

Sprawozdanie z pracowni specjalistycznej

Bezpieczeństwo Sieci Komputerowych

Ćwiczenie numer: 3-4

Temat: *Implementacja podstawowych modułów kryptograficznych*

Wykonujący ćwiczenie: Wioletta Rushnitskaya, Marcin Paulouski, Igor Balyka

Studia dzienne

Kierunek: Informatyka

Semestr: VI

Grupa zajęciowa: Grupa PS 9

Prowadzący ćwiczenie: mgr inż. Katarzyna Borowska

Data wykonania ćwiczenia:

24.03.2022

Treść zadania

Zaimplementuj szyfrowanie oraz deszyfrowanie za pomocą wszystkich algorytmów znajdujących się w pliku [Sposób implementacji zadań](#) (w sumie 6 algorytmów).

Krótkie wprowadzenie teoretyczne

Kryptografia - nauka o tworzeniu szyfrów, zabezpieczaniu wiadomości przy pomocy kluczy szyfrujących.

Kryptoanaliza - nauka o łamaniu szyfrów, tzn. o odczytywaniu zaszyfrowanej wiadomości bez znajomości klucza.

Szyfr – to algorytm służący do zaszyfrowania i odszyfrowania wiadomości.

Szyfry dzieli się na:

- **Symetryczne** – wykorzystują do szyfrowania i odszyfrowania ten sam klucz szyfrujący, który w związku z tym musi być chroniony
- **Asymetryczne** – wykorzystują parę kluczy szyfrujących, jeden do szyfrowania a drugi do odszyfrowania. Działają zwykle dużo wolniej niż szyfry symetryczne.

Szyfry przestawieniowe - polegające na przestawieniu znaków w tekście, np. Szyfr Atbasha lub Szyfr Cezara.

Szyfry podstawieniowe – polegające na podstawieniu znaku lub sekwencji znaków w miejsce szyfrowanego znaku lub sekwencji znaków.

Podział pracy

Algorytmy Rail Fence i Szyfrowanie Vigenere'a - Wioletta Rushnitskaya.

Algorytmy Przestawienie macierzowe. Przykład B i Szyfr Cezara - Marcin Paulouski.

Algorytmy Przestawienie macierzowe. Przykład A i C - Igor Balyka.

Link do repozytorium na githubie: <https://github.com/VioRush/ModulyKryptograficzne>

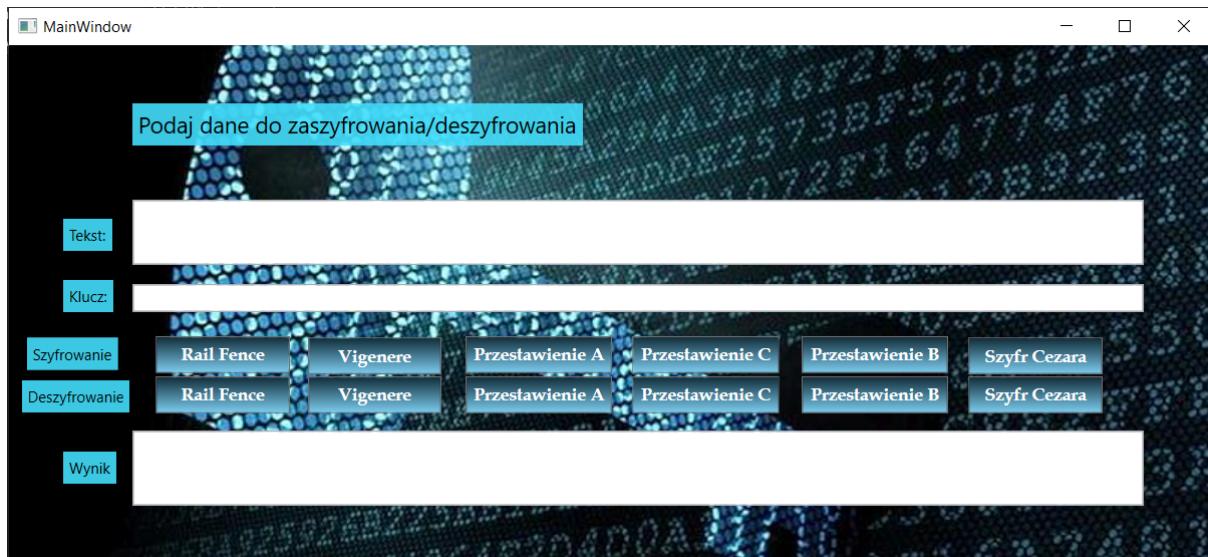
Realizacja zadania

Wykorzystane oprogramowanie: Visual Studio Community 2019

Technologia: WPF (.NET Framework)

