JOURNAL
COMPUTER SCIENCES INSTITUTE

JCSI 21 (2021) 383–390
Received: 23 September 2021
Accepted: 26 October 2021

An analysis of the possibility of realization steganography in C#

Badanie możliwości realizacji steganografii w języku C#

Piotr Pawlak*, Jakub Bogdan Podgórniak*, Grzegorz Kozieł

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract

The computing power of modern computers is sufficient to break many cryptographic keys, therefore it is necessary to create an additional security layer which hides the very fact of transmitting a secret message. For this purpose, steganographic methods can be used. The article is devoted to the analysis of the possibility of implementing digital images steganography with the use of the C # programming language. Firstly, existing libraries and mathematical transformations which can help with performing steganography were found. Also, own code solutions were implemented. In order to objectively evaluate the methods of data hiding, the parameters describing the degree of distortion of transforms and hidden images were calculated. Subsequently, optimal solutions for specific problems were identified and demonstrational data hiding was performed. Based on the obtained results, it can be concluded that it is possible to successfully implement steganography in the C # language. There are many ready-made libraries and tools, the effectiveness of which has been verified in the conducted analysis. Due to the contradictory of stenographic requirements, it is not possible to meet all of them optimally, i.e. undetectability, resistance to destruction and information capacity. For this reason, it is not possible to clearly indicate the best solutions. In order to achieve satisfactory results, one should look for compromises between the set requirements.

Keywords: steganography; C# programming language; data hiding; digital image processing

Streszczenie

Moc obliczeniowa współczesnych komputerów jest wystarczająca do łamania wielu zabezpieczeń kryptograficznych, w związku z powyższym konieczne jest utworzenie dodatkowej warstwy bezpieczeństwa polegającej na ukryciu samego faktu przekazywania tajnej wiadomości. W tym celu mogą zostać wykorzystane metody steganograficzne. Artykuł poświęcono analizie możliwości realizacji steganografii w obrazach cyfrowych przy wykorzystaniu języka programowania C#. Wytypowane zostały istniejące biblioteki, przekształcenia matematyczne, a także zaimplementowane zostały własne rozwiązania. W celu dokonania obiektywnej oceny metod ukrywania danych obliczono parametry opisujące stopień zniekształceń transformat oraz ukrywanych obrazów. Następnie wyłoniono optymalne rozwiązania dla konkretnych problemów oraz przeprowadzono demonstracyjne ukrycie danych. Na podstawie otrzymanych rezultatów można stwierdzić, że możliwe jest kompleksowe zrealizowanie steganografii w języku C#. Istnieje wiele gotowych bibliotek i narzędzi, których skuteczność została zweryfikowana w przeprowadzonej analizie. Z racji sprzeczności wymagań stenograficznych nie jest możliwe optymalne spełnienie ich wszystkich tj.: niewykrywalności, odporności na zniszczenie i pojemności informacyjnej. Z tego powodu nie jest możliwe jednoznaczne wskazanie najlepszych rozwiązań. Aby osiągnąć zadowalające rezultaty należy szukać kompromisów pomiędzy stawianymi wymaganiami.

Słowa kluczowe: steganografia; język C#; ukrywanie danych; przekształcanie obrazów cyfrowych

*Corresponding author

Email address: piotr.pawlak3@pollub.edu.pl (P. Pawlak), jakub.podgorniak@pollub.edu.pl (J. B. Podgórniak)

©Published under Creative Common License (CC BY-SA v4.0)

1. Wstęp

Steganografia jest nauką zajmującą się ochroną cennej informacji poprzez ukrywanie samego faktu jej przekazywania. Dzięki temu osoby postronne nie są w stanie wykryć obecności przekazu, co jest szczególnie istotne w obliczu obecnego postępu technologicznego. Narzędzia łamiące szyfry mają przewagę nad mechanizmami zabezpieczającymi przepływ danych. W momencie, gdy osoby nieupoważnione nie są świadome faktu przekazywania wiadomości, nie mogą podjąć się próby jej przechwycenia. Skutkuje to wypełnieniem braków dotychczasowych systemów zabezpieczeń. Istnieje wiele technik steganografii, co jest jej istotnym atutem w kontekście doboru odpowiedniej techniki do zaistniałych okoliczności, tak aby spełniała ona wymagania ukrywającego i pozwoliła na skuteczną i tajną komunikację [13].

Wraz z rosnącą ilością danych przekazywanych przez Internet, a także z ciągłym rozpowszechnianiem się metod transmisji wszelkich rodzajów informacji, zwiększyła się liczba sposobów na jakie można realizować ukrywanie informacji, a także sama potrzeba niejawnego przekazywania danych. Wszystkie te czynniki przyczyniły się do wzrostu zainteresowania steganografią, dzięki czemu dostępnych jest wiele narzędzi i rozwiązań pozwalających na jej przeprowadzenie. Wzrost popularności tej dziedziny nauki poskutkował publikacją wielu pozycji naukowych traktujących o problemie zachowania bezpieczeństwa w tajnej komunikacji [10][12][18].

Obecna sytuacja pokazuje, że steganografia i związane z nią ukryte kanały transmisji danych będą zyskiwać na znaczeniu w kontekście bezpieczeństwa przekazywanych informacji. Jest to związane ze słabościami istniejących systemów bezpieczeństwa. W związku z tym ważne jest ciągłe przeprowadzanie badań dotyczących steganografii, tak aby lepiej poznać tę dziedzinę nauki.

