**PRACA INŻYNIERSKA**

**Temat: Zastosowanie algorytmu wykorzystującego ruchy skoczka szachowego do szyfrowania chaotycznego plików graficznych.**

***The application of the algorithm that uses the movements of a chess knight to encrypt chaotic graphic files.***

Imię i nazwisko: Michał Kaliszczyk

**Spis treści**

1. **Wstęp**

1.1Cel i założenia projektu

1.2 Uzasadnienie wyboru algorytmu

1. **Algorytm szyfrowania plików graficznych**

2.1 Wprowadzenie

2.1 Reguły ruchów skoczka szachowego

2.2 Model scramblera

2.3 Opis algorytmu

2.4 Lista kroków

2.5 Schemat blokowy

1. **Algorytm przeszukiwania szachownicy**

3.1 Przedstawienie problemu

3.2 Opis rozwiazań

3.3 Opis algorytmów

3.4 Analiza

3.5 Przykłady Rozwiązań(Eulera, , ,, , ,)

……...

**4. Oprogramowanie**

4.1 Wprowadzenie

4.2 Microsoft Visual Studio

4.3 Visual Paradigm

4.4 Matlab

4.5 Uzasadnienie wybranych technologii

**5. Projekt aplikacji**

5.1 Model aplikacji oraz wykorzystane diagramy UML

5.2 Przypadków użycia

5.3 Pakietów

5.4 Klas

5.5 Aktywności

5.6 Sekwencji

5.7 Komponentów

**6. Opis implementacji**

6.1 Opis użytych narzędzi

6.2 Opis klas oraz metod

6.3 Przepływ sekwencji aplikacji

6.4 Wyniki przeprowadzonych testów jednostkowych

6.5 Opis trudności w implementacji

**7. Opis analizy oraz przeprowadzonych eksperymentów**

7.1 Czas wykonywania algorytmu

(Analizy Wnioski)

**8. Podsumowanie**

**Bibliografia**

1. Wstęp

W dzisiejszych czasach zabezpieczanie danych na komputerze jest jedną z ważniejszych rzeczy o ,którym każdy użytkownik powinien pamiętać. Wynika to z tego ,że coraz więcej informacji przechowywanych jest na naszym komputerze nie tylko prywatnych ,ale także służbowych, dlatego bardzo istotne jest żeby zadbać o ich bezpieczeństwo.Szczególnie istotnym jest utrata danych ,które przechowujemy na naszych urządzeniach, ponieważ mogą się dostać do niepowołanych osób w przypadku kradzieży komputera, lub kiedy nasze urządzenie zostanie zainfekowane przez złośliwe oprogramowanie. Dlatego bardzo ważne jest żeby każdy użytkownik ,który posiada poufne informacje szyfrował je. Dzięki temu mamy zabezpieczenie ,że tylko odpowiednie osoby będą wstanie odczytać informacje a w przypadku gdy zostaną one skradzione przez jakąś osobę nie zostaną odczytane ani w żadnym stopniu wykorzystane bez naszej zgody.

1.1Cel i założenia projektu

Celem projektu jest stworzenie aplikacji ,która będzie skutecznie i w jak najkrótszym czasie szyfrowała i odszyfrowywała pliki graficzne. Bardzo ważne jest żeby chronić zdjęcia tak samo jak inne informacje ,ponieważ powszechnie wiadomo ,że fotografie często zawierają ogromne ilości informacji jak na przykład zapisane hasła ,dane osobowe oraz informacje o naszych znajomych, rodzinie. W przypadku przedostania się zdjęcia do niepowołanej osoby nie będzie ona wstanie dowiedzieć się co jest zapisane na obrazie oraz odszyfrować go nie znając klucza. Dzięki temu użytkownicy będą mogli bezpiecznie przechowywać swoje zdjęcia na komputerze oraz przesyłać je między sobą bez obawy o wykorzystanie ich przez niepowołane osoby. Założeniem projektu jest użycie techniki szyfrowania obrazów kolorowych poprzez transpozycję pikseli między kanałami RGB wykorzystując ruchy skoczka szachowego i mapy chaotycznej do generowania reguł dla każdego piksela w obrazie.