Język: C#

Interfejs użytkownika:



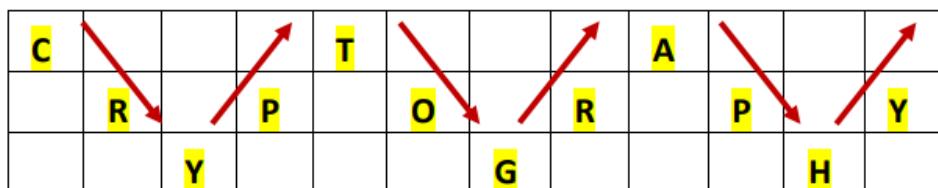
Rail Fence

Przykład działania algorytmu:

Dane wejściowe:

M = CRYPTOGRAPHY

n = 3



Wynik:

C = CTARPORPYGH

- **Szyfrowanie**

Zaczęto od realizacji szyfrowania za pomocą danego algorytmu według powyższego schematu. Kod:

```

private void RailFence_button(object sender, RoutedEventArgs e)
{
    WczytajDane();
    int w = M.Length;
    if (!int.TryParse(klucz, out k) || klucz == ".") || Convert.ToInt32(klucz) < 2) //sprawdzenie, czy podany klucz jest liczbą > 1
    {
        Klucz.Background = Brushes.Red;
        Klucz.ToolTip = "Niepoprawnie podano klucz! Podaj liczbę całkowitą.";
    }
    else
    {
        Klucz.Background = Brushes.White;
        Klucz.ToolTip = "";
        k = Convert.ToInt32(klucz);
        tab = new char[k, w];
        bool do_dolu = false;
        int wiersz = 0;

        for (int i = 0; i < w; i++) //wypełnienie utworzonej tabeli według schematu
        {
            if (wiersz == 0 || wiersz == k - 1)
            {
                do_dolu = !do_dolu; //zmiana kierunku, gdy dojdzie do ostatniego albo pierwszego wierszu
            }

            tab[wiersz, i] = M[i];

            if (do_dolu == true) { wiersz++; }
            else { wiersz--; }
        }
    }

    C = null;
    for (int i = 0; i < k; i++) //odczytanie tekstu zaszyfrowanego według schematu wierszami
    {
        for (int j = 0; j < w; j++)
        {
            if (tab[i, j] > 0)
            {
                C += tab[i, j];
            }
        }
    }

    TekstWynikowy.Text = C;
}

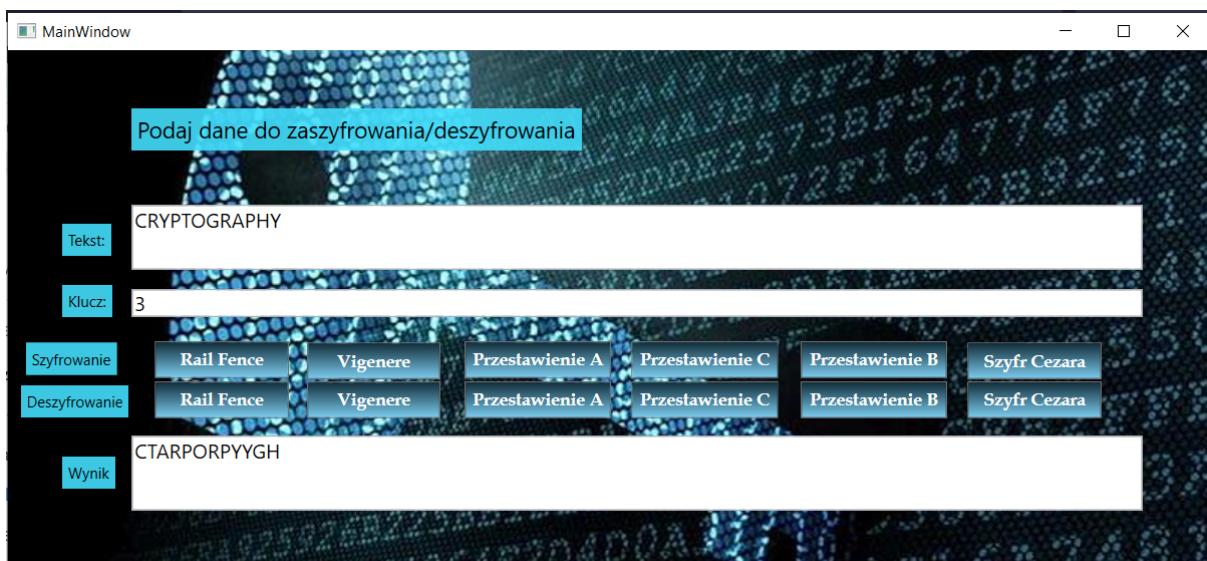
```