W artykule przedstawiono i przeanalizowano wybrane metody realizacji steganografii przy wykorzystaniu obiektowego języka programistycznego C#. Dodatkowo określono skuteczność wybranych narzędzi z wykorzystaniem obiektywnych jak również subiektywnych form oceny, tak aby wyznaczyć optymalne rozwiązania dla poszczególnych problemów pod względem trójkąta sprzeczności wymagań.

2. Analiza literatury

W artykułach, zebranych podczas przeglądu literatury, poruszone zostały kwestie wykorzystania metod steganografii takich jak m. in. przetwarzanie obrazu w celu ukrycia danych. Informacje te, po analizie, zostaną wykorzystane jako jeden z obszarów badawczych.

W publikacji „Image steganography techniques: an overview" [12] autorzy skupili się na wykorzystaniu obrazu jako nośnika informacji, dlatego przedstawiona została taksonomia obecnych technik steganografii dla plików graficznych. Omówione techniki przetwarzania obrazów zostały przeanalizowane i omówione pod kątem zdolności ukrywania informacji w plikach obrazów, ilości informacji, które można ukryć, a także odporności na różne ataki, mające na celu zniszczenie przekazywanej informacji.

W przeglądzie „Reversible steganography techniques: A survey" [14] autorzy przygotowali kompleksowe zestawienie metod przeprowadzenia steganografii na bazie obrazów. Opisane techniki to rozszerzanie różnic (Difference Expansion), przesuwanie histogramów (Histogram-Shifting), porządkowanie zbioru próbek wektorowych (Pixel-Value-Ordering), schematy bazujące na dwóch obrazach oraz schematy bazujące na interpolacji. Każda z metod została szczegółowo opisana oraz zobrazowana przykładami jej wykorzystania, co czyni z tego przeglądu cenne źródło informacji podczas zagłębiania się w naukę jaką jest steganografia.

Autorzy artykułu „Analiza właściwości metod steganografii odwracalnej" [19] dokonali analizy trzech metod steganografii odwracalnej z wykorzystaniem obrazów cyfrowych jako nośnika. Wspomniane metody to: rozszerzanie różnić, obszar zainteresowania oraz modyfikacja histogramu. Każda ze zbadanych metod okazała się mieć nieco inną charakterystykę, co pozwoliło autorom wysnuć wniosek, że reagują one w różny sposób na cechy obrazu nośnego. Opracowanie przedstawia również szczegółowe wyniki pomiaru parametrów każdej z testowanych metod oraz prezentuje sposób porównania ich między sobą.

Artykuł "An overview of image steganography" [15] zawiera przegląd różnych technik steganografii z wykorzystaniem nośnika w postaci plików graficznych. Autorzy jako cel postawili sobie odnalezienie odpowiedzi na pytanie, która technika ukrywania danych jest najbardziej odpowiednia w zależności od różnych zastosowań oraz to jakie cechy pozwalają zidentyfikować dobry algorytm steganograficzny. Omawiane przez autorów techniki to: ukrywanie danych w najmniej znaczącym bicie bitmapy, steganografia z wykorzystaniem kompresji JPEG, metoda mozaiki oraz magazynowanie informacji w zdjęciu jako szum Gaussowski.

3. Metodyka badawcza

Na podstawie przeglądu literatury określono jakie przekształcenia są wykorzystywane w steganografii. Dla tychże przekształceń zidentyfikowano istniejące dla nich implementacje w języku C#. Następnie przy użyciu tych rozwiązań przeprowadzono transformacje na identycznych obrazach źródłowych. Obliczone zostały parametry służące porównaniu skuteczności danych metod.

Dla każdego przekształcenia wybrane zostały biblioteki dostępne w języku C#. Wykorzystane biblioteki to: Accord.NET [3], Aforge.NET [4], Math.NET [5], NWaves [6], UMapx[7], Universal.Common.Mathematics [8], TrentTobler.Algorithms.FourierTransform [9] oraz inne implementacje znalezione w repozytoriach GitHub. Jeżeli dla danej operacji nie była dostępna wystarczająca liczba gotowych rozwiązań, zaimplementowane zostało własne. Dzięki temu możliwe było wskazanie najlepszego narzędzia do rozwiązania konkretnego problemu. Z bazy flickr.com pobrano 100 publicznie dostępnych obrazów i na każdym z nich, przeprowadzono transformację z wykorzystaniem wybranych wcześniej rozwiązań. W kolejnym kroku, tak uzyskane transformaty poddano przekształceniu odwrotnemu do tego, z którego została ona uzyskana. Otrzymane obrazy wynikowe porównano z ich obrazami źródłowymi. Uzyskane wyniki uśredniono i porównano w obrębie metod, oraz pomiędzy każdą z metod.

Parametry wytypowane do przeprowadzenia oceny zastosowania danych bibliotek to: błąd średniokwadratowy (ang. Mean Square Error), znormalizowany błąd średniokwadratowy (ang. Normalized MSE) oraz szczytowy stosunek sygnału do szumu (ang. Peak SignaltoNoise Ratio). Pozwolą one na obiektywne porównanie obrazów wynikowych, uzyskanych poprzez zastosowanie poszczególnych bibliotek, z obrazami źródłowymi, a następnie określenie optymalnych rozwiązań.