1.2 Uzasadnienie wyboru algorytmu

Wybrałem ten algorytm ,ponieważ wykorzystuje on ruchy skoczka szachowego z gry w szachy co jest ciekawym sposobem szyfrowania oraz używa on generatora opartego na chaosie do losowanie bitów pseudolosowych z ,których tworzone są reguły do przemieszczania się konika szachowego posiada on dobre cechy losowości .W wyniku tego otrzymujemy efektywnie zaszyfrowany obraz ,ponieważ mamy niski współczynnik korelacji sąsiednich pikseli .

// TO DO

Czas potrzebny do wykonania algorytmu jest niewielki oraz klucz ma dużą przestrzeń.

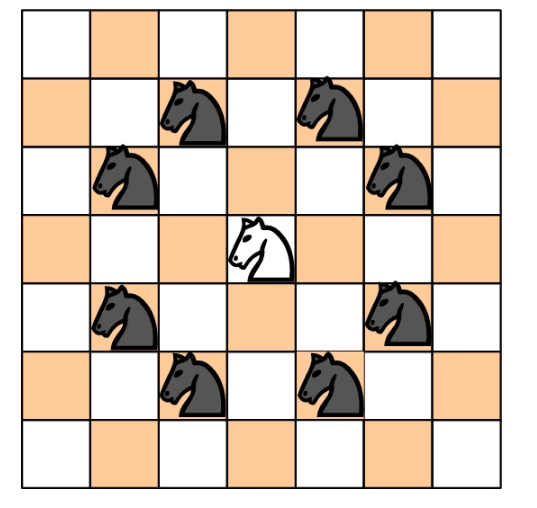
**2. Algorytm szyfrowania plików graficznych**

2.1 Wprowadzenie

Jest to algorytm służący do szyfrowania plików graficznych ,który wykorzystuje technikę scramblingu do transpozycji pikseli, między kanałami RGB. Używane są do tego ruchy skoczka szachowego oraz model skramblera opartego na chaosie.

2.1 Opis ruchów skoczka szachowego

Konik szachowy jest to figura w popularnej grze szachy, która porusza się w kształcie litery “L” Przemieszcza się pionowo o dwa pola a następnie poziomo o jedno pole lub poziomo o dwa pola a następnie pionowo o jedno pole, można to zrobić w dowolnym kierunku. Maksymalna liczba możliwych ruchów konika szachowego wynosi 8. Możliwe ruchy zostały przedstawione na rysunku poniżej.



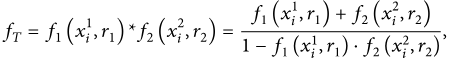
**Rysunek 1 Ruchy skoczka szachowego**

Żródło :https://www.hindawi.com/journals/mpe/2014/932875/fig1/

2.2 Model scramblera

Scrambler jest to metoda, która służy do randomizacji sekwencji bitowych. Jego głownym celem jest osiągniecie jak najbardziej prawdziwie losowej sekwencji bitów. W prezentowanym modelu scramblera zostały użyte dwie identyczne chaotyczne mapy, których równanie zostało przedstawione poniżej.

Do generowania bitów z przedstawionych równań zostało użyte równanie Ft



To równanie zostało zaprojektowane w oparciu o skład binarny



W wyniku czego posiadamy mapy chaotyczne, które są sprzężone, czyli wyjście pierwszego jest podawane do kolejnego

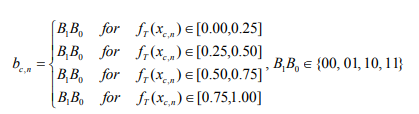
 

Wartości początkowe do wyliczania równania są przedstawione następująco :



Równanie Ft zostaje wyliczone odpowiednią ilość razy w zależności od wielkości zdjęcia, czyli dla wszystkich kolorów każdego piksela. Po każdym obliczeniu tego równania zostaje on poddany metodzie dyskretyzacji.