Na początku wczytano podane przez użytkownika dane: tekst do zaszyfrowania M i klucz klucz. Następnie sprawdzono, czy wprowadzony klucz jest liczbą. Jeśli nie, wyświetlany jest błąd. Natomiast jeśli klucz jest liczbą, to zaczyna się algorytm szyfrowania. Pierwszym krokiem utworzono tablicę o wymiarach klucz x liczba liter w tekście. Następnie jest wypełniana według schematu, czyli po przekątnych. Zaszyfrowana wartość jest odczytywana kolejno wierszami.

Wprowadzono przykładowe dane i sprawdzono działanie algorytmu. Wyniki przedstawiono na poniższym screenie.

Przykład 1.

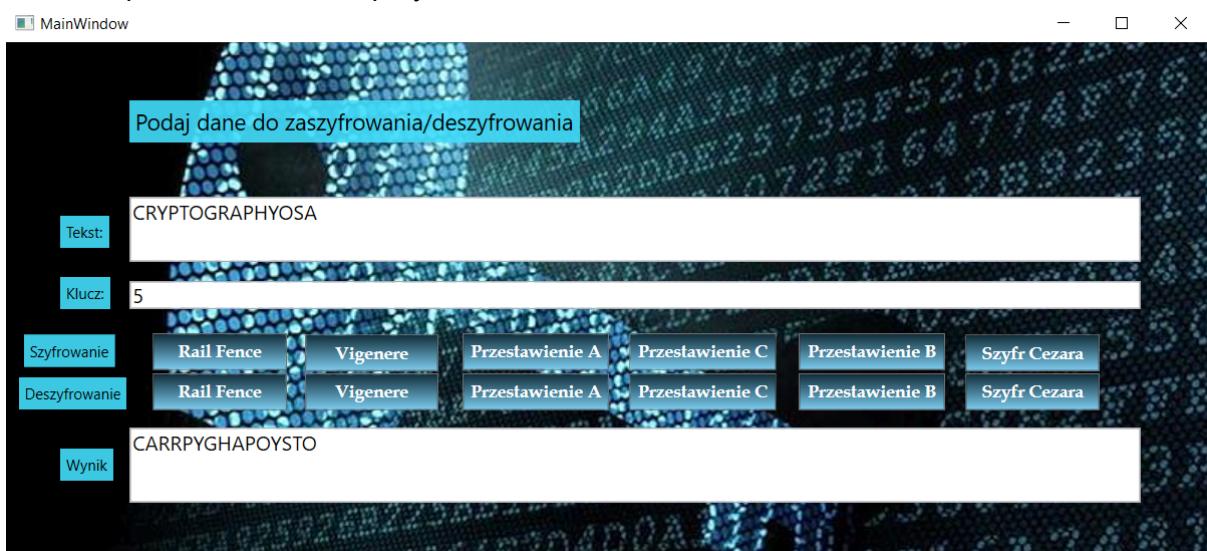
Wprowadzono dane podane powyżej w przykładzie działania algorytmu.



Wynik jest zgodny z wynikiem w przykładzie działania.

Przykład 2.

Wprowadzono nowe przykładowe dane $M = \text{CRYPTOGRAPHYOSA}$ i $k = 5$.



Aby sprawdzić, czy wynik jest poprawny, obliczono go teoretycznie:

C								A						
	R							R		P				
		Y					G			H				A
			P		O						Y		S	
				T								O		

Wynik: CARRPYGHAPOYSTO.

Wynik obliczony teoretycznie jest taki sam jak wynik przedstawiony na screenie.

