które zbyt mocno ingerują w pierwotny kształt nośnika. Pomocnym w trafnym dobieraniu do siebie algorytmów i kontenerów jest opracowanie sposobu oceny poziomu dokonanych w nośniku, zauważalnych zmian. Rozwiązaniem może być organoleptyczna ocena nośnika lub poddanie go działaniu opracowanego przez nas samych algorytmu oceniającego niewidoczność zmian. Metoda ta pozwoli nam skutecznie analizować poziom bezpieczeństwa konstruowanego przez nas stegosystemu. Przykładowa implementacja metody dla stegosystemów opartych o obrazy cyfrowe poddawała by analizie poziom istotności elementów obrazu oraz określała widoczność wprowadzanych w poszczególne jego obszary zmian, w ten sposób możliwe było by wyznaczenie maksymalnej pojemności steganograficznej konkretnego kontenera. Metody tego typu usprawniły by opracowywanie nowych i dostosowywanie już istniejących algorytmów zabezpieczających przed wizualnym/słuchowym wykrywaniem wiadomości. Za przykład takiej metody stosowanej w przypadku obrazów może posłużyć znormalizowana funkcja czułości kontrastu CSF.

Naprawdę dobry i bezpieczny kanał utajonej komunikacji musi spełniać poniższe założenia, choć do utworzenia doraźnego kanału komunikacji, w przypadku, gdy nie mamy akurat pod ręką odpowiednich technik, ani narzędzi wystarczy spełnienie 2 pierwszych punktów:

1. Zawartość kontenera nie może być zmodyfikowana w stopniu wprowadzającym powstawanie artefaktów, czy zakłóceń dostrzegalnych nieuzbrojonym okiem, lub uchem.
2. Podczas osadzania bitów wiadomości nie wolno naruszyć struktury kontenera.
3. Dobrze jest gdy osadzona informacja jest odporna na uszkodzenia kontenera lub celowe przetwarzanie jego zawartości poprzez edycję, kompresję stratną, lub inne metody.
4. Dobrze jest gdy ukrytą wiadomość (a co najmniej jej cześć) można wyodrębnić nawet w przypadku zniszczenia bądź utraty części kontenera [7,21].

Dodatkowo przed rozpoczęciem komunikacji, obie strony muszą koniecznie uzgodnić ze sobą algorytm kodująco-dekodujący informację w nośniku, przypomina to trochę uzgadnianie klucza szyfrującego w kryptografii z tym, że w przypadku steganografii może być to czynność jednorazowa. Dla bezpieczeństwa kanału komunikacji sposób ekstrakcji wiadomości z nośnika należy utrzymać w sekrecie. Warto też unikać sekwencyjnego wbudowywania wiadomości w danych pliku stosując funkcję generującą losową kolejności wybieranych miejsc w których będą osadzane kolejne bity informacji, rozpraszając tym samym wiadomość w objętości całego nośnika. Można też stosować wprowadzanie szum w niewykorzystane obszary nośnika. Istnieją metody pozwalającej na wybieranie jedynie fragmentów obrazu/sygnału audio/innego kontenera najlepiej maskując dane, w których wprowadzone zmiany pozostaną najbardziej niewidoczne dla potencjalnego agresora. Takie działania utrudniają przeprowadzenie skutecznych ataków statystycznych i stegoanalitycznych.

Warto wiedzieć jak wyglądają ataki na stegosystemy i tak o to atak na ukryty kanał komunikacji może znaleźć się w trzech stanach:

1. Atakujący nie wie o istnieniu ukrytej informacji, bądź ma tylko delikatne domysły o możliwości jej istnienia, a ostatecznie nie wykrywa informacji. Jest to błąd typu II. Ukryty kanał komunikacji nie zostaje skompromitowany, a osadzane dane nadal są bezpieczne,

2. Atakujący wie o istnieniu ukrytej informacji i podejmuje próbę jej wyekstraktowania jednak bez skutku - popełnia błąd I typu – stegosystem jest bliski kompromitacji, chociaż informacja jest nadal bezpieczna,
3. Atakujący wykrywa osadzoną informację. Dochodzi do pełnej kompromitacji stegosystemu [5, 7].

Tworząc ukryty kanał komunikacji dąży się do zmaksymalizowania wystąpienia błędu II typu. Szansa na wykrycie wiadomości zależy od użytej techniki osadzania danych oraz od jakości i typu kontenera. Jeżeli zostanie wybrany nieodpowiedni sygnał, taki w którym łatwo jest zauważyć zanieczyszczenia wprowadzone przez funkcję steganograficzną to prawdopodobieństwo wykrycia znacznie wzrasta. Dysponując pewną bazą wiedzy o statystyce i charakterystyce danego typu sygnału można przeprowadzić atak statystyczny, przynoszący dobre efekty.

Tak jak to zostało ujęte w zasadzie Kerckhoffa [9], bezpieczeństwo dobrze skonstruowanego systemu kryptograficznego opiera się na kluczu, nawet w tedy gdy wszystkie szczegóły jego działania są znane, oczywiście oprócz klucza. Musimy mieć na uwadze utrzymywanie w tajemnicy algorytmu osadzania danych w nośniku, który w steganografii pełni rolę wspomnianego klucza. W związku z czym do zasadniczych wad ukrywania danych należy fakt, iż w momencie jednorazowej kompromitacji danego systemu steganograficznego jego bezpieczeństwo zostaje bezpowrotnie utracone, m.in. z powodu nasilenia czujności agresorów i ataków danego typu na dany kontener, a przede wszystkim przez to, że algorytm, czyli klucz staje się odtąd powszechnie znany. Optymizmem napawa fakt, że co rusz powstają nowe algorytmy steganograficzne.

Jako historyczny przykład kompromitacji stegosystemu może posłużyć tatuowanie wiadomości na ogolonej głowie, czy rycie wiadomości na drewnianych tablicach, które przestało być skuteczne w chwili kiedy większość ludzi dowiedziało się, że istnieją takie sposoby na utajone przenoszenie informacji. W razie potrzeby strażnicy uprzedzeni o takich metodach szmuglowania wiadomości mogli odpowiednio przeciwdziałać poprzez ścinanie włosów wszystkim ludziom przechodzącym przez punkt kontrolny lub topienie wosku z każdej odnalezionej tabliczki. W przypadku współczesnych technik steganograficznych sytuacja ma się podobne, z tym tylko wyjątkiem, że rolę takiego strażnika pełni dziś komputer.

W związku z powyższym bezpieczeństwo technik steganograficznych przedstawionych przeze mnie w tej pracy w wielu przypadkach jest już dzisiaj mocno wątpliwe, a ich główna siła w większości przypadków opiera się na istnieniu niezliczonej ilości danych mogących stanowić potencjalne kontenery przepływające codziennie przez publiczne kanały komunikacji takie jak Internet, czy częstotliwości radiowe oraz problemu z flitowaniem ich wszystkich jednocześnie, lub tez stosowaniu dodatkowych technik wzbogacających algorytm, np. stegokluczy. Optymizmem napawa także fakt, iż prawdopodobnie nikt z przeciętnych lub też bardziej zaawansowanych użytkowników jawnych kanałów komunikacji nie podejmie nagle próby doszukiwania się ukrytej zawartości w nagłówkach maili, obrazach, czy plikach dźwiękowych powszechnie dostępnych w Internecie, chyba, że będzie miał ku temu jawne przesłanki.

Podsumowując ten rozdział należy podkreślić, że w dużej mierze bezpieczeństwo tworzonego ukrytego kanału komunikacji opiera się na synergia kilku istotnych czynników, a nie tylko jednym najważniejszym, tak więc tworząc bezpieczny, odporny na infiltracje kanał komunikacji należy zadać o odpowiednią jakość ich wszystkich.

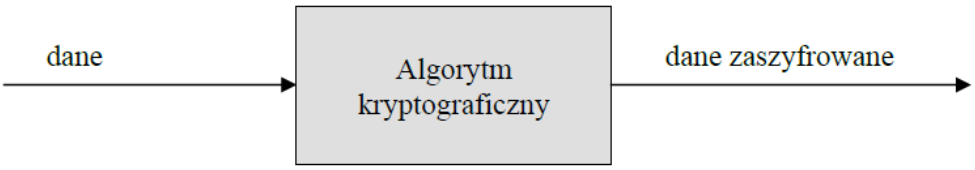
Chciałbym też zwrócić uwagę, że żadnym problemem jest prewencyjne, do pewnego stopnia skuteczne blokowanie ukrytych kanałów komunikacji tworzonych z

wykorzystaniem omawianych przeze mnie metod steganografii cyfrowej w wybranym obszarze Internetu, fal radiowych na określonym terenie, czy poprzez inne drogi jawnego transportu danych, jednakże wymaga to dysponowaniem odpowiednimi zasobami, wkładem pieniężnym oraz odpowiednim nakładem czasu. Tak więc, metody te same w sobie stanowią raczej namacalny przykład możliwych do zastosowania metod steganograficznych, a przedstawiając je miałem na celu zarysowanie schematu możliwości konstruowania takich metod.

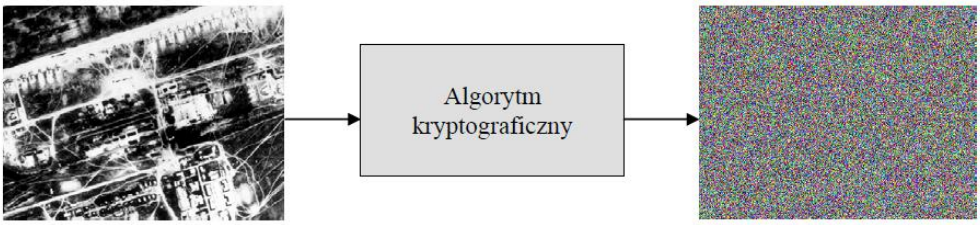
2.2. Połączenie steganografii z kryptografią

Zarówno steganografia jak i kryptografia są dziedzinami nauki zajmującymi się ochroną informacji. Jednak każda z nich działa w inny sposób i ma inne cele do osiągnięcia.

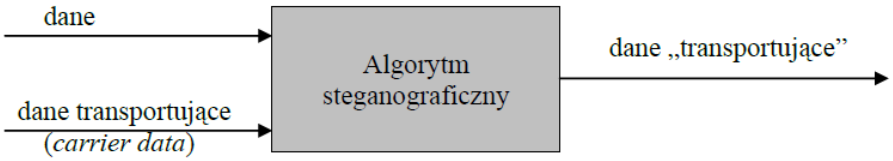
Kryptografia ma za zadanie uczynić wiadomość nieczytelną (rys. 1 i 2), a steganografia ukryć fakt istnienia wiadomości (rys. 3 i 4).



Rys. 1 – Sposób działania kryptografii [11]



Rys. 2 – Przykład szyfrowania bitmapy [11]



Rys. 3 – Sposób działania steganografii [11]

Rys. 4 – Przykład ukrywania bitmapy w bitmapie stanowiącej kontener [11]

Łącząc techniki tych dwóch dziedzin (Rys. 5), możemy zwiększyć stopień ochrony informacji. Informacja, która zostanie zaszyfrowana, a następnie ukryta jest o wiele bezpieczniejsza, niż gdyby była zabezpieczona jedynie za pomocą jednej z tych metod.

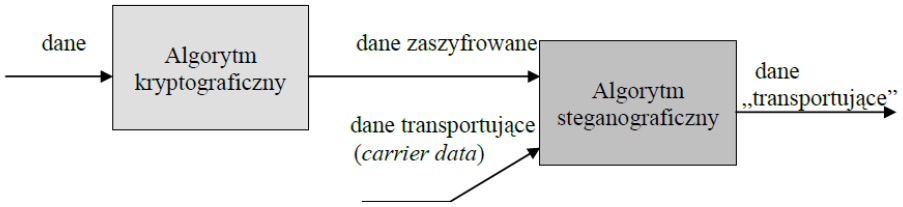
Kanał komunikacji w którym wiadomości są wstępnie szyfrowane jest niemal nie do wykrycia. Ktokolwiek próbował by go zdemaskować natrafia na trudność w stwierdzeniu, czy ta wiadomość tam rzeczywiście jest, czy jej nie ma. Może podejmować próby rozkodowania ciągu bitów zebranych ze zdjęcia/nagrania audio stanowiącego nośnik w takim kanale komunikacji nie będąc do końca pewnym, czy dany ciąg bitów tworzy, czy też nie jakiegokolwiek sensownego szyfrogramu, lub wręcz przeciwnie przeoczyć istniejący szyfrogram ukryty w innym zdjęciu, nośniku informacji.

Proces łamania szyfru jest procesem pracochłonnym, wymagającym ogromnych mocy obliczeniowej, w związku z czym ewentualny kryptolog/napastnik staje się bezradny w takiej sytuacji. Dopóki nie rozszyfruje wiadomości, dopóki nie uda mu się osiągnąć sensowej informacji z domniemanego szyfrogramu, sam do końca nie wie czy ta wiadomość jest tam ukryta czy nie.

Dzięki metodom stenograficznym jesteśmy w stanie ukryć fakt transmisji istotnych dla nas danych poprzez jakikolwiek jawny kanał komunikacji, maskując komunikaty we wnętrzu pliku, natomiast dzięki metodom szyfrującym, takim jak szyfry podstawieniowo – permutacyjne (RSA, AES) albo szyfry typu one-time-pad, uodpornimy nasz kanał komunikacji na inwigilację zapewniając nieczytelność informacji dla osób postronnych w razie podjęcia próby wydobycia jej z nośnika.

Ponadto szyfrowanie uodpornienia stegosystem na ataki bazujące na wykrywaniu charakterystycznych sygnatur znaków, ciągów bitów, zmierzające ku wykryciu bądź wydobyciu ukrytej wiadomości. Szyfrując tekst przed osadzeniem odpowiednio silnym algorytmem kryptograficznym powodujemy mniejsze zachwianie równowagi statystycznej kontenera, gdyż osadzane znaki w dużej mierze nie będą już znakami ze zbioru {a-z, A-Z, 0-9}, lecz ich wartość będzie równomiernie rozproszona. Dzieje się tak ponieważ algorytmy szyfrujące na ogół posiadają wysoki stopień losowości rozłożenia bitów, w wyniku czego do pliku wprowadzamy „naturalny" szum, a nie stanowiący pewien wzór ciąg bitów. Ponieważ stosowanie losowości i zmienności algorytmów jest sojusznikiem steganografa, warto nie poprzestawać na stosowaniu krypto-klucza, ale zastosować także i stegoklucz, który dodatkowo zwiększy dezorientacje atakującego i poprawi prawdopodobieństwo wystąpienia błędu II typu. System stanie się jeszcze bezpieczniejszy. Stegoklucz może być zarówno kluczem symetrycznym jak i niesymetrycznym. Będą od niego zależne pewne charakterystyczne

właściwości danego algorytmu, które pomogą nam skuteczniej umieszczać i rozpraszać bity wiadomości w nośniku. Użycie odpowiednio skonstruowanego stegoklucza gwarantuje bezpieczeństwo informacji, nawet wówczas gdy na jaw wyjdzie algorytm osadzania danych, przypomina to klucz wykorzystywany w kryptosystemach. Powszechnie znana jest bowiem budowa symetrycznego szyfru DES i asymetrycznego szyfru RSA, w Internecie bez trudu można znaleźć ich dokładną specyfikację wraz z kodami źródłowymi, nie umożliwia to jednak prawidłowego odszyfrowania danych bez znajomości odpowiedniego klucza [8].