Jeżeli wartość Ft znajduje się w przedziale od 0 do 0.25 to zapisujemy do pierwszego i drugiego pliku wartość 0

Jeżeli wartość Ft znajduje w przedziale od 0.25 do 0.50 to zapisujemy do pierwszego pliku wartość 0 a do drugiego wartość 1

Jeżeli wartość Ft znajduje się w przedziale od 0.50 do 0.75 to zapisujemy do pierwszego pliku wartość 1 a do drugiego wartość 0

Jeżeli wartość Ft znajduje się w przedziale od 0.75 do 1.00 to zapisujemy do pierwszego i drugiego pliku wartość 1

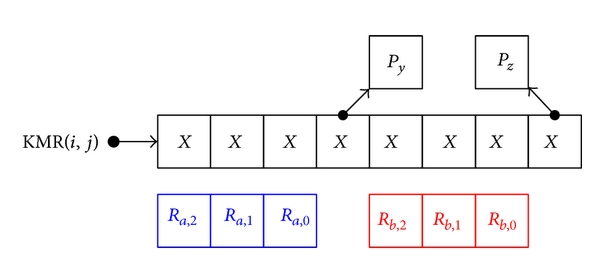
Odpowiednio od przekazanego wyniku z równania Ft otrzymujemy dwa bity, z których jeden jest zapisywany do pierwszego pliku a drugi do drugiego pliku tak jak to zostało przedstawione powyżej. Potrzebne one będa do ustalenia ilości przemieszczeń skoczka szachowego, jego kierunku ruchu oraz transpozycji wartości koloru. W rezultacie osiągniemy zmniejszoną wrażliwość korelacji między wartością przemieszczanego piksela.

2.3 Opis algorytmu

Algorytm rozpoczynamy od wczytania zdjęcia w celu pobrania jego wartości pikseli oraz wymiarów. Odpowiednio od pobrania wysokości i szerokości obrazu generujemy dwa pliki, w których zawarte są sekwencje zer i jedynek przy użyciu scramblera opisanego w podpunkcie 2.2. Te sekwencje potrzebne będą nam do określania reguł przemieszczania się pikseli między kanałami RGB używając ruchy skoczka szachowego. Tworzymy dwie tablice trójwymiarowe KMR i RNS w, których będziemy przechowywać reguły. Wielkość tych tablic jest taka sama do jakiej zapisaliśmy wartości pikseli obrazu dla przykładu jeżeli posiadamy zdjęcie o wymiarach 512 na 256 to wtedy tworzyme tablice o wymiarach 512 x 256 x 3 dla każdego koloru RGB (czerwonego, zielonego, niebieskiego) i do każdej komórki zapisujemy po 8 bitów. Z pierwszego pliku zapisujemy wartości do tablicy KMR a z drugiego do tablicy RNS. Czynności, które zostały przedstawione są wykonywane tak samo przy szyfrowaniu oraz odszyfrowywaniu obrazu.

Szyfrowanie :

Dla wszystkich pikseli rozpoczynając od koloru czerwonego potem zielonego a na końcu niebieskiego musimy wykonać następujące procedury. Wykonujemy generowanie reguł z tablic KMR i RNS dla poszczególnego piksela. Dla tablicy KMR robimy to następująco