Do przeprowadzenia analizy możliwości realizacji steganografii w języku C# wykorzystano komputer stacjonarny z zainstalowanym systemem Windows 10 64 bit w wersji 10.0.19042, posiadający procesor AMD Ryzen 5 2600 oraz pamięć RAM G.SKILL 16GB (2x8GB) 3000MHz CL16 Aegis. Do implementacji przekształceń matematycznych, a także algorytmów służących ukrywaniu informacji wykorzystano język C# w wersji 8 oraz środowisko programistyczne .NET w wersji 5.0.7

4. Analiza możliwości realizacji przekształceń obrazów w C#

W procesie realizacji steganografii wykorzystującej grafiki komputerowe jako nośnik ukrywanych danych, możliwe jest wykorzystanie całego spektrum przekształceń matematycznych, takich jak: dyskretna transformacja Fouriera, szybka transformacja Fouriera, dyskretna transformacja cosinusowa, transformacja falkowa oraz transformacji do nich odwrotnych. W przypadku formatów obrazów cyfrowych takich jak BMP, gdzie nie jest stosowana kompresja, możliwe jest również przeprowadzenie steganografii opierającej się na fundamentalnych właściwościach przetwarzania cyfrowego, czyli z użyciem operacji binarnych. Kluczowym aspektem dobrze zrealizowanego ukrycia danych jest ich niewykrywalność. W związku z tym należy zbadać, czy podane przekształcenia mogą zostać zrealizowane za pomocą języka programowania C#, tak aby otrzymany w ich wyniku błąd wprowadzał zniekształcenia trudne do wykrycia gołym okiem. Poza subiektywną oceną obrazów uzyskanych w wyniku transformacji, wyliczone zostały parametry określające stopień błędu.

4.1 Operacja podmiany bitów piksela

W grafice komputerowej wykorzystywane bitmapy zbudowane są z tablic liczb całkowitych przedstawionych w formie binarnej. Większość języków programowania, w tym C#, pozwala na manipulowanie wartościami pojedynczych bitów danych. Możliwość ta może zostać wykorzystana do przeprowadzenia steganografii. Jedną z najbardziej popularnych metod ukrywania danych jest metoda najmniej znaczącego bitu.

Aby zobrazować możliwość realizacji steganografii z użyciem manipulacji wartości bitów wykonano następujące czynności: wczytany został obraz źródłowy, wartość najbardziej znaczącego bitu została odwrócona przy wykorzystaniu operacji alternatywy rozłącznej, po uzyskaniu grafiki z przekłamanymi bitami przeprowadzono proces odwrotny, tak by odzyskać pierwotne dane. Badanie to ma na celu zweryfikowanie czy obraz został zniekształcony poprzez przeprowadzenie badanego przekształcenia, a następnie jego odwrócenie.

Zgodnie z oczekiwaniami obraz otrzymany po przekształceniu, a następnie przywróceniu wartości początkowych jest identyczny do obrazu źródłowego. Zatem w momencie wykonania manipulacji wartości poszczególnych bitów i późniejszych operacji odwrotnych przeprowadzonych na dokładnie tych samych bitach, uzyskany efekt jest taki jakby przekształcenie nigdy nie miało miejsca. Fakt ten potwierdzają otrzymane wyniki porównania nośnika oryginalnego z nośnikiem po przeprowadzeniu opisanej powyżej operacji, które przedstawiono w Tabeli 1.

Tabela 1: Wartości parametrów dla operacji bitowej XOR

Metoda	MSE	NMSE	PSNR [dB]
XOR	0	0%	-

4.2 Dyskretna Transformacja Fouriera

Transformacja Fouriera jest matematyczną transformacją, która transformuje sygnał z dziedziny czasu do dziedziny częstotliwości. Wynikiem tej transformacji jest transformata Fouriera i zawiera ona informacje o częstotliwościach składowych sygnału źródłowego [16]. Transformacja ta jest przydatna przy analizie częstotliwości sygnałów. W przypadku przetwarzania sygnałów cyfrowych koniecznie jest zastosowanie dyskretnej transformacji Fouriera.

Możliwe jest również przeprowadzenie odwrotnej transformacji Fouriera, która pozwala odzyskać oryginalne dane z transformaty. Zatem przekształca ona sygnał z dziedziny częstotliwości do dziedziny czasu.

Znalezione zostały dwie istniejące biblioteki dostarczające implementację dyskretnej transformacji Fouriera. Pierwszą z nich jest TrentTobler.Algorithms.FourierTransforms (TAF), a druga to Universal.Common.Mathematics (UCM). Dodatkowo algorytm DFT został zaimplementowany od podstaw jako trzecie rozwiązanie.

Aby dokonać obiektywnej oceny wybranych bibliotek dla każdej otrzymanej grafiki wyliczone zostały parametry: MSE, NMSE oraz PSNR. Otrzymane wyniki zostały przedstawione w Tabeli 2.

Tabela 2: Wartości parametrów dla poszczególnych metod

Metoda	MSE	NMSE	PSNR [dB]
TAF	0,000146	0,0379%	38,36
UCM	0,000154	0,0401%	38,11
Własne	0,000154	0,0401%	38,11

Z otrzymanych wartości wynika, że najmniejszy błąd został otrzymany przy użyciu biblioteki TAF. Pozostałe dwa rozwiązania wygenerowały identyczne wyniki, co może wskazywać na to, że wykorzystują one identyczny model matematyczny.

Wszystkie wykorzystane implementacje DFT oraz transformacji odwrotnej, mogą zostać z sukcesem wykorzystane do przeprowadzenia steganografii.

4.3 Szybka Transformacja Fouriera

Szybka transformacja Fouriera (ang. Fast Fourier Transform, FFT) to algorytm wykorzystywany do określenia dyskretnej transformaty Fouriera, a także transformaty do niej odwrotnej. Ma ona taką przewagę nad transformacją Fouriera, że w wyniku wykorzystania algorytmu Cooley-Tukeya, działającego zgodnie z maksymą „dziel i rządź", jej złożoność obliczeniowa jest mniejsza, a zatem przeprowadzenie transformacji będzie trwało krócej. Porównując, złożoność obliczeniową dyskretnej transformacji Fouriera to O(N2), a złożoność szybkiej transformacji Fouriera wynosi O(Nlog2N), widoczny jest duży zysk czasu w przypadku wybrania takiej implementacji [11].