Rys. 5 – Najskuteczniejsze rozwiązanie. Połączenie steganografii z kryptografią [11]

3. Nośniki danych

3.1. Przykładowe nośniki danych

Nośnikami wiadomości we współczesnych czasach mogą być dowolne pliki cyfrowe: pliki tekstowe wraz z zawartym w nim tekstem, pliki graficzne, pliki dźwiękowe, kontenery wideo, ale po za tym nagłówki pakietów TCP/IP, systemy plików, pole elektromagnetyczne oraz akustyczne, a nawet ludzkie ciało; istotne jest by nośniki te zawierały pewną dozę nadmiarowości.

Obecnie do najpopularniejszych kontenerów należą pliki multimedialne (obrazy cyfrowe, pliki audio, wideo), co wynika z faktu łatwego i szybkiego osadzania w nich danych oraz ich stosunkowo dużej pojemności steganograficznej wynikającej ze sporej nadmiarowości bitów, po za tym powszechnie występują w życiu codziennym każdego z nas. Dzień w dzień miliony różnych plików multimedialnych wędruje poprzez sieć komputerową (wymiana peer to peer, poczta elektroniczna, publiczne serwery, portale społecznościowe, itd.), które potencjalnie mogą stanowić nośnik danych. Wykrycie konkretnego pliku zawierającego ukryte dane wśród tak dużej liczby innych nośników jest bardzo trudne, lub wręcz nie możliwe - jeśli nie wiemy gdzie szukać. Przeanalizowanie takiej ilości danych zajęło by nam niewyobrażalna ilość czasu. Poza tym w wielu przypadkach nawet jeśli natrafimy na kontener zawierający z dużym prawdopodobieństwem ukrytą wiadomość i ostatecznie uda nam się ją odczytać, dość trudne może się dla nas okazać ustalenie nadawcy oraz adresata wiadomości. Wszystko to sprzyja utrzymaniu względnego bezpieczeństwa naszego kanału komunikacji.

Nadmiarowe dane gromadzone w nośnikach cyfrowych odzwierciedlają mało istotną, bądź wręcz nie istotną dla odbiorcy informacje, w związku z czym mogą być dowolnie modyfikowane bez wzbudzania podejrzeń, iż doszło do modyfikacji nośnika, gdyż dokonanie drobnych zmian jest w efekcie niezauważalne dla człowieka. Istotna cechą takich plików jest fakt, iż składają się z wielu niezależnych próbek, które można w prosty sposób modyfikować. Ponadto w ich przypadku stworzenie własnego nośnika jest bardzo łatwe, co daje pewność, że nikt nie będzie posiadał kopi oryginału pomocnego przy wykrywaniu dokonanych zmian.

Wraz z upływem czasu i rozwojem technologii cyfrowej powstały nowe pola do działania dla steganografii, zautomatyzował się proces ukrywania, wyodrębniania i przesyłania wiadomości, tak, że cały nakład pracy zmniejszył się do działań kilkuminutowych. Steganografia przeszła ewolucję. Z technik, które można nazwać analogowymi (tekst pisany) wkroczyła w świat technik cyfrowych i wszechobecnych bajtów. Cyfrowe nośniki informacji tworzą możliwości ukrywania informacji, a tym samym tworzenia ukrytych kanałów komunikacji. Dziś, to co kiedyś wymagało umiejętności, wyobraźni i czego nie należy ukrywać, spore ilości czasu, dziś za steganografów dokonują automatyczne programy jak zwykle wykonujące wszystkie nudne i żmudne czynności wymagające precyzji. Prawdopodobnie nawet głęboka wiedza o stosowanych metodach nie jest niezbędna, gdyż taka przydaje się jedynie przy tworzeniu narzędzi steganograficznych, a już do samego ich wykorzystania nie koniecznie.

3.2. Nadmiarowość w plikach.

Wszystkie pliki cyfrowe zawierające pewną dozę nadmiarowości mogą być wykorzystane do celów steganograficznych, jednak te zawierające największy odsetek nadmiarowości są najlepszymi kandydatami. Nadmiarowość jest pewną sferą w pliku, którą możemy dowolnie wykorzystać i modyfikować nie wprowadzając tym samym zbyt rażących i zauważalnych zmian w treści pliku.

Każdy plik cyfrowy jest kompozycją sekwencji cyfr binarnych (zer i jedynek). Prostym zadaniem jest modyfikowanie zawartości pliku poprzez zmianę pojedynczych bitów w sekwencji(ciągu znaków), jednakże realizacja modyfikacji bez wprowadzania rażących, dostrzegalnych przez użytkownika zmiany jest nieco bardziej wymagająca i wymaga od nas istotnej wiedzy o konstrukcji pliku i sposobie magazynowania w nim informacji. Zmiany dokonywane w nieodpowiednich miejscach pliku cyfrowego mogą doprowadzić do jego uszkodzenia, lub też do zdemaskowania tworzonego przez nas tajnego kanału komunikacji. W momencie, kiedy wiemy już które wartości w pliku możemy dowolnie modyfikować, musimy mieć na uwadze zakres liczbowy osiągany przez te wartości oraz współzależności między grupami wartości istniejących w pliku (taka sytuacja ma miejsce np. w plikach JPEG i nagłówkach TCP/IP).

Nadmiarowość przejawia się w subiektywnie nadmiernej ilości bitów wykorzystywanych na przechowywanie wartości. Wygląda to tak, że w momencie kiedy pozbywamy się nadmiarowych bitów wartość do złudzenia wyglądała na taką samą. Żeby dobrze to zobrazować przytoczę teraz przykład z odmierzaniem poziomu napięcia.

W chwili gdy wykorzystujemy tylko jeden bit do zapisu wartości próbkowanego poziomu napięcia jesteśmy w stanie prezentować jedynie dwa stany dla danego przedziału czasowego (jest napięcie: 1 albo nie ma napięcia: 0). Nie możliwe jest przedstawienie bardziej precyzyjnych wartości takich jak np. +3,3 V o ile napięcie nie jest napięciem granicznym i wysokość tego napięcia w danym momencie wynosi +3,3 V.

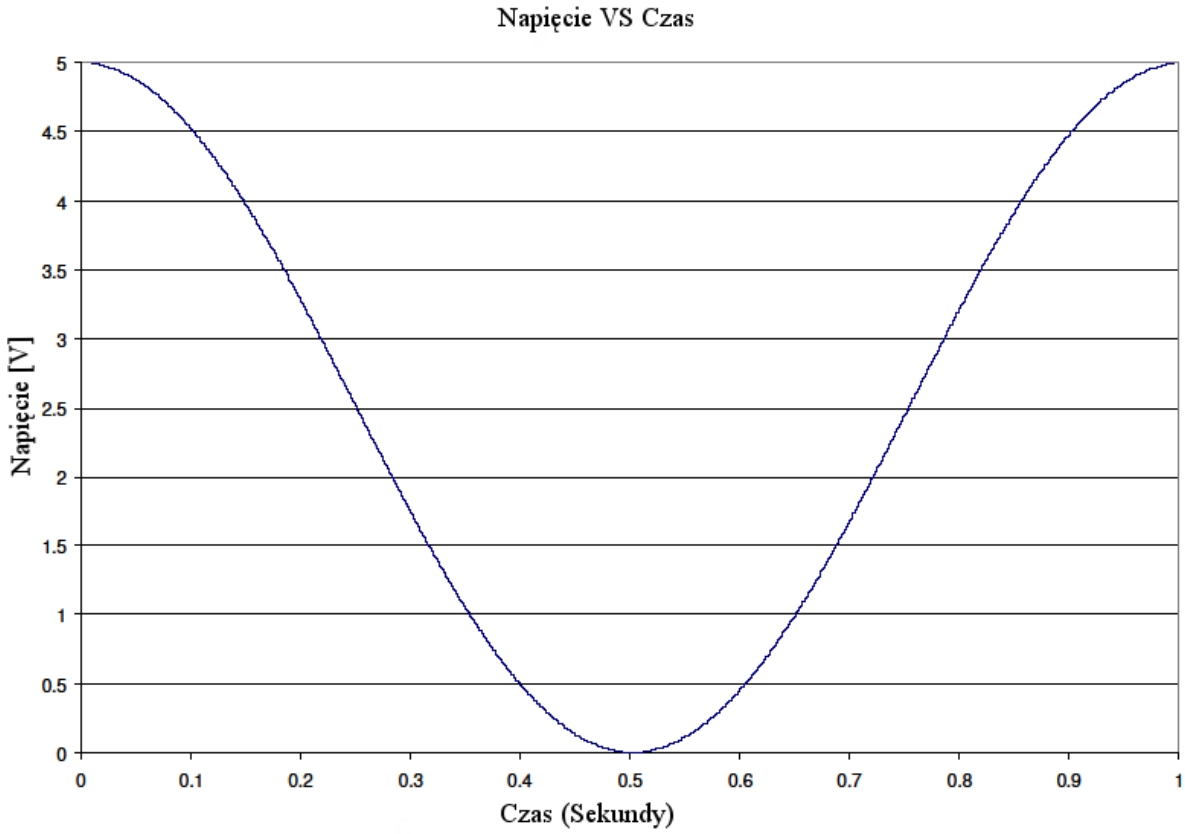
Po przez zwiększenie ilości bitów reprezentujących pomiar napięcia możemy zwiększyć czułość pomiaru. Co daje nam możliwość reprezentacji wysokości napięcia pomiędzy wartościami granicznymi (górną i dolną). Mając dwa bity jesteśmy w stanie zdefiniować do czterech stanów napięcia (np. 0; 1,1; 2,2 oraz 3,3 V), mając trzy bity możemy zdefiniować osiem stanów, cztery zdefiniować szesnaście i tak dalej. Poziom dokładności pomiaru jest proporcjonalny do ilości bitów wykorzystanych w dwójkowej reprezentacji poziomu napięcia. Wzrost ilości wykorzystywanych dodatkowych bitów na jedną jednostkę informacji uwidacznia się we wpływie na rozmiar oraz szczegółowość przechowywanych danych, które odpowiadają za reprezentowanie mierzonych wartości zmian napięcia.

W zależności od wymaganego poziomu wierności odzwierciedlenia dokonanych pomiarów dodatkowe bity w pewnym momencie mogą nie mieć już wpływu na pożądaną dokładność pomiarów reprezentując jedynie błędy zaokrąglania, a mimo wszystko znajdować się w pliku (np. 4.999999999999 vs 4.999999999998). W konsekwencji stają się zbędne ponieważ pewna subiektywna dokładność odwzorowania przebiegu zmian amplitudy napięcia zostaje już osiągnięta.

Kompromis pomiędzy rozmiarem pliku, a dokładnością próbkowania jest często spotykany, sama głębia bitowa (ilość bitów na próbkę) częstokroć wybierana jest na podstawie jakiejś rozsądnej średniej wyciągniętej ze wszystkich przypadków. Wybór optymalnej ilości bitów potrzebnych do reprezentowania informacji wykorzystując jak najmniejszą ilość miejsca w pamięci jest głównym celem wielu formatów danych.

Mając już pewną serię danych o napięciu próbkowanym co 1/25000 sekundy możemy przedstawić je graficznie na wykresie dwuwymiarowym, który wizualnie ukaże nam zmiany w napięciu zachodzące na przestrzeni czasu. Oś Y reprezentuje napięcie, zaś oś X upływ czasu (Rys. 6).

Przy zastosowaniu wystarczającej ilości bitów do zapisu danych możliwe jest osiągniecie lepszego przybliżenia aktualnie mierzonego sygnału dzięki czemu wierniej można odwzorować pomiar, a potem doskonalej odtworzyć sygnał. W przykładzie przedstawionym na rysunku 6, napięcie miedzy -5, a +5 Voltów jest reprezentowane za pomocą 8 bitów. Najbardziej znaczący bit (ten najbardziej po lewo) odpowiada za ustalenie znaku (+/-), natomiast pozostałe siedem bitów zapewniają 255 dyskretnych wartości dla wyznaczania amplitudy próbkowanego napięcia. W ten sposób każda dyskretna wartość wynosi 0,04 wolty.



Rys. 6 - Przykład zmiany napięcia zapasany w pliku nośnym [1].

4. Algorytmy osadzania danych

Przystąpię teraz do omówienie niektórych podstawowych algorytmów osadzania danych wraz z nośnikami do których się je stosuje.

4.1. Algorytmy tekstowe

Za każdym razem, czy jest to tekst, czy ciąg bitów pliku cyfrowego, modyfikacje w nośniku wykonywane są w według odpowiedniego wzoru wyznaczającego występowanie kolejnych składowych ukrywanej wiadomości, litery bądź, bitu.

W tym podrozdziale omówię algorytmy tekstowe, które dzisiaj są raczej zaszłością historyczną i archaizmem, niż praktycznie wykorzystywaną metodą. Algorytmy tekstowe stosowane zarówno w drukowanych dokumentach jak i tekstach elektronicznych mogą mieć niewielkie zastosowanie przy utworzeniu ukrytego kanału komunikacji na miarę XXI wieku i powszechnej cyfryzacji z powodu swej nikłej, wręcz bardzo małej pojemności steganograficznej, co miało by przełożenie na niską przepustowość utworzonego z ich pomocą kanału komunikacji.

Pierwsza metoda wykorzystująca tekst jako kontener polegała na odczytywaniu pojedynczych, określonych liter w tekście. Mogły to być litery oddalone od siebie o równą ilość znaków, bądź konkretne, któreś z kolei litery w każdym wierszu. Później pojawiły się takie metody jak Line-shift-coding, Word-shift-coding, Feature-coding i bardziej współczesne metody, metoda białych znaków i metoda syntaktyczna.

Do wad tekstu jako kontenera należy zaliczyć niewielką pojemność steganograficzną, prostą detekcję steganogramu oraz równie prosty i powszechnie znany algorytm odczytu osadzonych informacji, ponad to możliwość łatwego uszkodzenia informacja lub całkowitego jego zniszczenia.

Bazujące na tekstowych kontenerach algorytmy steganograficzne można podzielić na:

1. Algorytmy głownie stosowane w przypadku drukowanych dokumentów (Line-shift-coding, Word-shift-coding, Feature-coding, metoda semantyczna i syntaktyczna [7,21]).
2. Algorytmy stosowane w plikach cyfrowych (algorytm białych znaków).

4.1.1. Line-shift-coding

Algorytm ten polega na przesuwaniu o stałą wartość (np. 1/300 cala) wiersza tekstu w stosunku do wierszy nad i pod nim. Można go stosować zarówno do plików cyfrowych, jak i dokumentów drukowanych. Dokument cyfrowy z zakodowaną tą metodą informacją będzie ją zawierał nawet po wydrukowaniu. Możliwe jest poprawne wyodrębnienie osadzonej wiadomości nawet do 10 pokolenia przy dokonywaniu kopi dokumentu za pomocą fotografowania. Wadą tej metody jest wysokie

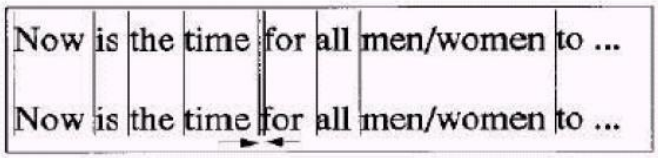
prawdopodobieństwo detekcji przez niepowołane osoby, gdyż nie są do tego celu wymagane specjalistyczne narzędzia. Przesunięciu ulegają jedynie wiersze nieparzyste, a parzyste pozostają na swoim miejscu. Bez tego brakowało by nam punktu odniesienia do wyliczenia przesunięcia. W Line-shift-coding możliwe jest oddanie reprezentacji dwóch stanów, np.: wiersz w górę to binarnie 1, a wiersz w dół to binarnie 0, w związku z czym możliwe jest zakodowanie dowolnej informacji. W teksie posiadającym 50 wierszy możliwe jest osadzenie 25 bitów, co jest bardzo niską pojemnością steganograficzną.