**Rysunek generowanie reguł z tablic KMR**

Żródło : <https://www.hindawi.com/journals/mpe/2014/932875/fig3/>

Reguły jakie otrzymujemy z tablic KMR są następujące: Ra, Rb, Py, Pz

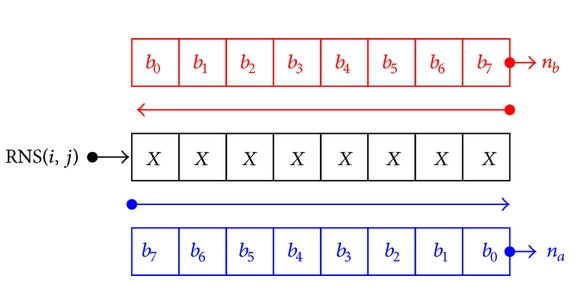
Pz jest pobierana z pierwszego bitu

Rb jest pobierana od 2 do 4 bitu

Py jest pobierana z 5 bitu

Ra jest pobierana od 6 do 8 bitu

Natomiast dla tablicy RNS



**Rysunek generowanie reguł z tablic RNS**

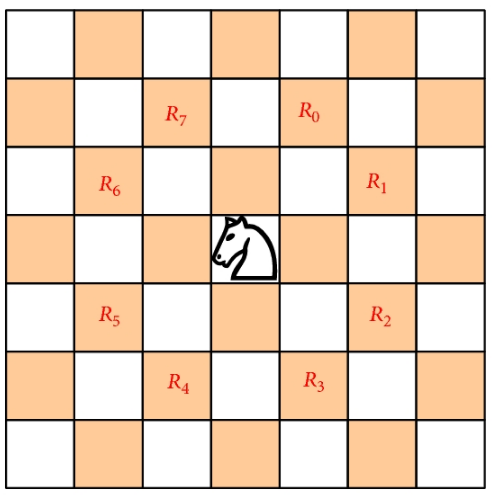
Żródło : <https://www.hindawi.com/journals/mpe/2014/932875/fig3/>

Reguły jakie otrzymujemy z tablic RNS są następujące: Na, Nb

Na jest pobierana od pierwszego do ostatniego bitu

Nb jest pobierana od ostatniego do pierwszego bitu

Kiedy wygenerowaliśmy reguły możemy dla obecnego piksela użyć ich żeby znaleźć dwa inne piksele między którymi będziemy zamieniać wartości. Robimy to następująco pierwszy piksel znajdujemy za pomocą uzyskanej zmiennej **Ra** ,który wskazuje nam kierunek ruchu skoczka szachowego natomiast ze zmiennej **Na** dostajemy ile razy ruch ma zostać wykonany. Następnie jak już posiadamy współrzędne pierwszego znalezionego piksela od tej pozycji za pomocą zmiennej **Rb** i **Nb** w ten sam sposób otrzymujemy współrzędne drugiego piksela. Zmienne **Ra i Rb** przyjmują wartości od zera do siedmiu co daje nam to przedstawiony poniżej ruch skoczka szachowego w celu znalezienia pikseli.

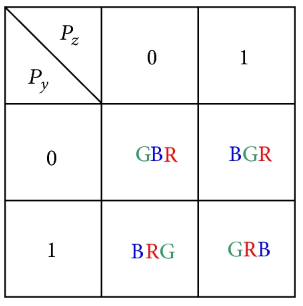
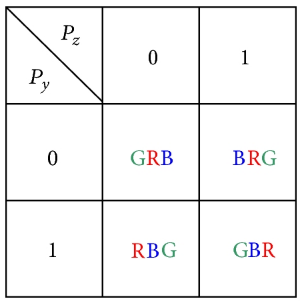
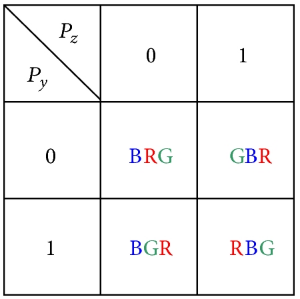


**Rysunek Kierunek ruchu skoczka**

Żródło : <https://www.hindawi.com/journals/mpe/2014/932875/fig1/>

W tym momencie posiadamy współrzędne trzech pikseli obecnie na ,którym się znajdujemy oraz dwa ,które otrzymaliśmy przy pomocy wygenerowanych reguł. Ostatnim krokiem jaki musimy wykonać jest przenoszenie kolorów między znalezionymi pikselami.

Do tego używamy zmiennych **Py** i **Pz** , które przyjmują wartości zero lub jeden i przenoszą wartości tak jak na przedstawionych zdjęciach.

**Czerwony Zielony Niebieski**

Dla przykładu aktualnie szyfrujemy piksel z koloru czerwonego i wylosowaliśmy

Py = 0 i Pz = 1. W takim wypadku do wartości koloru czerwonego z obecnego piksela wpisujemy wartość koloru niebieskiego z trzeciego piksela i odwrotnie. Natomiast wartość drugiego piksela pozostaje bez zmian. Po wykonaniu tych operacji dla wszystkich pikseli otrzymujemy zaszyfrowaną tablice z wartościami obrazu ,którą na samym końcu pozostaje nam zapisać jako zdjęcie.