Po przeprowadzeniu analizy dostępnych bibliotek wytypowano 3 rozwiązania, dostarczające implementację algorytmu FFT. Pierwszą z wytypowanych bibliotek

jest AForge.NET. Jej ogólnym przeznaczeniem jest dostarczenie szkieletu programistycznego dla programistów zajmujących się sztuczną inteligencją oraz cyfrowym rozpoznawaniem obrazów, przy których to FFT znajduje zastosowanie. Kolejną biblioteką jest Accord.NET. W tym przypadku kluczowym obszarem, dla którego wykorzystano algorytm FFT jest uczenie maszynowe. Ostatnią z testowych bibliotek jest Math.NET. Implementacja FFT w tej bibliotece pozwala na zastosowanie różnych trybów skalowania oraz określenie znaku wykładnika.

Aby obiektywnie porównać wybrane biblioteki dla każdej otrzymanej grafiki wyliczone zostały parametry MSE, NMSE oraz PSNR. Otrzymane wyniki zostały przedstawione w Tabeli 3.

Tabela 3: Wartości parametrów dla poszczególnych metod

Metoda	MSE	NMSE	PSNR [dB]
A.Forge.NET	0,000156	0,0404%	38,082
Accord.NET	0,00015	0,0391%	38,222
Math.NET domyślny	0,000149	0,0386%	38,275
Math.NET skalowanie	0,000151	0,0391%	38,222
Math.NET nieskalowany	0,319949	83,0941%	4,949

Ze wszystkich zbadanych metod najmniejszy błąd wprowadza algorytm biblioteki Math.NET wykorzystany w trybie domyślnym. Drugim najbardziej efektywnym rozwiązaniem jest to zaoferowane przez bibliotekę Accord.NET. Największy błąd, widoczny również na grafice, wystąpił przy wykorzystaniu biblioteki Math.NET w trybie bez skalowania. Porównując dyskretną transformację Fouriera do szybkiej transformacji Fouriera warto zauważyć, że najmniejszy poziom zniekształceń uzyskano przy wykorzystaniu pierwszej z nich. Jednakże, różnica pomiędzy najlepszym wynikiem DFT, a FFT jest znikoma i niezależnie od wybranej metody, nie powinna wprowadzać ona błędu, który znacząco obniżyłby jakość przeprowadzonej steganografii. Oznacza to, że z powodzeniem można zastosować obie te metody w celu ukrycia faktu przekazywania tajnych informacji

4.4 Dyskretna transformacja cosinusowa

Następnym przekształceniem, mogącym posłużyć do przeprowadzenia steganografii z wykorzystaniem obrazu jako nośnika danych jest dyskretna transformacja cosinusowa (ang. discrete cosine transform, DCT). DCT jest to rodzaj transformacji blokowej przeprowadzonej na wartościach dyskretnych [1][19]. Dzięki zastosowaniu transformacji odwrotnej do DCT możliwe jest przywrócenie pierwotnych danych.

DCT jest powszechnie używana do kompresji obrazów cyfrowych. Na jej podstawie opracowany został format JPEG. Struktura tego formatu pozwala na ukrycie danych w współczynnikach DCT, dlatego format ten jest odpowiednim kandydatem do bycia bazą operacji steganograficznej.

Spośród dostępnych rozwiązań, pozwalających na przeprowadzenie dyskretnej transformacji cosinusowej, wytypowano trzy następujące: Accord.NET, Universal.Common.Mathematics oraz autorską implementację udostępnioną przez nauczyciela akademickiego dr Vinyaka Bharadi (VB).

Dla przeprowadzenia obiektywnej oceny wyliczone zostały parametry MSE, NMSE oraz PSNR, umożliwiające porównanie skuteczności wytypowanych bibliotek. Wartości parametrów dla poszczególnych rozwiązań widoczne są w Tabeli 4.

Tabela 4: Wartości parametrów dla poszczególnych metod

Metoda	MSE	NMSE	PSNR [dB]
Accord.NET	0,000637	0,1652%	31,96
VB	0,000628	0,1628%	32,02
UCM	0,000639	0,1656%	31,95

Na podstawie wykonanej analizy, można wnioskować, że każde z badanych narzędzi może zostać z sukcesem użyte do przeprowadzenia steganografii. Najlepsze wyniki zostały uzyskane przy wykorzystaniu implementacji dr Vinayaka Bharadi. Jednakże wartości parametrów dla wszystkich trzech metod są do siebie zbliżone i optymalne.

4.5 Transformacja falkowa

Transformacja falkowa jest to przekształcenie, podobnie jak w przypadku transformacji Fouriera, oparte na operacji iloczynu skalarnego badanego sygnału. Istnieje wiele sposobów przeprowadzenia transformacji, uzyskanych w wyniku doboru różnych falek. Falką nazywamy jądro przekształcenia transformacji, czyli funkcję spełniającą wymagania analizy czasowo-częstotliwościowej. Funkcja ta jest natury sinusoidalnej [11].

Do przeprowadzenia analizy transformacji falkowej wytypowano trzy rodzaje falek: falka Haara, falka CDF97 oraz falka Daubechies rzędu 4. Falka Haara to najstarsza oraz najprostsza znana falka. Jej autorem jest Alfréd Haar, a utworzona została na początku XX wieku.