4.1.2. Word-shift-coding

W metodzie tej manipulujemy położeniem pojedynczych słów lub bloków słów w wierszu przesuwając je w lewo, lub w prawo. Wykorzystanie tej metody najlepiej udaje się ukryć w dokumentach w których występuję wyrównane do obu krawędzi strony, chociaż gdy tekst jest wyrównany inaczej zastosowanie tej metody także jest możliwe lecz może być bardziej widoczne.

Istnieją dwie odmiany tejże metody:

1. W każdym wierszu odnajduje się najmniejszy i największy odstęp pomiędzy słowami, następnie największy odstęp jest pomniejszana o jakąś stałą wartość, natomiast najmniejszy powiększany o taką samą wartość.
2. Wiersze w dokumencie dzielone są na trzy bloki słów. Przesunięciom ulegają jedynie blok środkowy względem bloku skrajnie prawego lub skrajnie lewego.



Rys. 7 - word-shift-coding i przesuniecie słowa "for" [7].

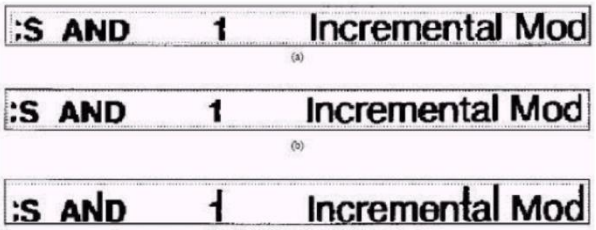
Dzięki temu, że możemy przesuwać słowa, albo w prawo, albo w lewo, możliwe jest zakodowanie dowolnej informacji wykorzystując w tym celu system binarny. Jednemu stanowi przyporządkowujemy 0, a drugiemu 1. Word-shift-coding to metoda odporniejsza na wykrycie niż Line-shift-coding, gdyż prawdopodobieństwo wykrycia przesunięcia pojedynczych słów lub bloków słów jest mniejsze, ponieważ dokonane zmiany są mniej widoczne. W przypadku tej metody wzrasta jednak ryzyko zniszczenia osadzonej informacji, gdyż już w przypadku drugiego pokolenia fotokopii dokumentów osadzona wiadomość zanika.

Odczytanie osadzonych informacji odbywa się na drodze różnicowania (wyłapywania różnic) steganogramu oraz oryginalnego dokumentu wykorzystanego w roli kontener. Taką konieczność można wyeliminować stosując dokładniejsze uzgodnienie stegoklucza, poprzez doprecyzowanie sposobu osadzania informacji. W tym przypadku można określić, które słowa lub bloki słów wolno przesunąć, a które nie [13]. Pojemność steganograficzna tej metody klasyfikuje się na poziomie 50 bitów na stronę tekstu A4. W każdej linijce teksu możliwe jest zakodowanie jednego bitu danych.

4.1.3. Feature-coding

Kodowanie to może przejawiać się na wiele sposobów, do najciekawszej i najpopularniejszej jej odmiany należy modyfikowanie wysokości liter: {b, d, f , h, k, l, t} występujących w tekście. Znowu mamy możliwość ustanowienia dwóch stanów gdzie np. litera o normalnej wysokości to binarne 0, a litera wydłużona to binarne 1.

Dla wzmocnienia bezpieczeństwa metody przed przystąpieniem do osadzania informacji dokonuje się losowego zmniejszania, bądź zwiększania o losową wartość liter wykorzystanych w późniejszym etapie do kodowania informacji. Utrudnia to atakującemu jednoznaczne określenie liter w stanie 0 i stanie 1. Ostatecznie informacja osadzana jest w tekście poprzez modyfikację wysokości wybranych (bądź wszystkich) liter: { b, d, f , h, k, l, t } poprzez ich zmniejszenie lub zwiększenie. W skutek tego, że drobna zmiana wysokości litery jest ignorowana przez ludzki mózg metoda Feature-coding zapewnia wysoki poziom skuteczności i bezpieczeństwa ukrywania danych. Jak i we wcześniejszym przykładzie tak i w tym odczytanie osadzonej informacji odbywa się na drodze różnicowania steganogramu i oryginalnego tekstu. Konieczność tą można wyeliminować poprzez zastosowanie stegoklucza jednoznacznie wskazującego na znaki wykorzystane w procesie osadzania lub zrezygnowanie z wstępnego losowego zmieniania wysokości liter.



Rys. 8 – Przykład zastosowania techniki feature-coding [7].

4.1.4. Metoda białych znaków

Metody tej nie można zastosować w drukowanych dokumentach, ale za to świetnie sprawdza się przy zestawieniu z cyfrowymi plikami tekstowymi. Sama w sobie może nieco przypominać metodę Word-shift-coding, gdyż w wyniku stosowania dodatkowych białych znaków słowa w tekście mogą ulegać przesunięciu.

Metoda ta polega na wykorzystaniu białych, niedrukowalnych znaków, poprzez umieszczenie ich w tekście w odpowiednich miejscach i w odpowiednim porządku. Stosując np. podwójną spacje na końcu wiersza można określić stan 1, a stosując jedną spację stan 0. Można też próbować wariacji z twardą spacją, lub innymi białymi znakami nie wprowadzającymi zbytniego chaosu do tekstu. Z powodu, iż białe znaki widoczne są dla człowieka jedynie wówczas, gdy zostaną umieszczone pomiędzy drukowalnymi znakami, lub słowami, lub gdy występują na początku wiersza (jeśli tekst jest wyrównany do lewej), lub na końcu wiersza (jeśli tekst jest wyrównany do prawej), należy unikać takich sytuacji.

Pojemność steganograficzną dla pliku tekstowego na którym zastosowano metodę białych znaków bazująca na kodowaniu z wykorzystaniem max. do 2 spacji na końcu wierszy jest niezwykle niska i wynosi dla przykładu 50 bitów dla dokumentu złożonego z 50 wierszy (wyrównanego do lewej).

Przykładowy wzór dla tej metody:

$$S = \frac{liczba\_wierszy}{8} \text{ [bajty]}$$

Zaznaczam, że możliwe jest wykorzystanie innego białego znaku, niewidocznych dla człowieka, w zamian za znak spacji oraz umieszczanie go w innym miejscu, niż na końcu linii, o ile nie zaburzy to formatowania tekstu. Metoda ta jest sztandarowym przykładem tego, iż stosunkowo prostoty algorytm ukrywania danych jest domeną stegosystemów obarczonych wieloma wadami, gdyż metoda białych znaków jest:

1. łatwo wykrywalna,
2. dość podatna na uszkodzenia,
3. nieefektywna pod względem pojemnościowym,
4. nieekonomiczna, rozmiar kontenera dramatycznie wzrasta w stosunku do ilości osadzanych danych.

4.1.5. Metoda syntaktyczna

Ostatnim algorytmem jaki omówię w kontekście ukrywania wiadomości w tekście jest metoda syntaktyczna. Siła tej metody opiera się na tendencji ludzi do popełniania błędów nie tylko życiowych, ale i tych mniej ważnych, ortograficznych, gramatycznych i interpunkcyjnych.

Istnieje kilka podejść do tej metody. Pierwsza z nich zakłada istnienie dwóch różnych wariantów gramatycznych/interpunkcyjnych równie poprawnych w swej budowie. Np.:

I) procesor, pamięć ram i dysk twardy,
II) procesor, pamięć ram, i dysk twardy.

W takim wypadku, gdy istnieją dwa stany, każdemu z nich możemy nadać wartość binarną, i tak konstruując stegosystem bazujący na interpunkcji możemy założyć, że: wyrażenie z przecinkiem przed spójnikiem będzie reprezentowało binarną 1, a wyrażenie pozbawione przecinka przed spójnikiem będzie reprezentowało binarne 0.

Osadzanie wiadomości polega na ustawianiu, bądź nie, przecinka przed spójnikiem „i” w odpowiednich miejscach kontenera.

Inne podejście bazuje na błędach ortograficznych i interpunkcyjnych. W takim przypadku sobie strony umawiają się jakie zasady ortograficzne i interpunkcyjne brane są pod uwagę. Np. Będą wykorzystywać fakt, że przed „że” należy pisać przecinek. Jeśli przecinek wystąpi będzie to oznaczało binarną 1, zaś jeśli nie wystąpi binarne 0.

Oba algorytmy steganograficzne bazujące na syntaktyce są dość bezpieczne. W praktyce występuje duże prawdopodobieństwo wystąpienia błędu II typu. Jedynie w drugim przypadku w razie zastosowania oprogramowania do automatycznej korekcji błędów nasza osadzona wiadomość rozpłynie się w niebycie, pierwsza metoda jest odporna na taki atak.

W przypadku tej metody steganograficznej, tak jak i w przypadku pozostałych czterech wcześniej omówionych, w żadnym razie nie można liczyć na wysoką pojemność steganogarficzną. Rozwiązaniem tego problemu może być preparowanie odpowiednich kontenerów pod dane osadzane, lub w przypadku wykorzystywania gotowych tekstów dobieranie odpowiednich fragmentów, lecz nawet taki zabieg nie pozwala osiągnąć zadowalających pojemności steganogarficznych.

Steganogramy utworzone metodą syntaktyczną można przekazywać zarówno droga elektroniczną jak i w formie drukowanej.

4.2. Algorytm binarny LSB

Przystąpię teraz do omówienia flagowego algorytmu steganografii cyfrowej - algorytmu podmiany najmniej znaczących bitów.

4.2.1. Idea działania

Idee algorytmu LSB przybliżę na podstawie moich wcześniejszych rozważań na temat nadmiarowości w plikach cyfrowych. Algorytm ten jest najprostszą metodą osadzania danych w nośniku i polega na podmienianiu wartości najmniej znaczących bitów – LSB (the least significant bit) pewnych wartości znajdujących się w pliku.

Gromadząc odczytywane co 1/25000 sekundy wartości napięcia w pliku oraz zapisując każdą wartość na 8 bitach, generujemy co jedną sekundę pomiarów 25 Kilobajtów danych, które zawierają pewną z góry założoną nadmiarowość. Naszym zadaniem jest modyfikowanie najmniej znaczącego bitu pierwszej warstwy (tj. pierwszego bitu liczonego od prawej) każdej z próbki wbudowując kolejne bity wiadomości w kształt fali bez wywierania znaczącego wpływu na graficzną reprezentację danych. Wiemy bowiem, że zmieniając ten bit w pesymistycznej sytuacji zmieniamy wartość liczby o +/-1, a tym samym wartość napięcia reprezentowana przez ta liczbę o +/- 0,04 V. Jest to wręcz niezauważalna zmiana, tak więc świadoma modyfikacja LSB każdej z próbek przechodzi niezauważona i w konsekwencji umożliwia wbudowanie pewnych danych.

Za pomocą sekwencyjnie występujących punktów danych w pliku jesteśmy w stanie przenieść naszą wiadomość wbudowując do 25 000 bitów wiadomości na każdą sekundę zebranych danych. W momencie kiedy podglądamy tak zmodyfikowaną falę, różnice w napięciu w każdym miejscu na fali są niedostrzegalne dla nieuzbrojonego oka, a o to przykład dobrze ilustrujący to zagadnienie:

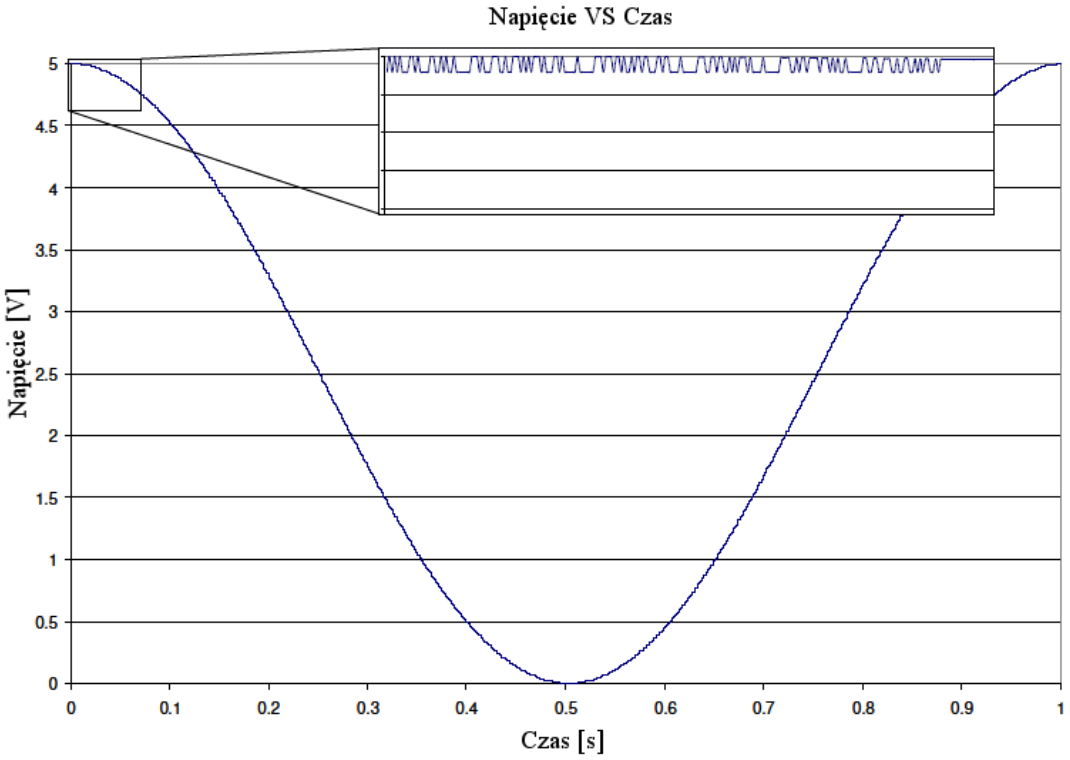
Mamy ciąg bitów odpowiadający kolejnym pomiarom napięcia:

01001010, 01001011, 01001100, 01001101, 01001110, 01001111, 01010000, 01010001 …

Chcemy wbudować 8 bitową wiadomość (11110000) w dane pliku, naszym zadaniem jest modyfikowanie odpowiedniego najmniej znaczącego bitu w ciągach bitów zaprezentowanych powyżej, tak aby pasowały do odpowiednich bitów naszej wiadomości. Wynikowy steganograficzny strumień danych mógłby wyglądać tak:

01001011, 01001011, 01001101, 01001110, 01001110, 01001110, 01010000, 01010000 …

Zmodyfikowane bity zostały pogrubione i podkreślone. Zauważmy, że w momencie gdy dane nośnika ulegają zmianie, to co jest przez nie reprezentowane lub tworzone na ich podstawie i prezentowane użytkownikowi, także ulega zmianie jednak w niezauważalnym stopniu. Rysunek 8 prezentuje nam nasz przykładowy przebieg fali z wbudowaną wiadomością zapisaną za pomocą kodu ASCII która brzmi: "**The truth shall set you free**". Fakt istnienia wbudowanej wiadomości może być tylko i wyłącznie zauważony w dostatecznie dużym powiększeniu pierwszych kilku próbek odtwarzających przebieg fali.



Rys. 9 – Przykładowa fala reprezentująca zmianę napięcia z wstrzykniętymi danymi [1]

Odtworzenie tak ukrytej wiadomości polega na odczycie odpowiednich bitów z nośnika, ułożenia ich w jedną całość i właściwą ich interpretację.