Deszyfracja:

Przy deszyfracji wprowadzamy poprawny klucz jaki użyliśmy do szyfrowania obrazu oraz zaszyfrowane zdjęcie. Deszyfracja przebiega bardzo podobnie tylko zaczynamy od końca obrazu i ostatniego koloru czyli (niebieski, zielony ,czerwony). Dwa piksele jak w przypadku szyfrowania do zamiany wartości kolorów otrzymujemy najpierw przez zmienna **Rb** z pomocą zmiennej **Nb** a trzeci pikseli przez zmienna **Ra** przy użyciu zmiennej **Na**. Ostatnią różnicą jaką musimy zmienić przy odszyfrowaniu jest zamienić kierunek zmiennej **Rb** i **Ra** na przeciwny. Czyli jak uzyskaliśmy kierunek R3 to w odszyfrowaniu wykonujemy ruch kierunkiem R7 , który jest przeciwny.

**2.4 Lista kroków**

Szyfrowanie:

Dane wejściowe: zdjęcie

Dane wyjściowe: klucz, zaszyfrowane zdjęcie

1 Wczytaj zdjęcie

2 Wygeneruj bity do plików Bits\_A.txt i Bits\_B.txt używając skramblera

3. Oblicz tablice KMR i RNS

4. Podstaw K = 1

5. Jeżeli K = 3 przejdź do kroku 14

6. Podstaw I = 1

7 . Jeżeli I = M, K = K + 1 przejdź do kroku 5

8. Podstaw J = 1

9. Jeżeli J = M , I = I + 1 przejdź do kroku 7

10 Oblicz Reguły(I, J, K)

11. Ia (i,j,k) = Oblicz współrzędne pierwszego piksela(Io(i,j,k), Ra, Na)

12. Ib (i,j,k) = Oblicz współrzędne drugiego piksela(Ia(i,j,k), Rb, NB)

13. TransponujWartościMiedzyKanalamiRGB(Io(i,j,k), Ia(i,j,k), Ib(i,j,k),Py,Pz,k)

13. J = J + 1 przejdź do kroku 9

14. Wyprowadź klucz Wyprowadź zaszyfrowane zdjęcie

15 .Wyprowadź zaszyfrowane zdjęcie

**2.3 Schemat Blokowy**

**3. Algorytm przeszukiwania szachownicy**

3.1 Wstęp

Algorytm przeszukiwania szachownicy jest problemem matematycznym. Problem ten polega na odwiedzeniu każdego pola szachownicy dokładnie jeden raz używając dozwolonych ruchów skoczka szachowego.

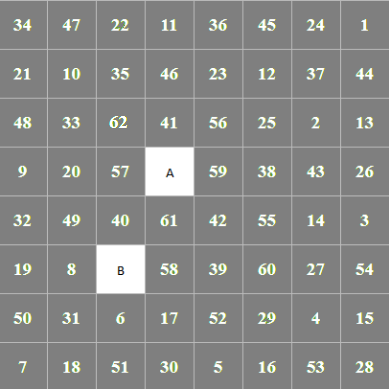
3.3 Rozwiązanie Eulera

Euler przedstawia rozwiązanie problemu przeszukiwania szachownicy dla planszy o wymiarach 8x8, co daje nam szachownice składającą się z 64 pól. Trasa jaką musi pokonać skoczek składa się z 63 ruchów, co skutkuje odwiedzeniem każdego pola tylko jeden raz. Liczba ścieżek jaką może poruszać się skoczek żeby osiągnąć oczekiwany rezultat jest skończona.Ta metoda jest skuteczna kiedy startujemy z dowolnego rogu planszy . Według Eulera rozwiązania problemu przeszukiwania szachownicy można podzielić na dwa rodzaje otwarte i zamknięte.

Pierwsza z nich polega na tym ,że skoczek odwiedza każde pole, ale nie jest w stanie wykonać ruchu z ostatniej pozycji na pole startowe.