Następną wykorzystaną falką jest CDF97 (Cohen-Daubechies-Feauveau 9/7). Należy ona do rodziny falek biortogonalnych [2]. Jest ona wykorzystywana m. in. w standardzie kompresji JPEG 2000 do przeprowadzenia kompresji stratnej. Ostatnią wytypowaną falką jest falka Daubechies rzędu 4, należąca do rodziny falek Daubechies. Rodzina ta swą nazwę wywodzi od nazwiska francuskiej uczonej Ingrid Daubechies. Istnieją falki różnych rzędów, jednak dla każdej z nich otrzymane wyniki oscylowały wokół zbliżonych wartości, dlatego zdecydowano się na szczegółowe opisanie wyłącznie falki rzędu 4-go.

Do przeprowadzenia transformacji falkowej wybrano trzy biblioteki: UMapx, Accord.NET oraz NWaves. UMapx jest to biblioteka udostępniająca funkcje do

procesowania sygnałów cyfrowych. Ze wszystkich trzech rozwiązań, to ona zawiera największą liczbę obsługiwanych falek. Kolejne badane narzędzie – Accord.NET – zawiera interfejs wspierający jedynie transformację z użyciem falki Haara oraz CDF97. Biblioteka NWaves służy głównie do pracy na 1-wymiarowych sygnałach dźwiękowych, jednakże umożliwia również przeprowadzenie transformacji z użyciem falek Haara, Daubechies, Coiflet oraz Symlet.

Aby dokonać obiektywnej oceny, obliczone zostały parametry MSE, NMSE oraz PSNR, tak aby wyznaczyć optymalne rozwiązania dla konkretnych rodzajów falek. Wartości parametrów uzyskane z użyciem falki Haara przedstawiono w tabeli 5, falki CDF97 w Tabeli 6, natomiast wartości dla falki Daubechies 4-go rzędu w Tabeli 7. Dla wszystkich metod wartości parametrów wykazały wystarczająco niski stopień zniekształceń.

Tabela 5: Wartości parametrów (Haar)

Metoda	MSE	NMSE	PSNR [dB]
UMapx	0,000156	0,0404%	38,08
Accord.NET	0,000146	0,0379%	38,36
NWaves	0,000155	0,0402%	38,098

Tabela 6: Wartości parametrów (CDF97)

Metoda	MSE	NMSE	PSNR [dB]
UMapx	0,000155	0,0402%	38,101
Accord.NET	0,000151	0,0393%	38,198

Tabela 7: Wartości parametrów (Daubechies 4-go rzędu)

Metoda	MSE	NMSE	PSNR [dB]
UMapx	0,000156	0,0404%	38,081
NWaves	0,000155	0,0404%	38,086

Na podstawie analizy uzyskanych danych stwierdzono, iż możliwe jest zrealizowanie steganografii z powodzeniem dla każdego z testowanych narzędzi przy użyciu każdej z wybranych falek. Najlepsze wyniki uzyskano dla biblioteki Accord.NET, jednak wadą tego rozwiązania jest to, że wspiera ona jedynie dwie falki. Najgorszy rezultaty uzyskano przy zastosowaniu biblioteki UMapx, jednakże są to wyniki zbliżone do tych otrzymanych przy użyciu NWaves. Warto zaznaczyć, że na korzyść UMapx działa fakt, iż narzędzie to wspiera wiele rodzajów falek.

5. Metoda LSB

Z uwagi na łatwość realizacji steganografii z wykorzystaniem operacji bitowych zdecydowano się na implementację własnej metody mającej na celu ukrywanie tajnych informacji w obrazie. Zaimplementowany algorytm pozwala na określenie liczby najmniej znaczących bitów wykorzystanych do przechowania ukrywanej informacji. Im większa jest ich liczba, tym większa jest pojemność informacyjna, jednak wraz z jej wzrostem zmniejsza się niewykrywalność faktu przeprowadzania steganografii. Na Rysunku 1 przedstawiono przykładowe grafiki wykorzystane w procesie ukrywania danych. Grafika z kotem posłużyła za nośnik, natomiast obraz przedstawiający chmurę stanowił ukrywaną informację.

Rysunek 1: Obraz nośnika i obraz ukrywany

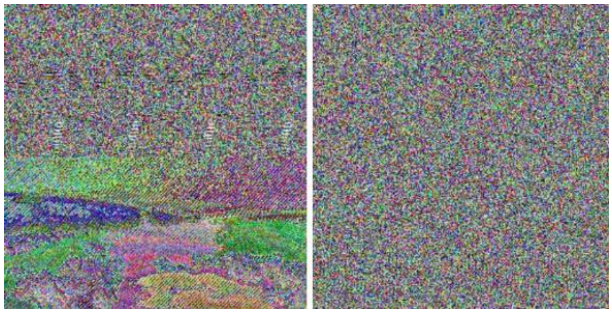
Przeprowadzono steganografię z użyciem różnych rozmiarów ukrywanej grafiki, a także za każdym razem iterowano liczbę podmienianych bitów danych nośnika. W każdym z procesów wykorzystano wszystkie trzy kanały kolorów: czerwony, zielony i niebieski. Im większa jest liczba zmanipulowanych bitów, a co za tym idzie również pojemność informacyjna, tym większe jest wygenerowane zniekształcenie. Obrazy wynikowe przedstawiono na Rysunku 2. Zniekształcenia występują w górnej części obrazów. Wynika to z logiki zaimplementowanej metody steganograficznej, która iteruje po kolejnych wierszach pikseli. Wielkość zniekształconego fragmentu obrazu zależna jest od liczby przekłamanych bitów, która jest pochodną rozmiaru tajnych danych.