4.2.2. Wykorzystanie algorytmu w plikach graficznych - bitmapa

Teraz skupie się na zaprezentowaniu możliwości wykorzystania tego algorytmu w plikach graficznych niepoddawanych kompresji stratnej. Posłużę się w tym celu bitmapą.

Osadzanie wiadomości metodą substytucji LSB w bitmapach jest najprostszą metodą obrazowej steganografii cyfrowej. Rastrowy obraz w bitmapie przechowywany jest jako ciąg wartości kolejnych pikseli. W tym każdy kolor piksela reprezentowany jest przez odpowiednie natężenie trzech barw składowych (czerwonej, zielonej i niebieskiej). Naszym zadaniem będzie modyfikowanie najmniej znaczącego bitu każdej wartości reprezentującej barwę składową. Zakładając, że składowe zapisywane są na 8 bitach w pesymistycznym wariancie zmieniamy natężenie konkretnej składowej o

1/256, a koloru piksela o 3/16777216, co ze względu na ograniczenia ludzkiego oka, jest zmianą całkowicie nie zauważalną przez człowieka. Mówiąc inaczej, jednocyfrowa zmiana bar składowych koloru jest niezauważalna dla ludzkiego oka np. piksel o wartości (255, 0, 0) jest nie do odróżnienia od (254, 1, 1) (rys. 10).

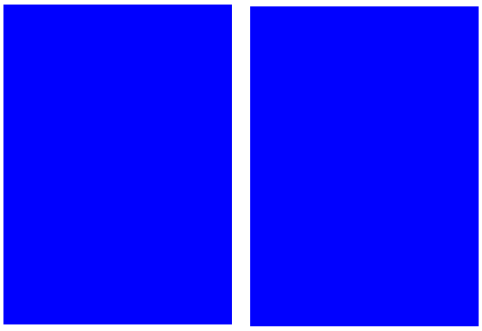
W tym rozdziale skupie się na bitmapach TrueColor, w których zapis informacji o każdym punkcie obrazu odbywa się na 24 bitach (po 8 bitów na składową), nic jednak nie stoi na przeszkodzie by podobne techniki wykorzystać do osadzania danych w bitmapach o głębi 16, czy 8 bitów.

Przykładowe wartości pikseli dla bitmapy 24 bitowej: (255, 0, 0) - 100% czerwony, (31, 187, 57) - ciemny zielony, (255, 255, 0) - czysty żółty.

Mieszając ze sobą barwy składowe osiągamy ogromną paletę barw, która jest dużo większa, niż zestaw rozpoznawanych przez człowieka kolorów, w związku z czym dysponujemy tutaj wymaganą dla technik steganogarficznych nadmiarowością danych. Wykorzystując ten fakt możemy dowolnie podmieniać bity z grupy najmniej znaczących bitów, na bity informacji, którą chcemy ukryć, wiedząc że nie dokonamy tym samym widocznych zmian w obrazie.

A o to przykład. Kolor niebieski reprezentowany jest przez trzy liczby dziesiętne: 255_{10} 0_{10} 0, co w zapisie binarnym daje nam: 11111111_2 00000000_2 00000000_2.

Modyfikując ostatni bit podmieniamy cyfry na końcu tych wartości na dowolnie wybrane przez nas. Np. 11111110_2 00000001_2 00000001_2.



Rys. 10 – Przykładowa zmiana koloru przy modyfikacji LSB we wszystkich trzech kanałach. Lewy (255, 0, 0); Prawy (254, 1, 1)

Rysunek 10 doskonale obrazuje fakt nierozróżniania przez ludzkie oko barw różniących się między sobą o 3 wartości oraz wpływ modyfikowania LSB we wszystkich 3 składowych.

Bitmapy 24 bitowe charakteryzują się sporą pojemnością steganograficzną, a poddając je bezstratnej kompresji PNG jesteśmy w stanie uzyskać jeszcze większy współczynnik upakowania poufnej wiadomości. Na przykład w bitmapie o wymiarach 1000 x 800 pikseli możemy stosując podstawowy wariant algorytmu LSB osadzić do 293 kilobajtów danych otrzymując tym samym współczynnik pojemności wynoszący: 12,5 %, zaś dla tego samego pliku poddanego kompresji bezstratnej: 26 %, co jest naprawdę godnym uwagi osiągnięciem.

Zaprezentuje teraz przykładowe osadzanie danych za pomocą podstawowego algorytmu LSB.

Pewnych kilka kolejnych pikseli obrazu opisanych jest za pomocą następującego ciągu bitów:
(RGB:) 00110110 11101001 01011000 (RGB:) 11000011 10101000 00011010

(RGB:) 00101011 11001011 11111111 (RGB:) 10101010 11111111 00000001
(RGB:) 01000001 11101001 11101100 (RGB:) 00101011 00000000 00111111

Chcemy w nim osadzić napis: „SD”, w który w kodzie ASCII ma reprezentację:
0101 0011 0100 0100

Osadzamy go w bitach obrazu w następujący sposób:

(RGB:) 00110110 11101001 01011000 (RGB:) 11000011 10101000 00011010
(RGB:) 00101011 11001011 11111111 (RGB:) 10101010 11111111 00000001
(RGB:) 01000001 11101001 11101100 (RGB:) 00101011 00000000 00111111

Wbudowane bity wiadomości zostały podkreślone. Dokonaliśmy zmiany wartości jedynie 5 bitów (zostały one pogrubione) co jest dobrym wynikiem. W najgorszym wypadku musielibyśmy zmodyfikować wszystkie 16 bitów.

Pojemność steganogarficzną bitmapy o 24 bitowej głębi kolorów przy stosowaniu metody substytucji LSB można policzyć za pomocą wzoru {1}:

$$S = W * H * 3$$

Gdzie:

W – to szerokość (width) bitmapy w pikselach,
H – to wysokość (hight) bitmapy w pikselach,
S – pojemność bitmapy w bitach,

Przykładowe obliczenia dla obrazu – bitmapy o wymiarach 1000 X 800 pikseli prezentują się następująco:

S = 1000 * 800 * 3;
S = 2400000 [bitów] = 300000 [bajtów] = 292,96875 [kB]

Z obliczeń wynika, że w tej bitmapie możemy umieścić do 293 kilobajtów danych bez znaczącego wpływu na jakość obrazu kontenera [5, 13].

Możliwe jest zwiększenie wydajności kontenera poprzez rozszerzenie algorytmu LSB oraz dokładniejsze wykorzystanie słabość ludzkiego oka przy rozpoznawaniu nieznacznie różniących się barw. Ludzki wzrok jest najbardziej podatny na zmiany koloru zielonego, a najmniej na zmiany koloru niebieskiego. W związku z czym, możemy dopuścić się do modyfikacji większej ilości młodszych bitów niebieskiej oraz czerwonej składowej koloru, tj. modyfikując dwa najmłodsze bity ze składowej czerwonej, jeden ze składowej zielonej i trzy bity ze składowej niebieskiej.

W takim wypadku, we fragmencie obrazu o postaci binarnej:
(RGB:) 00110110 11101001 01011000 (RGB:) 11000011 10101000 00011010
(RGB:) 00101011 11001011 11111111 (RGB:) 10101010 11111111 00000001
(RGB:) 01000001 11101001 11101100 (RGB:) 00101011 00000000 00111111

Napis „SD” (ASCII: 0101 0011 0100 0100) można zapisać w następujący sposób:

(RGB:) 001101**01** 1110100**0** 01011**1**00 (RGB:) 11000011 10101000 00011**10**0
(RGB:) 001010**01** 1100101**0** 1111111**0** (RGB:) 10101010 11111111 00000001
(RGB:) 01000001 11101001 11101100 (RGB:) 00101011 00000000 00111111

Bity, które uległy zmianę zostały pogrubione, wszystkie podkreślone bity stanowią kolejne bity osadzanej wiadomości. Metoda ta daje nam możliwość zapisania większej ilości danych w tym samym obrazie. Przy czym dokonane zmiany wciąż nie są zauważalne przez człowieka. W tym przypadku nowy wzór {2} wygląda tak:

$$S = W * H * (2 + 1 + 3)$$

Wynika z niego, że pojemność steganograficzna wzrosła dwukrotnie i wynosi obecnie: 25% całkowitej objętości pliku BMP oraz dla plików PNG 52%.

W każdym przypadku w kontenerze poza osadzoną wiadomością warto także odnotowywać rozmiar osadzonej wiadomości, w celu późniejszego uniknięcia odczytywania nic nieznaczących bitów oraz odnotować wynik stosowania funkcji skrótu (np. SHA-1) na wiadomości w celu umożliwienia późniejszego sprawdzenia poprawności odczytywanych danych. Najlepiej jest zapisać te dane na początku kontenera.

Tworząc kanał ukrytej komunikacji z wykorzystaniem bitmap należy pamiętać by był on wystarczająco duży by pomieścić wiadomość, (wzór {1} lub {2}) oraz o tym aby zawarty w nim obraz był dostatecznie dynamiczny. Należy unikać obrazów zawierających rozległe obszary o jednolitym kolorze oraz gradienty (płynne przejścia między odcieniami barw), gdyż w przypadku modyfikowania kilku bitów z pośród grupy najmłodszych bitów składowej może doprowadzić do powstania zaburzeń w płynności przejścia i wystąpienia widocznych artefaktów, pojedynczych pikseli o ciemniejszym, bądź jaśniejszym zabarwieniu. Warto przed osadzeniem wiadomości dokonać analizy statystycznej dynamiczności zmian barw składowych obrazu, co pozwoli na ewentualne wcześniejsze odrzucenie kontenera, bądź zmianę parametrów algorytmu osądzającego dane. Obrazy o dużej entropii oraz słabej korelacji barw umożliwiają lepsze maskowanie wiadomości oraz w niektórych przypadkach mogą osiągać ponadprzeciętne pojemności steganogarficzne [22].

Kanału komunikacji oparty na plikach graficznych i algorytmie LSB jest łatwy do utworzenia jednak nie gwarantuje odporności na różnego typu ataki, a zawłaszcza na ataki mające na celu zablokowanie przepływu informacji. Wystarczy dokonać na nośniku kompresji stratnej, albo przekształcenia obrazu (obrót, pochylanie, wyostrzanie, rozmycie, przycinanie, kadrowanie), aby w prosty sposób bezpowrotnie wymazać ukryte dane, lub znacząco utrudnić ich odczytanie. W wielu przypadkach wystarczy wyzerowanie wartości kilku bitów z grupy LSB lub celowe wprowadzenie w nie szumu. Sama detekcja ukrytej zawartości nie jest trudna. Dokonując analizy pliku możliwe jest w prosty sposób zdemaskowanie faktu jej istnienia. Trudności może przysporzyć próba wykrycia osadzonych danych wcześniej poddanych szyfrowaniu [2] oraz sytuacja w której bity wiadomości zostały rozproszone w ciele całego kontenera lub też do kontenera wprowadzono dodatkowy szum maskujący wiadomość (np. dokonano losowych zmian w niewykorzystanych najmniej znaczących bitach nośnika).

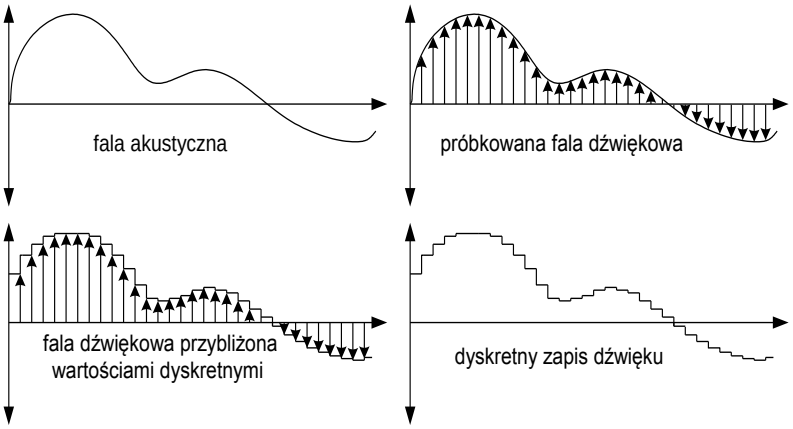
4.2.3. Wykorzystanie algorytmu w plikach graficznych - PNG

PNG to skompresowana algorytmem bezstratnym bitmapa. W jego przypadku możliwe jest zastosowanie wszystkich algorytmów, które mają swoje zastosowanie w przypadku bitmap, więc także i algorytmu substytucji LSB. Wymagana jest jedynie wstępna konwersja pliku do prostej bitmapy, którą potem poddaje się działaniu algorytmu steganograficznego, a następnie powrotnie konwertuje do formatu PNG.

4.2.4. Wykorzystanie algorytmu w plikach audio - WAV

Do skutecznego tworzenia kanałów ukrytej komunikacji wykorzystujących dźwięk, a więc i pliki audio, jako nośnik, konieczna jest wiedza z zakresu przetwarzania sygnału audio oraz jego przechowywania w postaci cyfrowej, a także świadomość tego w jaki sposób dźwięk interpretowany jest przez człowieka. Wiedza ta pozwoli nam na odtworzenie podstawowych, powszechnie znanych wykorzystujących cyfrowy dźwięk metod steganogarficznych, a także w przyszłości konstruowanie nowych, odporniejszych na „złamanie" algorytmów steganograficznych.

4.2.4.1. Cyfrowy zapis dźwięku



Rys. 11 - Próbkowanie fali dźwiękowej [www.google.com]

Dźwięk zapisywany w pliku cyfrowym jest dyskretną postacią analogowej (ciągłej) fali dźwiękowej rozchodzącej się w przestrzeni. Jego zapisu dokonuje się poprzez próbkowanie wartości w określonych odstępach czasu. Uzyskany w ten sposób ciąg próbek reprezentuje kolejne stany fali akustycznej. Przebieg procesu przekształcenia fali dźwiękowej na analogową zaprezentowany został na rysunku 11.

Im więcej próbek dźwięku jest rejestrowanych w jednej sekundzie tym doskonalej odwzorowuje się oryginalny, rejestrowany dźwięk oraz lepiej odwzorowuje wysokie tony (wysokie częstotliwości). Na jedną sekundę dźwięku może przypadać nawet 44 100 próbek dźwięku. Dzięki tak ogromnej ilości przechowywanych danych pliki audio, świetnie nadają się na kontener.

Dźwięk zapisany w postaci cyfrowej ma jeszcze jedną własność tak zwaną rozdzielczość próbki. Jest to ilość bitów przeznaczonych na zapis jednej próbki; im więcej jest to bitów tym dokładniej można odwzorować aktualny stan fali akustycznej.

Ważną cechą dźwięku w formie cyfrowej jest to, iż w przypadku zapisu próbek z dużą rozdzielczością (np. 16 bitów) bity najmniej znaczące najczęściej zawierają informację nie mającą znaczenia. Informacji tych albo ludzkie ucho nie jest w stanie wychwycić, albo są to jedynie nieistotne szumy powstające w skutek niedoskonałości urządzeń. Bity te stanowią tym samym doskonałe pole do manewru dla steganografów.

Dość często w pliku audio istnieją co najmniej dwie ścieżki dźwiękowe, po jeden na każdy głośnik. Uzyskuje się w ten sposób dźwięk stereo, który jest bardziej naturalny dla ludzkiego ucha niż dźwięk mono.