Natomiast w drugim typie rozwiązania skoczek może wykonać skok z pola końcowego do początkowego co daje nam możliwość wielokrotnego przechodzenia planszy.

Metoda otwarta wymaga od nas ustawienia konika na polu startowym czyli w dowolnym rogu szachownicy. Następnie używając reguł skoczka szachowego staramy się wykonać jak najwięcej możliwych ruchów pamiętając że nie możemy odwiedzić tego samego pola więcej niż jeden raz. Przy każdym przemieszczeniu się konika musimy zapisywać aktualny numer ruchu a pozostałe pola ,które nie zostały odwiedzone przez nas oznaczamy literami. Wykonujemy te operacje żebyśmy byli w stanie odtworzyć jego ruchy oraz określić, które pola nie zostały odwiedzone. Na rysunku 3 przedstawiona została trasa skoczka, który wykonał 61 ruchów i pozostały dwa nieosiągalne pola.



**Rysunek 3**

W momencie kiedy nie możemy wykonać ruchu musimy określić ,które pola muszą zostać użyte aby umożliwić nam dotarcie do miejsc ,które nie zostały wcześniej odwiedzone są one oznaczone literami A i B. Następnie musimy określić do jakich pól skoczek jest w stanie przejść z ostatniej pozycji.

W wyniku dostajemy numery miejsc takich jak:

21, 47,11,23,59,61,49 oraz 9.

Teraz musimy podzielić drogę od startu do podanych pól oraz sprawdzić czy umożliwi nam to dojście do pola A lub B.

W rezultacie otrzymujemy takie drogi.

1. 1 -> 21 oraz 62 -> 22
2. 1 -> 47 oraz 62 -> 48
3. 1 -> 11 oraz 62 -> 12
4. 1 -> 23 oraz 62 -> 24 Pole 23 prowadzi do miejsca A
5. 1 -> 59 oraz 62 -> 60
6. 1 -> 49 oraz 62 -> 50 Pole 50 prowadzi do miejsca B
7. 1 -> 9 oraz 62 -> 10

Z czego wybieramy dowolną drogę ,która przechodzi przez punkt ,który jest nie odwiedzony.

1 -> 49 oraz 62 -> 50 -> B

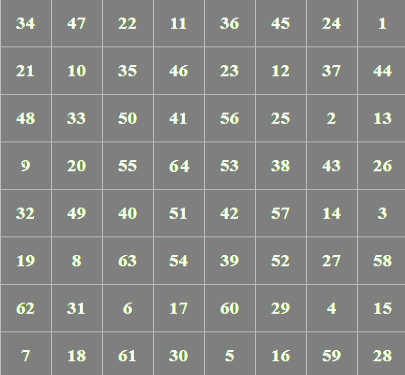
Pole B umożliwia nam przejście na następujące miejsca.

18,50,32,20,A,42,52,30.

Mamy możliwość przejścia z pola B bezpośrednio do pola A.

1-> 49 oraz 62 -> 50 -> B -> A

Teraz mamy kompletną trasę ,która prezentuje się na rysunku 4.



**Rysunek 4**

Metoda zamknięta, warunkiem użycia tej metody jest posiadanie trasy ,która obejmuje wszystkie pola szachownicy.Dla przykładu użyjemy wyniku z metody zamkniętej rysunek 4.

Posiadamy drogę od 1 - > 64 pola.

ostatnie miejsce prowadzi nas do pól takich jak

35,23,25,57,39,63,49,33,

1. 1 -> 35 oraz 64 -> 36
2. 1 -> 23 oraz 64 -> 24
3. 1 -> 25 oraz 64 -> 26
4. 1 -> 57 oraz 64 -> 58
5. 1 -> 39 oraz 64 -> 40
6. 1 -> 49 oraz 64 -> 50
7. 1 -> 33 oraz 64 -> 34

Nie sprawdzamy pola numer 63 ,ponieważ nie ma wpływu na wynik. Wykryte drogi odwracamy.