Rysunek 2: Obrazy wynikowe przy podmianie 5 oraz 7 bitów danych nośnika

Po otrzymaniu obrazów zawierających ukryte informacje przeprowadzono badanie odporności wykorzystanej metody na zniszczenie ukrywanych danych. Testowe zniekształcenia przeprowadzono na obrazie, w którym ukryty został obraz o rozmiarze 256x256. Dane zostały ukryte we wszystkich trzech kanałach kolorów w trzech najmniej znaczących bitach (Rys. 3). Wykonane zniekształcenia to: obrót, rozmycie oraz kompresja JPG. Rozmycie zastosowano w 5 różnych poziomach, a kompresję JPG w 10 stopniach jakości (od 100% do 10%). Zdecydowano się na zastosowanie takich zniekształceń i ich intensywności, które występują najczęściej podczas przekazywania i przetwarzania grafik cyfrowych.

Rysunek 3: Zniekształcone obrazy steganograficzne

Następnie podjęto się próby odzyskania ukrytych obrazów. Niektóre z otrzymanych rezultatów zostały przedstawione na rysunku 4. Jak można było oczekiwać, ukryte obrazy zostały niemal całkowicie zniszczone, co oznacza, iż metoda LSB nie jest odporna na zniekształcanie obrazu przechowującego ukryte informacje.

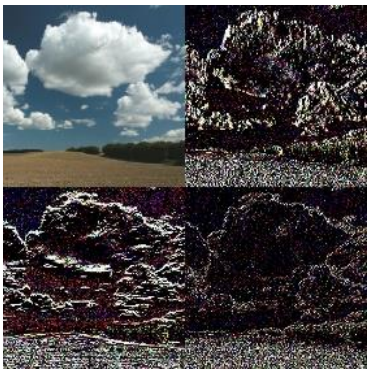


Rysunek 4: Odzyskane obrazy

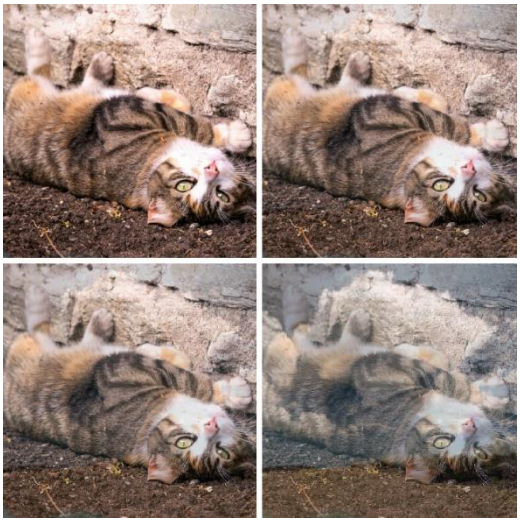
6. Ukrywanie danych przy wykorzystaniu transformacji falkowej

Z uwagi na satysfakcjonująco niski stopień zniekształceń wprowadzanych do grafiki źródłowej przy wykorzystaniu transformacji falkowej zdecydowano się na własną implementację metody bazującej na rozwiązaniu zamieszczonym w artykule „A Discrete Wavelet Transform: A Steganographic Method for Transmitting Images" [17]. Metoda ta bazuje na falce Haara. Tajna wiadomość ukrywana jest po przeprowadzeniu transformaty, której wynikiem jest grafika złożona z czterech obszarów (Rysunek 5), które kolejno reprezentują: LL (reprezentacja transformowanego obrazu po uproszczeniu), LH (ekspozycja krawędzi pionowych), HL (ekspozycja krawędzi poziomych), HH (ekspozycja krawędzi diagonalnych). Sekretne dane umieszczane są w obszarze LL.

Przeprowadzono ukrywanie danych z użyciem różnych rozmiarów tajnej grafiki. W każdym z procesów wykorzystano wszystkie trzy kanały kolorów: czerwony, zielony i niebieski. Ukrywana była grafika o rozmiarze 256x256. Zastosowana została falka Haara pierwszego rzędu. Wybrane grafiki zawierające ukryte dane zostały przedstawione na Rysunku 6. Na pierwszych dwóch grafikach ciężko jest odnaleźć różnice pomiędzy otrzymanymi grafikami, a obrazem źródłowym. Od trzeciej grafiki postępuje coraz bardziej widoczne zniekształcenie obrazu źródłowego. Im wyższa jest wartość współczynnika zagnieżdżenia, tym większy jest stopień zniekształceń wprowadzonych do grafiki wynikowej.

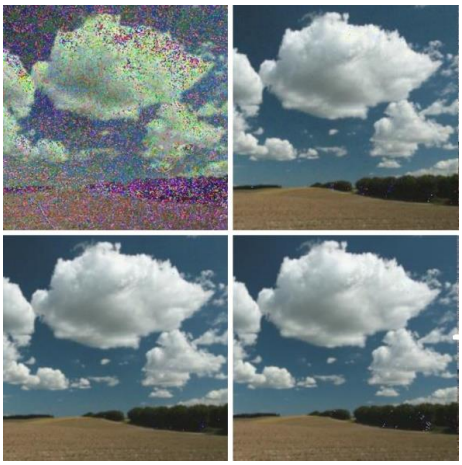


Rysunek 5: Transformata uzyskana z użyciem falki Haara



Rysunek 6: Obrazy wynikowe

Na Rysunku 7 widoczne są ukryte obrazy odzyskane z grafiki będącej nośnikiem tajnej informacji. Obrazy zachowały się najlepiej w przypadku wykorzystania współczynników zagnieżdżenia o wartościach 0,15 i 0,25. W przypadku wartości 0,02 widoczne jest znaczne uszkodzenie ukrywanej grafiki, jednakże sama jej struktura została zachowana. W momencie wykorzystania zagnieżdżenia na poziomie 0,5 zaczynają pojawiać się dodatkowe zniekształcenia ukrywanego obrazu. Jest to skutkiem samej logiki wykorzystywanej metody. Optymalne zatem jest korzystanie z zagnieżdżenia z zakresu od 0,15 do 0,25. Można zatem stwierdzić, iż proces ukrywania danych i ich ponownego odzyskania zakończył się powodzeniem dla współczynników o wartościach zagnieżdżenia 0,15 i 0,25.