4.2.4.2. Ludzki słuch

Ludzki słuchu jest dość czuły na zakłócenia spowodowane przez biały szum AWGN (Additive White Gaussian Noise). Człowiek ma możliwość wykrycia białego szumu nawet gdy jest on o 70 db cichszy od poziomu głośności normalnego dźwięku. W wielu przypadkach akceptujemy występowanie takiego szumu. Może to wynikać z wieloletniego korzystania z niskiej jakości odbiorników radiowych lub telewizyjnych, gdzie występowanie szumów, trzasków lub innych zakłóceń było rzeczą normalną, a więc z czasem zostało uznane za normę i przestało drażnić.

Wiadomo także, że ludzki słuch jest mało czuły na zmianę siły tonu dźwięku. Przejawia się to w ten sposób, że nie jesteśmy w stanie dosłyszeć dźwięków o niższej głośności kiedy są one przykrywane przez głośniejsze dźwięki. Nazywa się to maskowaniem dźwięku. Możemy wyróżnić dwa rodzaje maskowania równoczesne i nierównoczesne.

Maskowanie równoczesne (zwane inaczej częstotliwościowym) występuje wówczas gdy dźwięk głośniejszy – maskujący posiada zbliżoną częstotliwość do dźwięku maskowanego – cichszego. Istnieje jednak pewien warunek konieczny do zaistnienia maskowania w takiej sytuacji. Dźwięk maskowany musi znajdować się poniżej progu maskowania, który wyznaczany jest na podstawie częstotliwości oraz charakteru tonu maskującego i maskowanego.

Natomiast maskowanie nierównoczesne (zwane także czasowym) polega na blokowaniu percepcji cichszego sygnału przez głośniejszy sygnał występujący do 200 ms przed, lub do 40 ms po maskowanym sygnale. Zasady tej nie stosuje się do dźwięków głośniejszych niż ton maskujący pojawiających się w tych ramach czasowych, gdyż te w przeciwieństwie do relatywnie cichszego tonu maskującego nie zostaną zignorowane przez zmysł słuchu.

Przykładem wykorzystania tej własności ludzkiego ucha jest stratna kompresja MP3, która bazuje na odpowiednio zmodyfikowanej dyskretnej transformacie kosinusowej i matematycznym modelu psycho-akustycznym określającym jakie informacje o dźwięku są rozpoznawalne przez ludzki zmysłu słuchu, a jakie nie. Przykładowo ludzkie ucho jest czułe jedynie na dźwięki o określonej częstotliwości. Zakres częstotliwości słyszalnego dźwięku szacuje się w przedziale od 20 Hz do 20 kHz z maksymalną czułością w przedziale 2 kHz - 4kHz. Zaś infradźwięki - dźwięki bardzo niskie i ultradźwięki – dźwięki bardzo wysokie są nierejestrowane przez ludzkie ucho.

4.2.4.3. Podstawy ukrywania danych w plikach audio

Wymienione przeze mnie niedoskonałości ludzkiego układu słuchowego oraz udowodniony fakt akceptowania przez ludzi pewnego stopnia szumów w sygnale audio są podstawą do konstruowania systemów steganograficznych bazujących na ukrywaniu danych w sygnale audio służąc do zamaskowania wprowadzonych zmian.

Osadzanie danych w plikach audio polega na dodaniu odpowiednio skonstruowanych zanieczyszczeń do przechowywanego przez niego sygnału w taki sposób, aby jednocześnie zawierały informację niejawną i nie obniżały znacząco jakości dźwięku. Czasami dla zwiększenia bezpieczeństwa i uniknięcia wprowadzania dodatkowych zniekształceń sygnału modyfikacje są wykonane w istniejącym już w utworze szumie. Przykładowym rozwiązaniem jest stosowanie odpowiednio zmodyfikowanych algorytmów usuwania szumu poprawiających jakość dźwięku i jednocześnie osądzających bity wiadomości.

Istnieje wiele sposobów osadzania ukrytej informacji w sygnale audio. Należą do nich m. in. wykorzystanie niewielkiego przesunięcia w czasie, podmiana najmniej znaczących bitów próbek, kodowanie informacji za pomocą odpowiednich przekształceń matematycznych, modyfikowanie istniejącego sygnału w taki sposób, aby możliwe było uzyskanie nowych własności możliwych do odczytania podczas późniejszej analizy cyfrowej, lecz jednocześnie niesłyszalne dla człowieka, itd.

Przykładem tej ostatniej metody może być metoda dodania echa, albo metody modyfikujące fazę do których zalicza się metodę wykorzystującą kodowanie fazowe oraz metodę wykorzystującą modulowanie fazy [7, 20]. Metoda dodania echa bazuje na utworzeniu odpowiedniego sygnału echa dla istniejącego utworu i dołączeniu go do oryginału. Wykorzystuje ona kolejną niedoskonałość ludzkiego słuchu, który nie potrafi wychwycić echa (lub po prostu go ignoruje) następującego po sygnale w czasie krótszym niż 0,02 sekundy. Informacja może zostać zakodowana za pomocą zmieniającej się w pewnym zakresie odległości echa od sygnału. Warto nadmienić, że metody kodowania fazowego, są zaliczane do metod skutecznych i trudnych do wykrycia.

Innym sposobem umieszczania ukrytych danych w sygnale audio jest dołączenie dodatkowych ścieżek w widmie częstotliwościowym. Ścieżki te mogą być umieszczone w zakresie częstotliwości niesłyszalnych dla człowieka lub też mogą imitować szum. Często jednak w wyniku kompresji lub też obróbki w celu poprawienia jakości dźwięku dane te zostają przypadkowo usunięte. Sposobem na to jest rozciągnięcie informacji w przestrzeni pasma dźwięku. Jest ona wówczas kodowana jednocześnie na wielu częstotliwościach [1].

Innym sposobem umieszczenia danych steganograficznych jest zastosowanie transformaty Fouriera, Falkowej lub kosinusowej do wstępnego przekształcenia oryginału. Następnie wprowadza się informację do współczynników transformaty i dokonuje się przekształcenia odwrotnego.

Techniki osadzania danych różnią się pomiędzy sobą sposobem wykonywania jak również efektami jakie udaje się uzyskać za ich pomocą (pojemność steganograficzna, poziom zniekształcenia sygnału, odporność na przekształcenia, w tym kompresje stratną) i tylko niektóre nadają się do utworzenie odpornego na infiltrację ukrytego kanału komunikacji, gdyż tylko część z nich zapewnia dostatecznie wysoki poziom trwałości osadzonych danych.

W celu wzmocnienia odporności ukrytych danych stosuje się przekształcenia odporne na modyfikację sygnału dźwiękowego w tym i kompresje stratną. Istnieją algorytmy pozwalające ukrywać informacje w takich obszarach dźwięku, że ich poziom

bezpieczeństwa na wykrycie lub uszkodzenie jest bardzo wysoki. Dodatkowo można też stosować zwielokrotnienie zapisu danych wraz z korekcją błędów.

Dzięki temu, że sygnał audio jest niezwykle plastycznym materiałem zapewniającym różnorodność parametryzacji – zmiany częstotliwości, przepływności, różnorodność styli muzycznych oraz swoiste jest dla niego, w pewnych przypadkach, występowanie zakłóceń i defektów, zapewnia nam możliwość bezpiecznego ukrycia danych. Natomiast dzięki sporej ilości próbek występujących na jedną jednostkę czasu, zapewnia nam dużą pojemność steganogarficzną.

Na koniec warto zwrócić uwagę, iż podczas osadzania bitów danych w strumieniu audio zawsze dokonujemy pewnych zmian parametrów dźwięku. Powinniśmy dążyć do tego by były to zmiany jak najmniejsze, lub jak najmniej zauważalne. Istotnym jest zachowanie prawidłowych wartości metryk nagrań, tak by nic nie wskazywało na istnienie dodatkowych danych/dokonanych modyfikacji nośnika, podczas ewentualnych ataków wykorzystujących analizę statystyczną pliku.

4.2.4.4. Algorytm LSB

Stosowanie algorytmu modyfikacji najmniej znaczących bitów w przypadku nieskompresowanych plików audio sprowadza się do modyfikowania jednego, bądź kilu najmniej znaczących bitów porcji danych wyznaczających wartość próbki. Próbki te przechowują wartości wypadkowej fali dźwiękowej rejestrowanej podczas zapisu dlatego metodę tą często nazywa się metodą modyfikacji amplitudy. Bity o których tu mowa albo w większości przypadków przenoszą szum kwantyzacji, albo ogólnie mało istotną informację, której modyfikacja w niewielki stopniu wpłynie na jakość sygnału audio. W zależności od wyboru ilości modyfikowanych najmniej znaczących bitów próbki, modyfikacja wartości tych bitów może nie mieć wpływu na zmianę parametrów dźwięku i zazwyczaj jest niesłyszalna dla ludzkiego ucha.

Podstawowa wersja algorytmu przewiduje modyfikowanie jednego najmniej znaczącego bitu próbki. Taka modyfikacja jest całkowicie niedostrzegalna przez ludzkie ucho o ile kontener nie zawiera fragmentów z całkowitą ciszą. Rozszerzona wersja tego algorytmu przewiduje modyfikowanie do 4 pierwszych najmniej znaczących bitów próbki lub modyfikowanie jedynie bitu 5 lub 6 wraz z zastosowaniem korekcji wartości próbki [7].

Próba zwiększenia ilości modyfikowanych najmniej znaczących bitów prawie zawsze kończy się wprowadzeniem zbyt silnych zakłóceń do nośnika audio, gdyż modyfikowanie większej ilości bitów niż 4 drastycznie zwiększa ilość występującego białego szumu AWGN.

Prawidłowością jest, iż wraz ze zwiększaniem ilości modyfikowanych bitów oraz wzrostem faktycznej procentowej zmiany ich oryginalnych wartości sygnał ulega coraz silniejszej modyfikacji, a na pewnym etapie zmiany stają się zauważalne i z pewnością mogą doprowadzić do zdemaskowania wiadomości. W dużej mierze zależy to od jakości sygnału audio stosowanego do maskowania obecności naszej wiadomości. W celu zwiększenia niewykrywalności wiadomości należy stosować pliki z nagraniem audio o naturalnie występującym zaszumieniu oraz licznych dźwiękach drugoplanowych i tła. Przykładem mogą być nagrania na żywo z koncertów. Nośniki takie minimalizują prawdopodobieństwo wychwycenia zakłóceń wprowadzanych przez nasze modyfikacje.

W zależności od ilości modyfikowanych bitów na próbkę umieszczanie informacji we wszystkich kolejnych próbkach może prowadzić do uzyskania dużej pojemności steganograficznej kosztem zwiększenia prawdopodobieństwa odczytania dołączonej informacji przez osoby trzecie oraz występowania zwiększonego poziomu słyszalnego szumu i odgłosów przypominających wyładowania elektryczne (trzaski). Aby tego uniknąć można, albo zmniejszyć ilość wykorzystywanych bitów na próbkę, albo do ukrycia informacji wykorzystać tylko niektóre próbki. Do ich wyznaczania można użyć odpowiedniego algorytmu, którego celem było by losowe wybieranie miejsca umieszczenia kolejnego bitu danych. Algorytm powinien jednocześnie uwzględniać parametry dźwięku w celu osiągnięcia jak najlepszych efektów przy jak najmniejszym zniekształceniu sygnału (przykładowy algorytm korekcji błędów) [1].

Metoda ta, ani w wersji podstawowej, ani rozszerzonej nie przejawia odporności na kompresję stratną, ani jakąkolwiek obróbkę dźwięku. Poza tym jej stosowanie w pewnych przypadkach doprowadza do powstania łatwo zauważalnych zmiany statystyk znormalizowanych nośników, co dodatkowo zwiększa szansę kompromitacji stegosystemu.

W przypadku metody LSB w danych o rozmiarze 10 MB przechowujących 1 minutę nagrania w jakości stereo o 16 bitowym rozmiarze próbki można przechować do 646 kB danych osadzonych, co zapewnia nam pojemność steganograficzną na poziomie 6,25%. Wykorzystując 2 najmłodsze bity ilość możliwych do osadzania danych możemy powiększyć dwukrotnie. W ten sposób bez większego ryzyka możemy wykorzystać do 4 najmłodszych bitów z każdej próbki, co pozwala podnieść nam pojemność nośnika do 25%.

Stopień modyfikacji nośnika w pesymistycznym przypadku wynosi tyle samo co pozyskana pojemność steganograficzna, a naszym celem, jak już wiadomo, jest modyfikowanie nośnika w jak najmniejszym stopniu. Stopień wprowadzonych zmian można zmniejszyć poprzez odpowiednie dopasowywanie nośnika. Polega ono na takim doborze nośnika, aby jego odpowiednie bity mające stanowić miejsce zapisu bitów wiadomości posiadały tą samą wartość, co bity wiadomości jeszcze przed osadzeniem w nim danych. Przy takim założeniu w optymistycznym przypadku jesteśmy w stanie osiągnąć zgodność rzędu 100%. W praktyce zdarza się to jednak wyłącznie w przypadku gdy nośnik został wcześniej specjalnie spreparowany dla konkretnej wiadomości [2, 20].

Przedstawię teraz działanie jednej ze zmodyfikowanych wersji algorytmu LSB, o którym wspomniałem na wstępie do tego podrozdziału. Algorytm ten umożliwia w praktyce wykorzystanie dowolnego z pierwszych 6 najmniej znaczących bitów, jednakże skupia się głownie na bicie 6. Umieszczenie bitu informacji właśnie na 6 bicie próbki ma na celu zwiększenie bezpieczeństwa danych. Uodparnia je na ewentualne próby odfiltrowywania białego szumu oraz zwiększa dezorientację atakującego, który w większości przypadków nie będzie spodziewam się takiego działania, a tym samym spowoduje wystąpienie u niego błędu II typu. Technika ta polega na podmianie wartości 6 bit na bit wiadomości, a potem korygowaniu wartości danej próbki.

Aby lepiej to zobrazować przytoczę przykład:
Próbka ma wartość:
$1100\ 1000\ 0010\ 1111_2 = 51247_{10}$

Chcemy na 6 najmłodszym z kolei bicie umieścić bit naszej wiadomości. Będzie to: 0.
$1100\ 1000\ 0000\ 1111_2 = 51215_{10}$
Tym samym zmniejszyliśmy wartość próbki o 32.

Aby zniwelować ta różnicę do minimum, 5 najmłodszy bit ustawimy na: 1.
$1100\ 1000\ 0001\ 1111_2 = 51231_{10}$

Teraz różnica między oryginałem, a wersją z ukrytym bitem danych różni się jedynie o 16. Taki stopień zmiany w niektórych nośnikach może, lecz nie musi, być już zauważalny dla tego po każdorazowym dokonaniu modyfikacji nośnika należy dokonać organoleptycznej oceny stopnia wprowadzanych zmian. W ogólności można założyć, że pomimo zapisywania bitów danych na wyższej warstwie najmniej znaczących bitów dźwięk zawarty w kontenerze w znaczącym stopniu nie traci na jakości ponieważ zmiany w próbkach są odpowiednio kompensowane i w najgorszym przypadku ich różnica wynosi 32 w stosunku do wartości z przed osadzania danych, a nie aż 64, gdybyśmy niedokonywani kompensacji [7].

Metody tej nie można stosować dla bitów starszych niż 6 ponieważ modyfikacje dokonane w ten sposób wprowadziły by zbytnie zakłócenia. Amplituda dźwięku zmieniała by się o 64 i więcej [7].

Ostatecznie plik audio formatu WAV można poddać bezstratnej kompresji przy pomocy kodeka FLAC, który może zapewnić zredukowanie rozmiaru pliku o 40% - 60% przy pełnym zachowaniu jego jakości [19].