1. 36 -> 64 oraz 35 -> 1
2. 24 -> 64 oraz 23 -> 1
3. 26 -> 64 oraz 25 -> 1
4. 58 -> 64 oraz 57 -> 1
5. 40 -> 64 oraz 39 -> 1
6. 50 -> 64 oraz 49 -> 1
7. 34 -> 64 oraz 33 -> 1

Teraz musimy zobaczyć gdzie możemy się dostać z pierwszego pola.

Pole 1 prowadzi nas do miejsc 2 i 12

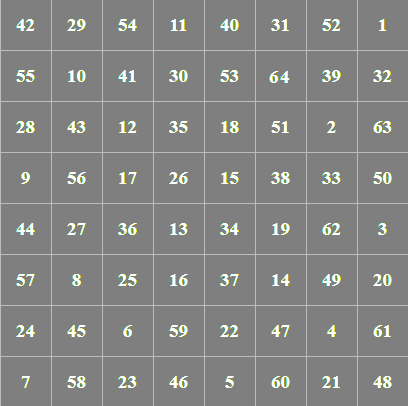
W wyniku dostajemy takie transformacje

1. 36 -> 64 oraz 31 -> 12 oraz 1 -> 11
2. 24 -> 64 oraz 23 -> 12 oraz 1 -> 11
3. 26 -> 64 oraz 25 -> 12 oraz 1 -> 11
4. 58 -> 64 oraz 57 -> 12 oraz 1 -> 11
5. 40 -> 64 oraz 39 -> 12 oraz 1 -> 11
6. 50 -> 64 oraz 49 -> 12 oraz 1 -> 11
7. 34 -> 64 oraz 33 -> 12 oraz 1 -> 11

Możemy zauważyć ,że w 6 trasie pole 50 łączy się z polem 11 możemy więc jedyne co zostało do zrobienia to przekształcenie tej transformacji na:

1 -> 11 oraz 50 -> 64 oraz 49 -> 12

Z każdej trasy ,którą sprawdzaliśmy można osiągnąć wynik różnica polega na ilości transformacji. Wynik tej metody przedstawiony jest na rysunku 5.Możemy teraz zauważyć ,że jesteśmy wstanie z ostatniego pola przejść bezpośrednio na początek.



**Rysunek 5**

Metoda wykorzystująca podział planszy..

(opis…………)

**4. Oprogramowanie**

4.1 Przedstawienie wybranych technologii

Do realizacji projektu zostały wykorzystane technologie takie jak Microsoft Visual Studio, Visual Paradigm oraz Matlab.

4.2 Microsoft Visual Studio

4.3 Visual Paradigm

4.4 Matlab

**6. Opis implementacji**

6.1 Opis użytych technologii i narzędzi

Do stworzenia aplikacji zostało użyte środowisko programistyczne Microsoft Visual Studio 2017 wersja Community. Jest to wersja bezpłatna, która umożliwia tworzenie oprogramowania konsolowego oraz z graficznym interfejsem użytkownika.

Wybrałem to narzędzie, ponieważ posiada mechanizmy refaktoryzacji kodu, zintegrowany debugger co usprawnia znajdowanie błędów w pisanym programie, designer ,który pozwala na łatwe i szybkie tworzenie graficznego interfejsu użytkownika gdyż ma dostęp do elementów graficznych Microsoft Windows oraz pisania testów jednostkowych dzięki, którym programista ma możliwość sprawdzenia poprawności napisanego kodu. Środowisko to jest bardzo popularne co również ułatwia prace, ponieważ w sieci możemy znależć bardzo duże ilości kursów oraz pomocy od innych użytkowników w przypadku ustawień projektu lub błędów. Algorytm do szyfrowania obrazów został zaimplementowany w języku programowania C++ wykorzystując paradygmat programowania obiektowego. Zostały użyty standardowe biblioteki tego języka oraz dodatkowo bilioteka OpenCV, w której zawarte są zestawy funkcji umożliwiające prace do obróbki obrazów.

6.2 Opis klas oraz metod

**Bibliografia**

[1] Microsoft Visual Studio IDE, https://docs.microsoft.com/pl-pl/visualstudio/ide/visual-studio-ide?view=vs-2017