Rysunek 7: Odzyskane obrazy dla współczynników zagnieżdżenia o wartościach 0,02, 0,15, 0,25 i 0,5

Aby dokonać obiektywnej oceny zniekształceń obrazów przechowujących ukrywaną informację względem obrazu źródłowego wyliczone zostały parametry: MSE, NMSE oraz PSNR. Otrzymane wyniki zostały przedstawione w Tabeli 8 i 9.

Tabela 8: Wartości parametrów dla różnych wartości współczynnika zagnieżdżenia

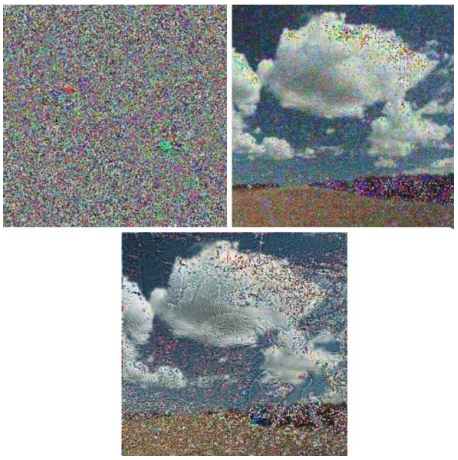
Współczynnik zagnieżdżenia	Obraz z ukrywaną informacją		
	MSE	NMSE	PSNR [dB]
0,02	0,000116	0,0331%	39,34
0,15	0,002217	0,6311%	26,54
0,25	0,005997	1,7066%	22,22
0,5	0,02362	6,722%	16,27

Tabela 9: Wartości parametrów dla różnych wartości współczynnika zagnieżdżenia

Współczynnik zagnieżdżenia	Odzyskany obraz		
	MSE	NMSE	PSNR [dB]
0,02	0,031519	11,7512%	15,01
0,15	0,000248	0,0926%	36,05
0,25	0,000093	0,0348%	40,3
0,5	0,000144	0,0536%	38,43

Dane wskazują na ujemną korelację pomiędzy jakością obrazu zawierającego ukryte dane, a jakością odzyskanego obrazu. Im większy stopień zniekształceń występuje w obrazie z ukrywaną informacją, tym mniej uszkodzony jest przekazywany, tajny komunikat. Analogicznie występuje to w drugą stronę. Wartości obliczonych parametrów potwierdzają przypuszczenia wynikające z naocznej analizy otrzymanych grafik.

Po otrzymaniu obrazów zawierających ukryte informacje przeprowadzono badanie odporności wykorzystanej metody na zniszczenie ukrywanych danych. Testowe zniekształcenia przeprowadzono na obrazie, w którym ukryty został obraz o rozmiarze 256x256. Kryterium doboru zniekształceń było identyczne jak w przypadku testowania metody LSB. Otrzymane rezultaty pozwolą na dokonanie oceny odporności badanej metody na zniszczenie ukrytych danych.



Rysunek 8: Odzyskane obrazy po obróceniu, kompresji JPG i rozmyciu nośnika

Następnie podjęto się próby odzyskania ukrytych obrazów. W przypadku obróconego obrazu ukryte dane zostały całkowicie uszkodzone. Dane można jednak odzyskać poprzez obrócenie obrazu do pierwotnej pozycji.

Kompresja JPG nośnika nie spowodowała zniekształcenia struktury ukrywanej grafiki. Im wyższa jakość kompresji, tym mniejsze zniekształcenie i uszkodzenie grafiki. Można postawić wniosek, że metoda ukrywania danych oparta na transformacji falkowej z wykorzystaniem falki Haara jest odporna na kompresję JPG.

Dla grafik odzyskanych po kompresji JPG obliczone zostały wartości parametrów MSE, NMSE oraz PSNR. Otrzymane rezultaty zostały przedstawione w Tabeli 10. Wraz ze wzrostem jakości kompresji spadają wartości parametrów MSE oraz NSME. Rośnie natomiast wartość parametru PSNR. Oznacza to, że im lepsza jest jakość kompresji, tym mniejsze jest zniekształcenie wprowadzone do grafiki wynikowej.

Tabela 10: Wartości parametrów dla różnych jakości kompresji JPG

Jakość	MSE	NMSE	PSNR [dB]
10%	0,061709	23,0079%	12,1
20%	0,036504	13,6098%	14,38
30%	0,025218	9,4021%	15,98
40%	0,019088	7,1166%	17,19
50%	0,015355	5,7246%	18,14
60%	0,011885	4,431%	19,25
70%	0,008629	3,2172%	20,64
80%	0,005521	2,0585%	22,58
90%	0,002567	0,9572%	25,91
100%	0,000591	0,2202%	32,29

Ostatnim przeprowadzonym zniekształceniem było rozmycie. Im wyższy jest poziom rozmycia, tym wyższy jest stopień zniekształceń ukrywanego obrazu. W

przypadku rozmycia 1-go poziomu odzyskany obraz zachował jedynie swoją pierwotną charakterystykę.