4.3. Algorytm osadzania danych w dziedzinie częstotliwości.

4.3.1. JPEG jako przykład pliku z kompresja stratną

DCT (ang. Discreet Cosinus Transform) – dyskretna transformata kosinusowa, została wynaleziona w 1974 przez Nasira Ahmeda, T. Natarajana oraz K. R. Rao. DCT jest podobna do transformacji Fouriera w tym sensie, że wytwarza pewien rodzaj widma częstotliwości przestrzennej. Jest jedna z najpopularniejszych blokowych transformat danych i szczególnie popularna w stratnej kompresji danych [16].

DCT znalazła zastosowanie w stratnej kompresji obrazu. Jest wykorzystywana w takim formacie plików jak JPEG oraz przez wszystkie kodeki z rodziny MPEG poświęcone kompresji wideo. Pliki formatu JPEG są bardzo popularne w sieci komputerowej, w związku z czym stanowią doskonałe medium do utworzenia ukrytego kanału komunikacji.

Plik JPEG można kodować na różne sposoby, jednak najpowszechniejsze jest kodowanie z udziałem JFIF, który uściśla pewne aspekty kodowania i zapisu pliku niesprecyzowane przez standard JPEG [4, 17]. Proces kodowania i kompresji stratnej dzieli się na kilka etapów:

1. Konwertuje się obraz z kanałów RGB na $Y'C_BC_R$ zawierający jedną składową luminancji (Y’), reprezentującą jasność oraz 2 kanały chrominancji, reprezentujące kolory. (Ten krok jest czasami pomijany).

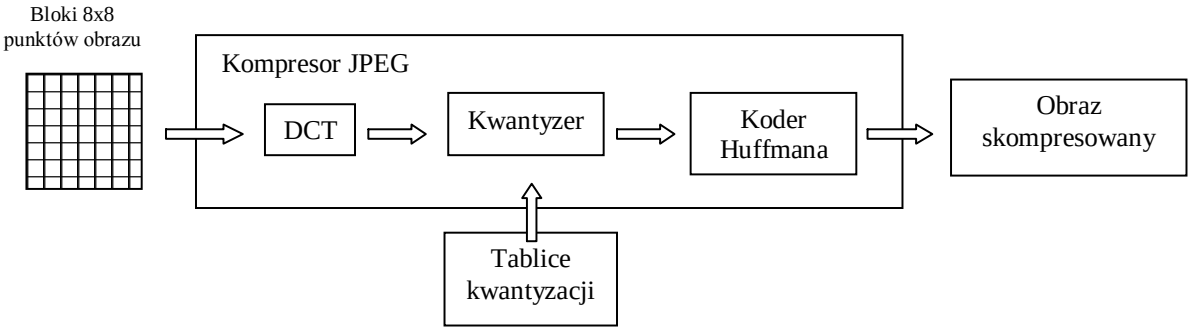
2. Jeśli dokona się konwersji z p. 1. rozdzielczość danych chromatycznych jest często redukowana poprzez dzielenie przez 2, albo 3. Wynika to z faktu, że ludzkie oko jest mniej wyczulone na drobne szczegóły kolorystyczne niż

drobne szczegóły w jasności. Są trzy współczynniki wyznaczające stopień kompresji występującym na tym etapie. Obraz bez redukcji to 4:4:4, obraz ze zredukowaną rozpiętością horyzontalną podzieloną przez dwa to 4:2:2, a obraz z rozpiętością horyzontalna i wertykalną podzielona przez 2 to 4:2:0.

3. Następnie każdy z kanałów (Y', C_B, C_R) dzielony jest osobno na bloki 8 x 8 pikseli (64 wartości w każdy bloku). Wartości w tak utworzonej macierzy które pryzmują wartości [0,255] rzutowane są na wartości z przedziału [-128,127], a następnie macierz poddawana jest Dyskretnej Transformacie Kosinusowej (Discrete Cosine Transform - DCT). Sprawiając, że poszczególne wartości nie reprezentują nasycenia w danym punkcie, a zmianę w nim zachodzącą.

4. Następnie dokonuje się kwantyzacji amplitud komponentów częstotliwościowych dla poszczególnych bloków z użyciem odpowiedniej predefiniowanej macierzy kwantyzacji, oraz ostatecznie zaokrągla liczby do pełnych wartości.

5. Tak pozyskane wynikowe dane dla wszystkich bloków 8 na 8 są następnie kompresowane za pomocą algorytmów kompresji bezstratnej - pewnego rodzaju kodowania Huffmana [4, 10, 17].



Rys. 12 – Schemat blokowy algorytmu JPEG

Poniżej prezentuje wzory stosowane podczas wyliczania DCT.

1. Dyskretna transformata kosinusowa:

$$G_{u,v} = \frac{1}{4} * \alpha(u) * \alpha(v) * \sum_{x=0}^{7}\sum_{y=0}^{7} g_{x,y} * \cos\left[\frac{(2x+1)u\pi}{16}\right] * \cos\left[\frac{(2y+1)v\pi}{16}\right]$$

Gdzie:

- „u" to indeks wiersza w macierzy wartości częstotliwości przestrzennej, pryzmuje wartości: $0 \leq u < 8$,
- „v" to indeks kolumny w macierzy wartości częstotliwości przestrzennej, przyjmuje wartości: $0 \leq v < 8$,

- $\alpha(u) = \begin{cases} \frac{1}{\sqrt{2}}, & gdy\ u = 0 \\ 1, & gdy\ u \neq 0 \end{cases}$ jest współczynnikiem skalowania, który normalizuje transformację ortonormalnych,
- $g_{x,y}$ to wartość piksela na współrzędnych (x, y),
- $G_{u,v}$ to WYNIK, czyli współczynnik DCT na współrzędnych (u, v). [4]

2. Kwantyzacja współczynników DCT dla elementów macierzy 8 na 8:

$$B_{j,k} = round\left(\frac{G_{j,k}}{Q_{j,k}}\right)\ dla\ j = 0,1,2,\dots,7; k = 0,1,2,\dots,7;$$

Gdzie:

- round() to funkcja zaokrąglająca wartość do części całkowitych,
- $G_{j,k}$ to wartość nieskwantyzowanego współczynnika DCT na współrzędnych (j, k),
- $Q_{j,k}$ to wartość z macierzy kwantyzacji na pozycji (j ,k),
- $B_{j,k}$ to WYNIK, czyli wartość kwantyzacji na pozycji (j, k). [4]

*Istnieją też przypadki, w których zamiast zaokrąglania liczby do najbliższej wartości stosuje się zaokrąglanie liczby do liczb całkowitych w dół:

$$B_{j,k} = \left\lfloor\frac{G_{j,k}}{Q_{j,k}}\right\rfloor\ dla\ j = 0,1,2,\dots,7; k = 0,1,2,\dots,7;$$

Proces dekodowania wykonuje wszystkie te kroki od tyłu, począwszy od kroku 5, z wyłączeniem kwantyzacji, która jest procesem nieodwracalnym.

Kompresja JPEG, jest jak wiadomo kompresją stratną, nawet przy doborze najbardziej liberalnych ustawień część informacji umieszczona w pierwotnym obrazie jest bezpowrotnie tracona. Głównym problemem kodowania z zastosowaniem transformat opartych na blokach DCT są gwałtowne skoki wartości na granicach zrekonstruowanych bloków (tzw. efekt blokowy). Uwidaczniają się one zwłaszcza przy bardzo rygorystycznych ustawieniach stopnia kompresji, prezentując się jako artefakty, nieciągłości kolorów na granicy dwóch odrębnie wyliczanych macierzy DCT. Są dowodem na znaczne pogorszenie jakości obrazu i utratę sporej porcji informacji o nim (Rys. 14).

Rys. 13 - Nieskompresowany obraz. Rozmiar: 3 010 654 bajtów

Rys. 14 - Obraz poddany bardzo silnej kompresji DCT. Rozmiar: 12 804 bajtów. Widoczne efekty blokowe.

Ludzkie oko świetnie radzi sobie z wychwytywaniem drobnych zmian na relatywnie dużym obszarze, jednak słabo radzi sobie z rozpoznawaniem dużej ilości zmian zachodzących na niewielkim obszarze. To pozwala nam na dość spore zredukowanie ilości informacji opisujących składniki o wysokiej częstotliwości zmian, ponieważ spadek jakości obrazu i tak nie będzie zauważalny. Dokonuje się tego poprzez proste dzielenie każdego komponentu (współczynnika) w domenie częstotliwościowej poprzez stałą wyznaczoną dla tego komponentu (element z tabeli kwantyzacji). Elementy w macierzy kwantyzacji odpowiadają za kontrolowanie stopnia kompresji. Z tą zależnością, że im większe przyjmują wartości, tym większa jest dokonywana kompresja. Ostatecznie dokonuje się zaokrąglania liczby do części całkowitych; proces ten nazywa się kwantyzacją. Akt zaokrąglania liczb jest jedyną operacją wprowadzającą utratę pewnej porcji informacji w całym procesie kompresji (innym niż subsampling składowych chromatycznych) o ile obliczenia DCT są dokonywane z dostatecznie wysoką precyzją. W skutek zaokrąglania wiele z wysokoczęstotliwościowych komponentów (współczynników) przyjmuje wartość zero, natomiast pozostałe stają się małymi dodatnimi, bądź ujemnymi liczbami, które potrzebują znacznie mniej bitów na zapis. I tu uwidacznia się zaleta stosowania transformaty DCT w kompresji.

Wartości znacznej części współczynników sprawdzana są do zera, w związku z czym zmniejszana jest ilość bitów potrzebnych do reprezentacji sygnału przy zachowaniu odpowiedniego poziomu wierności odtworzenia sygnału.

Dzięki DCT jesteśmy w stanie zredukować 24 bity potrzebne do opisania pojedynczego piksela do jedynie 0.5 na piksel przy akceptowalnej jakości, lub do 1 bitu przy dobrej jakości [10].

4.3.2. Omówienie algorytmu

Ukrycia poufnej wiadomości można dokonać także podczas wykonywania kompresji danych. Istnieją metody steganograficzne przystosowane do działania na danych poddawanych kompresji stratnej zarówno wśród plików graficznych jak i plików audio.

Przedstawię najprostszą odmianę algorytmu, który umożliwia osadzenie tajnej wiadomości zarówno w obrazach JPEG, jak i filmach poddanych kompresji MPEG, jest to modyfikacja metody LSB polegająca na umieszczaniu bitów wiadomości we współczynnikach transformaty poprzez podmianę najmniej znaczący bit współczynnika DCT na bit wbudowywanych danych. W przypadku tej metody należy pamiętać o starannym wyborze współczynników, tak aby nie dokonać zbyt drastycznych zmian w wyglądzie obrazu przejawiających się w powstawaniu widocznych granic miedzy obszarami oddzielnych kalkulacje DCT. Najczęściej dokonuje się zmian na współczynniku DC (direct current – prąd stały) oraz współczynnikach AC (alternate current – prąd zmienny), które mają wartość większą niż dwa [14,15].

Dla wzmocnienia bezpieczeństwa można osadzać bity wiadomości tylko w wybranych z pośród nadających się do tego współczynnikach. Ich wyboru można dokonywać za pomocą funkcji która na podstawie ziarna losowo wybiera współczynniki. Ziarno albo powinno być przekazywane wraz z nośnikiem informacji, albo być pewną znaną wartością dla obu stron komunikacji. Może być ono wyznaczane w pewien znany dla obu stron sposób. Np. być sumą długości i szerokości obrazu. Innym podejściem dającym w pewnym sensie zbliżone rezultaty jest wcześniejsze zaszyfrowanie danych za pomocą klucza kryptograficznego.

Każdy współczynnik reprezentuje zależność różnicy od wartości kwantyzacji wybranego kwadratu, zmiana najmniej znaczącego bitu któregokolwiek ze współczynników danej macierzy wpływa na zmianę wartości wejściowego kwadratu (wszystkich jego 64 pikseli) co przekłada się na 64 pomniejsze zmiany dla pojedynczego bloku i w wielu przypadkach ma wpływa na płynność przejścia kolorów miedzy blokami. Dopóki współczynników nie zmodyfikuje się na tyle, aby pojawiły się znaczące różnice w barwach pikseli leżących na granicy macierzy kwantyzacji wprowadzane modyfikacje mają niezauważalny wpływ na odtworzenie obrazu podczas jego podglądu.

Osadzanie bitów wiadomości jedynie we współczynnikach DC zwiększa prawdopodobieństwo nienaruszenia pierwotnego wyglądu obrazu. Do osadzania wiadomości wykorzystać można wszystkie trzy kanały (luminacji, i dwa chrominancji).

Metoda ta, tak samo jak metoda substytucji LSB w bitmapach jest nie odporna na ataki kompresją stratną. Wystarczy obraz zdekompresować poprzez konwersję do formatu BMP, lub PNG, a potem znowu poddać kompresji stratnej JPEG. Ponowne wyliczanie DCT, stosowanie kwantyzacji, w wyniku której przekształca się liczby rzeczywiste na całkowite oraz zastosowanie tabeli kwantyzacji nawet przy takich samych opcjach kompresji spowoduje utratę osadzonych danych w takim stopniu, że zrekonstruowanie wiadomości będzie niemożliwe lub jedynie szczątkowe. Jedynym wyjściem w takiej sytuacji jest przygotowanie kontenera z danymi, które przy

dekompresji i ponownej kompresji pozostaną nie zmienione, czyli spreparowanie nośnika, którego odpowiednie bity będą w 100% zgodne z bitami naszej wiadomości, jeszcze przed próbą osadzenia wiadomości w nośniku. Dokonuje się tego poprzez drobne zmiany w poszczególnych pikselach nieskompresowanej jeszcze bitmapy wchodzących w skład pojedynczej macierzy 8 x 8 w taki sposób, aby w wyniku poddania macierzy przekształceniu DCT wartość LSB bitu DC oraz odpowiednich bitów AC odpowiadały wartościom konkretnych bitów osadzanej wiadomości. Proces jest powtarzany aż do uzyskania zgodności wszystkich istotnych dla osadzania informacji bitów z bitami wiadomości. W celu przyspieszenia całego procesu do wyszukiwania odpowiednich układów pikseli stosuje się algorytm genetyczny. Jednakże i ta metoda ma swoje ograniczenia. Przy dekompresji i ponownej kompresji wraz z zastosowaniem innego niż pierwotnie stopnia kompresji obrazu (wybieranego z przedziału od 0 do 100, gdzie 100 to obraz o najlepszej jakości) metoda dostosowywania kontenera z wykorzystaniem algorytmu genetycznego może zupełnie nie zdać testu, gdyż w skutek ponownej kompresji osadzone dane zostaną utracone. Procentowa utrata danych zależna jest od wybranej siły kompresji [2].

Obraz wraz z osadzoną w nim wiadomością mogą ulec uszkodzeniu podczas ręcznej obróbki obrazu, modyfikacji jego parametrów (wysokości, szerokości, głębia koloru, itd.) lub poprzez manipulowanie stopniem kompresji. Żaden stegosystem oparty na obrazach nie jest w stanie zapewnić 100% bezpieczeństwa osadzonych danych przed uszkodzeniem we wszystkich możliwych przypadkach. Jednakże systemy najmniej podatne na zniszczenia opierane są na osadzaniu danych w najistotniejszych elementach nośnika, a zatem w przypadku obrazów algorytm osadzający dane w DCT świetnie się do tego nadaje.

Minimalną steganograficzną pojemność obrazu zapisanego w formacie JPG można policzyć ze wzoru:

$$S = \left\lceil \frac{W}{8} \right\rceil * \left\lceil \frac{H}{8} \right\rceil * 3$$

Gdzie:
S – to ilość bitów które można wykorzystać na osadzenie ukrytych danych,
W – to szerokość (width) obrazu,
H – to wysokość (hight) obrazu,

Dla obrazu o wymiarach 1000 x 800 pikseli obliczenia wyglądały by następująco:

$$S = \left\lceil \frac{1000}{8} \right\rceil * \left\lceil \frac{800}{8} \right\rceil * 3$$

S = 125 * 100 * 3 = 37500 [bitów] = 4687,5 [bajtów]

Tak więc minimalna pojemność sytuuje się na poziomie 1,9%. Różnice w poziomie pojemności JPEGa jako kontenera mogą wynikać z rozbieżności wśród możliwych do wykorzystania współczynników AC. Ponieważ pojemność tą można zwiększyć wykorzystując nie tylko współczynnik DC, ale i wybrane współczynniki AC, których użyteczna ilość może się różnić w zależności od doboru obrazu nośnego oraz ustawień kompresji.

W powyższym obrazku możliwe jest upakowanie do 4687,5 bajtów danych, nie naruszając przy tym wewnętrznej integralności pliku, jak i dokonania żadnych

widocznych gołym okiem zmian w prezentowanym przez niego obrazie. Jest to gwarantowana pojemność steganograficzna dla tego kontenera uzyskana przy wyłącznym wykorzystaniu współczynników DC. Wartość tą można podnieść nawet kilkudziesięciokrotnie wykorzystując dodatkowo współczynniki AC, w wyniku czego osiągalna pojemność steganograficzna znacząco wzrasta. W analizowanym przeze mnie teoretycznym przypadku klasyfikuje się ona na poziomie 130,9 kB, przy wykorzystaniu średnio 30 współczynników AC na macierz DCT. Plik poddany kompresji JPEG z jakością na poziomie 90% przyjmuje rozmiar 235 kB. Tak więc przy tych założeniach uzyskujemy pojemność rzędu 55,5%. Jednak na takim poziomie modyfikacji współczynników dokonane zmiany mogą znacząco wpłynąć na wygląd obrazu.

4.4. Wykorzystanie pustej przestrzeni dyskowej

Innym sposobem na przechowywanie ukrytych danych, który być może niema zbyt wielkiego zastosowania w tworzeniu dynamicznego ukrytego kanału komunikacji, ale zawsze może posłużyć jako pomocny, tymczasowy schowek dla danych, które chcieli byśmy ukryć przed światem, jest metoda zapisu danych w niewykorzystywanych przez system, albo ukrytych miejscach fizycznych nośników danych. Poprzez manipulowanie niewykorzystaną przestrzenią albo ukrytymi folderami nieuwzględnianymi przez system operacyjny możemy przechowywać dane pomiędzy plikami albo w jakiejkolwiek niewykorzystywanej przestrzeni systemu plików. Przykładem jest wykorzystywanie przestrzeni występującej pomiędzy prawidłowym końcem pliku, a początkiem następnego klastra. Ta przestrzeń w większości przypadków systemów plików pozostaje niewykorzystana w związku z czym świetnie nadaje się na zapis ukrytych danych. Istnieje jeden linuxowy system plików w którym nie występuje problem „wiszących ogonków” plików.

Ukryte foldery tworzy się w taki sposób, aby nie były odnotowywane w tablicy alokacji głównego systemu operacyjnego. Pliki przechowuje się w nich poprzez „widmową” strukturę folderów systemu operacyjnego zarządzaną przez nasze oprogramowanie. W celu zmniejszenia prawdopodobieństwa nadpisania przez system operacyjny istotnych dla nas danych wykorzystuje się przestrzeń dyskową mającą najmniejsze szanse na bycie wykorzystaną przez system operacyjny, bądź też wykorzystane przez nas samych sektory oznacza się jako sektory zepsute („bed sector”), albo niemożliwe do odczytania („unreadable”) w głównej tablicy alokacji systemu operacyjnego. Często także dokonuje się zwielokrotnienia zapis danych poprzez tworzenie kopi. Wszystko po to aby zredukować, albo całkowicie wyeliminować prawdopodobieństwo nadpisania ukrytych danych, ponieważ dane te są zaalokowane w przestrzeni dyskowej, która dla głównej tablicy alokacji systemu operacyjnego widnieje jako pusta przestrzeń i w każdym momencie mogą być narażone na nadpisanie.

5. Steganoanaliza – czyli blokowanie, detekcja i odzyskiwanie informacji

Znając metody steganoanalizy jesteśmy w stanie lepiej się przed nimi zabezpieczać, dla tego w tym rozdziale omówię potencjalne zagrożenia na jakie będzie narażony nasz kanał tajnej komunikacji.

5.1. Ataki steganoanalityczne

Steganoanaliza jest to nauka, która zajmuje się wykrywaniem ukrytych informacji w kanałach komunikacyjnych. Dzięki niej możemy zidentyfikować próby osadzania informacji w nośnikach. Steganoanalizę możemy podzielić na pasywną, której celem jest samo wykrycie istnienia kanału steganograficznego oraz aktywną, która stara się wyodrębnić, bądź zniszczyć zawartą w kanale steganograficznym informację. Próba odkrycia i oszacowania prawdopodobieństwa istnienia wiadomości w głównej mierze bazuje na wiedzy o wykorzystywanych algorytmach oraz nośnikach danych oraz przechwyconych nośnikach z osadzoną wiadomością. Brakuje jednak wiedzy o stanie kontenera z przed osadzenia w nim informacji.

Tworząc kanał komunikacji musimy być świadomi, że może być on narażony na przypadkowe uszkodzenia, albo celowe ataki mające na celu wyodrębnienie wiadomości, bądź jej zniszczenie.

Do ataków steganoanalitycznych zaliczamy:

1. *Steganography-only attack*:
   Atakujący dysponuje wyłącznie plikiem z ukryta wiadomością. Metoda cechująca się bardzo niską skutecznością ze względu na dużą ilość istniejących algorytmów steganograficznych oraz brakiem punktu odniesienia w postaci oryginalnego, niezmodyfikowanego obrazu. Metoda ta nie gwarantuje poprawności wykrycia czegokolwiek,
2. *Known cover attack*:
   Atakujący posiada plik z wbudowaną wiadomością oraz plik bez wbudowanej wiadomości (pierwotną wersję kontenera),
3. *Known message attack*:
   Atakujący jest w posiadaniu nośnika z ukrytą wiadomością oraz zna treść zapisanej w nośniku wiadomości – atak ma na celu zdemaskowanie algorytmu osadzającego,
4. *Chosen steganography attack*:
   Atakujący zna algorytm użyty do osadzenia danych oraz ostateczny plik z wbudowanymi danymi. 100% szansa skompromitowania ukrytego kanału komunikacji. Żadna wiadomość nie jest już bezpieczna o ile nie została zaszyfrowana hasłem kryptograficznym. Wykorzystanie tej techniki ma w większości na celu fałszowanie transmisji danych poprzez podszywanie się pod nadawcę.

5. *Chosen message attack*:
   Atakujący zna oryginalną wiadomość oraz pewną ilość algorytmów stosowanych do osadzania wiadomości, ale nic nie wie o nośniku, ani ostatecznym pliku, czyli nośniku wraz z wbudowaną informacją. Atak ten polega na porównywaniu samodzielnie przygotowanego pliku z wbudowaną wiadomością do przechwyconych plików z wbudowaną wiadomością w celu wykrycia wzorca do znanego algorytmu, bądź oszacowania prawdopodobieństwa istnienia wiadomości.
   Technika ataku przypominająca metodę łamania haseł brute-force.
6. *Known steganography attack* :
   Atakujący posiada wszystkie komponenty stegosystemu (oryginalną wiadomość/dane, oryginalny i zmodyfikowany nośnik wiadomości oraz algorytm). Stegosystem podatny na ten atak jest systemem słabym i nie powinien być dłużej wykorzystywany, gdyż nie zapewnia należytego bezpieczeństwa przekazywanych danych [8].

W przypadku poprawnego wykrycia ukrytej wiadomości można albo wyodrębnić wiadomość, albo ją usunąć w sposób nie wzbudzający podejrzeń zarówno u nadawcy jak i odbiorcy przekazu, albo też sfałszować wiadomość osądzając w miejsce wyodrębnionej wiadomości własną wiadomość.

W przypadku ataku *Steganography-only atack* celem analizy jest odkrycie faktu istnienia ukrytych danych. Bez pierwotnej wiedzy o sposobie ich kodowania, stanie kontenera sprzed osadzania danych, próba odzyskania tajnej zawartości, z wykorzystaniem tej metody o ile w ogóle było by to możliwe, zajęła by naprawdę sporą ilość czasu.

Natomiast w przypadku ataku *Known cover attack* cel steganograficznej analizy przesuwa się w stronę odzyskania osadzonej wiadomości poprzez zestawienie różnic pomiędzy dwoma plikami.

W momencie gdy poznajemy algorytm odpowiadający za umieszczanie informacji w nośniku czyli ataku *Chosen steganography attack*, możliwe jest odwrócenie procesu osadzania wiadomości, odzyskanie osadzonej wiadomości oraz bezproblemowe zastąpienie jej zupełnie inną, albo ostateczne zniszczenie.

System steganograficzny, którego steganoanaliza nie prowadzi do złamania nazywamy systemem bezwarunkowo bezpiecznym.

Do ataku na stegosystem może dojść na dwa sposób. Albo atakujący sprawdzając dużą ilość danych przypadkiem natrafi na kontener z danymi, co zdarza się niezmiernie rzadko i raczej jest mało prawdopodobne, albo atakujący miał podstawy sądzić, że przechwycony sygnał zawiera pewną ukrytą treść, np. na podstawie niepokojących szumów, czy zakłóceń w sygnale dźwiękowym, lub też istniejących niepokojących artefaktów na obrazie, co jest przypadkiem bardziej realnym.

Jeśli będzie miał miejsce ten drugi przypadek wówczas stegosystem można uznać za bliski kompromitacji, zaś całkowita kompromitacja systemu steganograficznego nastąpi wówczas, gdy atakujący system skutecznie wyodrębni dane oraz odszyfruje wcześniej zaszyfrowaną wiadomość jeśli ta została zaszyfrowana.

Podejmujący próbę zweryfikowania istnienia ukrytego kanału komunikacji najczęściej zaczynają od przeanalizowania odpowiednich części kontenera lub strumienia danych pod kątem wyszukania w nich jakiś anomalii, gdyż ukrywana informacja umieszczana jest zazwyczaj w miejscach, które nie są przeznaczone do przesyłania informacji lub w miejscu danych w pewnym sensie nadmiarowych.

Do dwóch głównych sposoby wykrywania anomalii należy:

1. Analiza części nośnika, których struktura jest znana i możliwe jest skontrolowanie oraz weryfikacja poprawności zawartych w nich danych. Ponieważ są one w pełni przewidywalne bądź też posiadają z góry zdefiniowane wartości wynikające z istniejących standardów lub stosowanych praktyk. Ważną czynnością jest kontrolowanie pojawiania się wartości nadmiarowych oraz elementów sygnalizujących pojawienie dodatkowych danych, które jeśli nie mają odzwierciedlenia w rzeczywistości powinny wzbudzić podejrzenie.
2. Analiza poszczególnych składowych kontenera pod kątem możliwości wystąpienia i ich poprawności dla danego systemu bądź protokołu. Takie podejście stosuje się do wartości które są ściśle określone lecz także do wartości pseudolosowych lub takich których histogram jest charakterystyczny.

5.2. Wykrywanie zawartości

Ze względu na sposób działania rozróżnia się kilka metod wykrywania ukrytej zawartości.

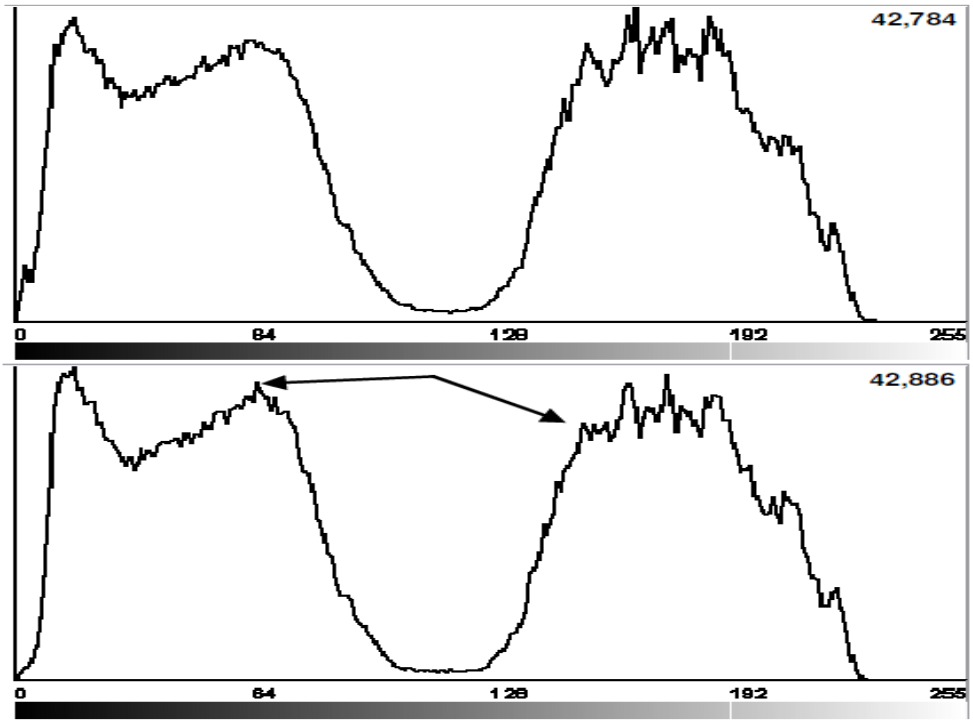
1. Steganoanalizę statystyczną - do tej grupy zalicza się metody wykorzystujące teorie informacji, regresję liniową, SVM (Support Vector Machines) i inne.
2. Steganoanalizę opartą na CI (Computational Intelligence) - do tej grupy zalicza się rozwiązania oparte o algorytmy genetyczne, sieci neuronowe i logikę rozmytą,
3. Rozwiązania hybrydowe - łączą w sobie funkcjonalności metod obu powyższych grup.

Najprostszym sposobem na wykrycie ukrytej zwartości jest porównanie oryginalnego nośnika z nośnikiem zawierającym osadzoną wiadomość (*Known-cover-attack*). Jest kilka metod na bazie których możemy wnioskować o prawdopodobnym istnieniu ukrytej zawartości.

Wyliczając sumę MD5, albo SHA-1 dla każdego pliku dowiemy się, czy pliki są identyczne, czy też nie. Podobne efekty osiągniemy porównując wyliczone statystyki dla każdego z plików, lub też porównując histogramy kolorów, lub amplitud dźwięku. Dane te mogą nam pomóc przy stwierdzeniu, czy badany nośnik zawiera potencjalnie ukrytą wiadomość, czy też powstałe różnice są wynikiem zwykłych przekształceń obrazu/sygnału audio.

Zestawiając ze sobą odpowiednie piksele jednego i drugiego obrazu, czy też obszary obrazu otrzymamy bardziej szczegółowe informacje na temat stopnia dokonanych zmian, które także mogą okazać się pomocne przy detekcji ukrytej informacji, a nawet przy próbę wydedukowania potencjalnie wykorzystanego algorytmu stagonograficznego. Podobna sytuacja ma się z plikami audio.

W przypadku wykorzystania histogramu barw pikseli dla obrazu (BMP, JPEG) zauważalne są różnice w rozkładzie i ilości kolorów. Często dla uproszczenia histogram taki tworzy się po uprzedniej konwersji obrazu do skali szarości, konwersji można dokonać w locie.