Aby otrzymać obiektywne dane obliczone zostały wartości parametrów MSE, NMSE oraz PSNR. Otrzymane rezultaty przedstawione zostały w Tabeli 11. Widoczne jest, że wraz ze wzrostem poziomu rozmycia wzrastają wartości parametrów MSE i NMSE, zaś wartości parametru PSNR maleją. Oznacza to postępujący wzrost wprowadzanych zniekształceń.

Tabela 11: Wartości parametrów dla różnych poziomów rozmycia

Poziom rozmycia	MSE	NMSE	PSNR [dB]
1	0,03918	14,6075%	14,07
2	0,072758	27,1263%	11,38
3	0,084515	31,5093%	10,73
4	0,091654	34,1709%	10,38
5	0,092999	34,6727%	10,32

7. Wnioski

Założeniem badania była analiza możliwości realizacji steganografii przy wykorzystaniu języka programowania C#. Dołożono szczególnych starań, aby badane narzędzia, biblioteki, a także samodzielnie zaimplementowany kod spełniały oczekiwania osoby chcącej zataić fakt przekazywania tajnych informacji. Wybrano popularne i skuteczne przekształcenia matematyczne, a następnie wyłoniono te, które w wyniku przekształcania obrazu źródłowego generują najmniejszy stopień zniekształceń.

Przeprowadzone zostało demonstracyjne ukrycie danych z wykorzystaniem samodzielnie zaimplementowanych narzędzi. Cały proces zakończył się sukcesem. Autorom udało się wykorzystać grafikę cyfrową jako nośnik tajnych informacji, umieścić w nim drugą, tajną grafikę, a na koniec odzyskać z powrotem ukryte dane z grafiki wynikowej. Wykorzystane metody steganograficzne zostały poddane testom odporności na zniekształcenia i niewykrywalności manipulacji obrazem pierwotnym.

Na podstawie przeprowadzonych działań można stwierdzić, że z powodzeniem możliwe jest zrealizowanie procesów steganograficznych przy wykorzystaniu języka programowania C#. Ponadto istnieje wiele gotowych bibliotek i narzędzi, których skuteczność została zweryfikowana w niniejszej pracy, które upraszczają proces ukrywania danych.

Z racji sprzeczności wymagań steganograficznych nie jest możliwe optymalne spełnienie ich wszystkich tj.: niewykrywalności, odporności na zniszczenie i pojemności informacyjnej. Z tego powodu nie jest możliwe jednoznaczne wskazanie najlepszych rozwiązań. Aby osiągnąć zadowalające rezultaty należy szukać kompromisów pomiędzy stawianymi wymaganiami, a także dopasowywać odpowiednie narzędzie do zaistniałego problemu.

Jako że steganografia dynamicznie się rozwija, a większość znanych metod bazuje na popularnych przekształceniach, ich implementacja w C# jest w pełni możliwa. Pozwala to wnioskować, iż język C# jest narzędziem wystarczającym do kompleksowej realizacji steganografii.

Literatura

[1] N. Ahmed, T. Natarajan, K.R. Rao, Discrete Cosine Transform, IEEE Transactions on Computers, Volume: C-23, 1 (1974) 90-93.

[2] J. Białasiewicz, Falki i aproksymacje, Wydawnictwa Naukowo-Techniczne, Warszawa, 2000.

[3] Biblioteka Accord.NET, github.com/accord-net/framework, [1.04.2021].

[4] Biblioteka AForge.NET, github.com/andrewkirillov/AForge.NET, [1.04.2021].

[5] Biblioteka Math.NET, github.com/mathnet/mathnet-numerics, [1.04.2021].

[6] Biblioteka NWaves, github.com/ar1st0crat/NWaves, [1.04.2021].

[7] Biblioteka UMapx, https://github.com/asiryan/UMapx, [1.04.2021].

[8] Biblioteka Universal.Common.Mathematics, nuget.org/packages/Universal.Common.Mathematics/, [1.04.2021].

[9] Biblioteka TrentTobler.Algorithms.FourierTransform, github.com/trenttobler/FourierTransform, [1.04.2021].

[10] S. Dhawan, R. Gupta, Analysis of various data security techniques of steganography: A survey, ISJ: A Global Perspective, 30(2) (2021) 63-87.

[11] Z. Fortuna, B. Macukow, J. Wąsowski, Metody numeryczne, Wydawnictwa Naukowo-Techniczne, Warszawa, 2015.

[12] N. Hamid, A. Yahya, R. B. Ahmad, O. M. Al-Qershi, Image steganography techniques: an overview, IJCSS, 6(3) (2012) 168-187.

[13] P. Kopniak, Metody cyfrowego przetwarzania sygnałów na potrzeby steganologii komputerowej, Politechnika Lubelska, Lublin, 2007.

[14] T. C. Lu, T. N. Vo, Reversible steganography techniques: A survey, In Digital Media Steganography, Elsevier (2021) 189-213.

[15] T. Morkel, J. H. Eloff, M. S. Olivier, An overview of image steganography, ISSA (2015).

[16] P. Strumiłło, M. Strzelecki, Przekształcenie Fouriera obrazów, Politechnika Łódzka, Łódź, 2006.

[17] M. A. Wakure, A. N. Holambe, A Discrete Wavelet Transform: A Steganographic Method for Transmitting Images, IJCA, 129 (2015) 26-29.

[18] Z. Yuan, D. Liu, X. Zhang, Q. Su, New image blind watermarking method based on two-dimensional discrete cosine transform, Optik, 204 (2020) 164152.

[19] P. Zimnicki, G. Kozieł, Analiza właściwości metod steganografii odwracalnej, JCSI, 8 (2018) 292-297.