Rys. 15 - Przykładowe różnice jakie mogą wystąpić w histogramie [1]

Strzałki na histogramie (rys.15) pliku z wiadomością wskazują na dwie oczywiste różnice w przebiegu wykresu (wzrost ilości pikseli w granicach jasności 64 oraz zredukowanie ilości pikseli w okolicach poziomu 175).

W przypadku JPEG zamiast porównywania histogramów dokonuje się porównania odpowiednich wartości DCT oryginału i pliku z osadzoną wiadomością. W momencie analizy podejrzanego pliku warto zwrócić uwagę na zbyt duże różnice miedzy poszczególnymi macierzami DCT nieadekwatnymi do zastosowanych opcji kompresji, gdyż algorytmy które modyfikują kolejne współczynniki DCT w pliku JPEG wpływają na powstawanie niepłynności miedzy sąsiadującymi macierzami DCT [6].

Często jeśli dane nie są szyfrowane przed osadzeniem oraz zostają zapisane sekwencyjnie, bit po bicie, w kolejnych możliwych do podmieniania bitach kontenera, lub osadzenia bitów dokonano w jeden ze standardowych sposobów, możliwe jest podjęcie próby odnalezienia charakterystycznych ciągów bitów symbolizujących np. znaki doku ASCII. Inny prosty test, który można zastosować w przypadku obrazów JPEG polega na porównaniu oczekiwanego rozkładu współczynników DCT z wartościami próbkowymi [6]. W przypadku gdy mierzone wartości powodują odchylenia od przewidywanych wartości, wówczas amplituda odchylenia jest proporcjonalna do prawdopodobieństwa istnienia steganograficznej zawartości w danym miejscu w pliku.

Ponieważ wiadomość wcale nie musi być wbudowana w całą objętość nośnika, nośnika należy poddawać analizie kawałek po kawałku.

5.3. Niszczenie zawartości

W związku z tym, iż spora cześć algorytmów steganograficznych zarówno dla plików multimedialnych, jak i tekstu, jest powszechnie znana możliwe jest blokowanie

komunikacji dokonywanej za ich pomocą, poprzez prewencyjne, w wielu przypadkach skuteczne, usuwanie tajnej zawartości bez dogłębnego rozpoznania algorytmu wykorzystanego przy osadzaniu danych. Najprostszym podejściem jest poddawanie pliku pewnym transformacjom, lub pozbawianie go nadmiarowej porcji danych. Może to być konwersja do innego formatu (ponowne kodowanie), poddanie kompresji stratnej, poddanie korektom/filtracji itp. lub też wprowadzenie pewnej dozy szumu np. w najmniej znaczących bitach. Wadą tego rozwiązania jest to, iż możemy błędnie przewidzieć wykorzystane metody steganograficzne i w konsekwencji nie usunąć w pełni wiadomości z nośnika. 100% gwarancję na zablokowanie komunikacji poprzez utajony kanał komunikacji mamy jedynie wówczas, gdy w pełni i skutecznie rozpoznamy wykorzystywane algorytmy.

W zależności od zastosowanych technik steganograficznych oraz algorytmów zastosowanych do blokowania utworzonych z pomocą tych technik kanałów komunikacji okazać się może, że przygotowana przez nas metoda obrony nie była wystarczająco dobra. Nasz algorytm obronny usuną tylko niektóre bity ukrytej wiadomości, lub był całkowicie źle dobrany. Ponad to stegosystem przewidywał metody pozwalające odtworzyć uszkodzone fragmenty (zwielokrotnienie zapisu, dane naprawcze, kod Hamminga), lub osadzał dane w kontenerze wykorzystując kilka wzajemnie niewykluczających się algorytmów steganogarficznych przy czym przynajmniej jeden oparł się naszemu algorytmowi obronnemu. W takiej sytuacji możliwe jest odtworzenie teoretycznie wykasowanej wiadomości. Trywialnym przykładem błędu w zabezpieczeniach może być próba usunięcia osadzonych danych poprzez kompresję stratną nośnika, wówczas gdy algorytm osądzający dane przewidywał oporność danych na kompresje stratną.

Podczas, gdy mamy całkowita pewność co do stosowanych algorytmów steganogarficznych samo wykasowanie ukrytej zawartości jest rzeczą prostą. W większości przypadków sprowadza się to do zastąpienia odpowiednich bitów wartościami losowymi, lub ich wyzerowania.

Istnieją proste sposoby blokowania utajonej komunikacji prowadzonej poprzez steganograficzne algorytmy tekstowe, zarówno gdy mamy do czynienia z tekstem w postaci obrazu jak i zakodowanym przy pomocy któregokolwiek ze sposobów kodowania znaków (ASII, UTF-8, UNICODE, itd.). Jedynym wymaganiem jest przygotowanie odpowiedniego algorytmu, jak i wydajnej jednostki obliczeniowej.

Eliminacji ukrytej wiadomości z tekstu można dokonywać poprzez usuwanie niepotrzebnych, nadmiarowych białych znaków, automatyczną korekcję błędów oraz wprowadzanie w pewnym stopniu jednolitego formatowania tekstu dla wszystkich plików wpływających do zabezpieczanej sieci z zewnątrz, jak i z sieci wychodzących lub też konwersje do innego formatu pliku. Istnieją bowiem formaty plików tekstowych umożliwiające dowolne formatowanie teksu, a zatem mogące stać się potencjalnym medium do utworzenia kanału komunikacji. Za przykład mogą posłużyć pliki formatu PDF, lub DOC. Musimy być jednak przygotować na sytuacje, w której dokument tekstowy może być całkowicie pozbawiony swojego oryginalnego formatowania. Natomiast jeśli tekst dla utrudnienia detekcji ukrytej zawartości został zapisy w postaci obrazu, rozwiązaniem może być wykorzystanie oprogramowania odpowiedzialnego za rozpoznawanie tekstu - OCR, które pomoże odczytać tekst z obrazu i zapisać go w postaci dokumentu tekstowego o zdefiniowanym przez nas formatowaniu.

W dzisiejszych czasach wielu właścicieli i administratorów systemów i sieci komputerowych poszukuje sposobów na zakłócenie komunikacji prowadzonej poprzez kanały utajonej komunikacji tworzone przez zastosowanie metod steganograficznych. W takiej sytuacji prostym i skutecznym rozwiązaniem okazuje się stosowanie

wartowniczego procesora na wejściu oraz na wyjściu z sieci komunikacyjnej systemu. Do jego zadań należało by dokonywanie transformacji plików przesyłanych jawnymi kanałami komunikacji, zarówno tych wchodzących do sieci jak i wychodzących. Biorąc pod uwagę fakt, że często wykorzystywanymi plikami są pliki audio-wideo (obraz, muzyka, wideo) *wartownik* dokonywał by zmiany formatu pliku poprzez ponowne kodowanie w alternatywne formaty wykorzystując inny kodek oraz stosował by wszystkie inne technik omówione przeze mnie w tym podrozdziale.

6. Podsumowanie

Podsumowując moją pracę chciałbym zestawić ze sobą w pewien sposób uśrednione wartości możliwych do osiągnięcia pojemności steganograficznych w analizowanych przez mnie nośnikach danych: WAV, BMP, JPEG, dokument testowy, przy zastosowaniu adekwatnych omówionych przeze mnie algorytmów.

W zależności od przyjętych założeń wszystkie nośniki multimedialne osiągnąć mogą różną pojemność steganograficzną. Jednakże w najlepszym wypadku osiągają jednakowe wyniki umożliwiając osadzenie danych w ilości do 50% początkowego rozmiaru nośnika. W przypadku JPEG osiąga się to poprzez wykorzystanie większej ilości współczynników, kosztem pogorszenia jakości obrazu, natomiast w przypadku plików BMP oraz WAV dokonując kompresji bezstratnej odpowiednio do formatu PNG oraz FLAC.

Osiągalna maksymalna pojemność zawsze uzależniona jest od jakości wykorzystanego nośnika, im lepiej maskuje on zakłócenia wprowadzane przez nasz algorytm tym więcej możemy umieścić w nim danych.

Warto zaznaczyć, że nośnikiem charakteryzującym się największą chłonnością danych przy zachowaniu względnie niskiego zakłócenia pierwotnej jakości są obrazy BMP poddane bezstratnej kompresji. Wynika to z faktu, że format BMP posiada największą ze wszystkich zaprezentowanych przeze mnie nośników redundantność danych. Ponad to poziom 6 bitów wykorzystanych na każdy piksel obrazu nie jest wartością wyśrubowaną jak jest to w przypadku 31 bitów na jedną macierz DCT w JPEGach oraz 4 bitów w WAV o 16 bitowej próbce, a zatem dokonane zmiany są lepiej maskowane przez zawartość kontenera.

Ilość możliwych do osadzenia bitów jest odwrotnie proporcjonalna do odporności kontenera z danymi na wszelkiego rodzaju ataki, nawet organoleptyczne. Im mniej chcemy osadzić danych tym łatwiej jest je ukryć, a im więcej chcemy ich ukryć tym jest to trudniejsze, zaś zmiany coraz bardziej widoczne. Jako przykład może posłużyć algorytm wykorzystujący transformatę Falkową w plikach MP3, charakteryzujący się wysoką odpornością na kompresję i transformację nośnika, który umożliwia umieszczenie w nagraniu MP3 o przeciętnej długości raptem kilkudziesięciu bajtów danych [20].

Poddając zaś analizie sytuacje, w której za priorytet przyjęlibyśmy skuteczność ukrycia danych w nośniku na czele rankingu znowu sytuuje się BMP osiągający pojemność rzędu 12,5 % bez stosowania kompresji bezstratnej, zaś przy kompresji bezstratnej (PNG) 26%. WAV to 6,25% bez kompresji oraz mniej więcej 12,5% przy kompresji bezstratnej FLAC, a JPEG to jedyne 1,9%. Natomiast dokumenty testowe, w jednym jaki i drugim przypadku lokują się na końcu stawki, z najgorszym możliwym do osiągnięcia wynikiem. W dokumencie tekstowym zbliżonym do tej pracy dyplomowej, zawierającym 40 strony tekstu po 80 linijek na każdej, możliwe jest ukrycie mniej więcej 1600 bitów danych, co daje nam pojemność dla tego pliku rzędu 0,052 % w przypadku pliku DOCX oraz 0,103 % w przypadku pliku PDF, co jest najgorszym wynikiem ze wszystkich.

Tak jak wspomniałem we wstępie, bezpieczeństwo technik steganograficznych przedstawionych przeze mnie w tej pracy w wielu przypadkach jest już dzisiaj mocno

wątpliwe, a ich główna siła w większości opiera się na istnieniu niezliczonej ilości danych mogących stanowić potencjalne kontenery przepływające codziennie przez Internet oraz problemie flitowania ich wszystkich jednocześnie. Nieśmiało uwidacznia się w tym momencie mocna strona cyfrowych technik steganogarficznych. Potencjalny agresor chcąc odczytać wiadomość oraz ustalić jej nadawcę i adresata zmuszony jest poświęcić ogromną ilość czasu na przeanalizowanie wszystkich dostępnych w sieci lub danym obszarze sieci, nośników, co jest trudne, czasochłonne, bądź wręcz nie możliwe do osiągnięcia w realnym czasie. Takie działanie znacząco odsuwa w czasie fakt odczytania wiadomości i ma takie samo znaczenie i przeznaczenie jak szyfr w kryptografii, który aby złamać należy przeszukać cały zbiór potencjalnie poprawnych, pasujących kluczy. Tak więc obie te metody mogą wydać się ideologicznie do siebie zbliżone, z tą jednak różnicą, że informacja przesyłana w obrazie nie wzbudza takich podejrzeń jak szyfrogram.

Bibliografia

[1] Dickman Shawn D., *„An Overview of Steganography”*, James Madison University Infosec Techreport, 2007 r..
[2] Shih Y. Frank, *„Digital Watermarking and Steganography – Fundamentals and techniques”*, CRC Press, 2008 r..
[3] Provos Niels, Honeyman Peter, *„Hide and Seek: An Introduction to Steganography”*, University of Michigan, 2003 r..
[4] Adam Drozdek, *„Wprowadzenie do kompresji danych”*, Wydawnictwa Naukowo Techniczne, 2007 r..
[5] Katzenbeisser Stefan, Petitcolas A. P. Fabien, *„Information Hiding – techiques for steganography and digital watermarking”*, wydawnictwo Artech House, Boston 1999.
[6] Westfeld Andreas, Pfitzmann Andreas, *„Attacks on Steganographic Systems”*, Dresden University of Technology, 2000 r..
[7] Tomaszewski Marcin, *„Analiza algorytmów ukrywania w dźwięku”*, Politechnika Białostocka, 2006 r..
[8] Tomaszewski Marcin, *„Steganografia – tajniki ukrywania informacji. Grafika”*, http://software.com.pl/steganografia-%E2%80%93-tajniki-ukrywania-informacji-grafika/, [28.10.2014].
[9] Blackledge Jonathan, *„Cryptography and Steganography: New Algoritms and Applications”*, Dublin Institute of Technology, 2011.
[10] *„Kompresja obrazów cyfrowych wg standardu JPEG”*, Akademia Górniczo-Hutnicza w Krakowie, http://home.agh.edu.pl/~turcza/dsp/JPEG_pt.DOC, [28.10.2014].
[11] Drapała Jarosław, *„Zastosowanie technik steganograficznych do szyfrowania informacji”*, 2006, http://users.finemedia.pl/dloogie/bezpieczenstwo/steganografia.pdf, [14.11.2014].
[12] Johnson N.F., Jajodia S., *„Exploring Steganography: Seeing the Unseen”*, wyd. 31, nr. 2/1998, http://www.jjtc.com/pub/r2026.pdf, [28.10.2014].
[13] Brassil J. T., *„Document Marking and Identification using Both Line and Word Sifting”*, AT&T Bell Laboratories, Murray Hill NJ 07974, http://www.cs.ucla.edu/~miodrag/cs259-security/low94document.pdf, [28.10.2014].
[14] Fridrich Jessica, Goljan Miroslav, Hogea Dorin, *„Breaking the OutGuess”*, 2002r., http://www.sigmm.org/archive/MMSec/mmsec02/outguess.pdf, [28.10.2014].
[15] OutGuess, http://www.outguess.org/, [28.10.2014].
[16] Wikipedia, http://en.wikipedia.org/wiki/Discrete_cosine_transform, [28.10.2014].
[17] Wikipedia, *„JPEG File Interchange Format”*, http://en.wikipedia.org/wiki/JPEG_File_Interchange_Format**,** [28.10.2014].
[18] Maćkowiak Krzysztof, *“Tajemnica ukryta w DNA”*, http://software.com.pl/tajemnica-ukryta-w-dna/, [28.10.2014].
[19] Audio kodek FLAC, http://en.wikipedia.org/wiki/FLAC, [5.11.2014].
[20] Grzegorz Kozieł, *„Zmodyfikowane metody cyfrowego przetwarzania sygnałów dźwiękowych w steganografii komputerowej”,* Politechnika Lubelska, 2010 r..
[21] Duncan Sellars, „An Introduction to Setagography”, http://totse.mattfast1.com/en/privacy/encryption/163947.html, [14.11.2014].
[22] Marcin Wilczewski, *„Algorytmy graficzne – Charakterystyki oraz wyszukiwanie obrazów cyfrowych”*, Politechnika Gdańska, http://www.mif.pg.gda.pl/homepages/marcin/Wyklad2.pdf, [16.11.2014]