- **Deszyfrowanie**

Następnie zrealizowano deszyfrowanie za pomocą danego algorytmu według powyższego schematu. Kod:

```
private void RailFenceDeszyfrowanie_button(object sender, RoutedEventArgs e)
{
    WczytajDane();
    int w = M.Length;
    if (!int.TryParse(klucz, out k) || klucz == ".") || Convert.ToInt32(klucz) < 2) //sprawdzenie, czy podany klucz jest liczbą
    {
        Klucz.Background = Brushes.Red;
        Klucz.ToolTip = "Niepoprawnie podano klucz! Podaj liczbę całkowitą.";
    }
    else
    {
        Klucz.Background = Brushes.White;
        Klucz.ToolTip = "";
        k = Convert.ToInt32(klucz);
        tab = new char[k, w];
        bool do_dolu = false;
        int wiersz = 0;

        for (int i = 0; i < w; i++) //oznaczenie miejsc, gdzie mają być litery znakiem kropki
        {
            if (wiersz == 0 || wiersz == k - 1)
            {
                do_dolu = !do_dolu;
            }

            tab[wiersz, i] = '.';
            if (do_dolu == true) { wiersz++; }
            else { wiersz--; }
        }
    }

    C = null;
    int m = 0;

    for (int i = 0; i < k; i++) //uzupełnienie utworzonej tabeli wierszami, gdzie jest kropka
    {
        for (int j = 0; j < w; j++)
        {
            if (tab[i, j] == '.')
            {
                tab[i, j] = M.ElementAt(m);
                m++;
            }
        }
    }

    do_dolu = false;
    wiersz = 0;

    for (int i = 0; i < w; i++) //odczytanie oryginalnego tekstu przechodząc po przekątnych
    {
        if (wiersz == 0 || wiersz == k - 1)
        {
            do_dolu = !do_dolu;
        }

        C += tab[wiersz, i];
        if (do_dolu == true) { wiersz++; }
        else { wiersz--; }
    }

    TekstWynikowy.Text = C;
}
```

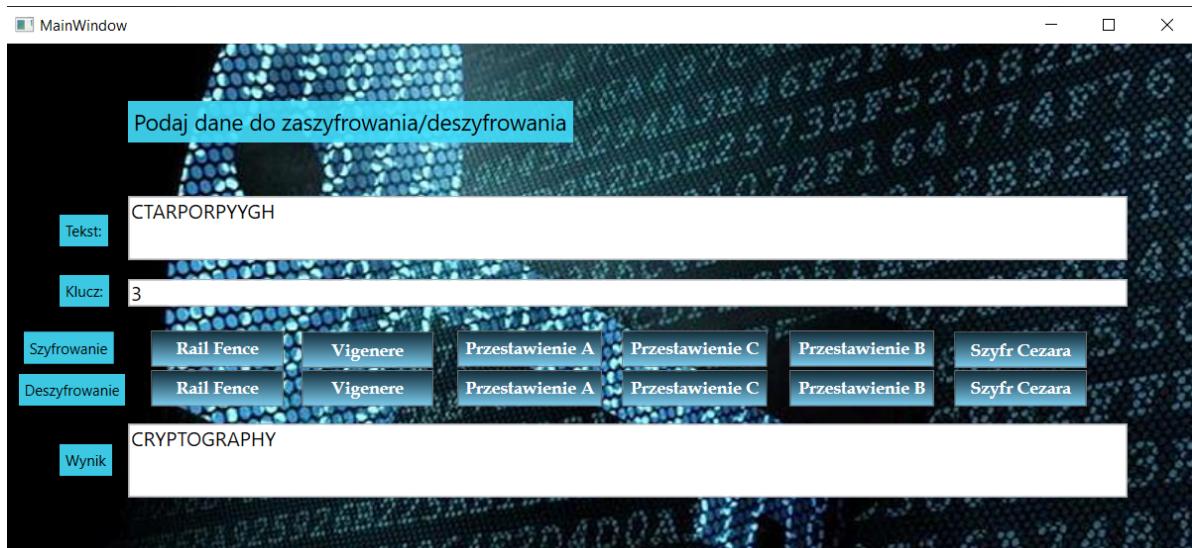
Na początku wczytano podane przez użytkownika dane: tekst do rozszyfrowania M i klucz klucz. Następnie sprawdzono, czy wprowadzony klucz jest liczbą. Jeśli nie, wyświetlany jest błąd. Natomiast jeśli klucz jest liczbą, to zaczyna się algorytm deszyfrowania. Pierwszym krokiem utworzono tablicę o wymiarach klucz x liczba liter w tekście. Następnie przechodzimy po przekątnych, aby zaznaczyć miejsca gdzie mają być

liter. Dalej wypełniono tablicę przechodząc po niej wierszami i wstawiając kolejno litery tekstu w miejsca oznaczone w poprzednim kroku. Oryginalna wartość jest odczytywana po przekątnym, jak była szyfrowana w szyfrowaniu.

Wprowadzono przykładowe dane i sprawdzono działanie algorytmu. Wynik przedstawiono na poniższym screenie.

Przykład 1.

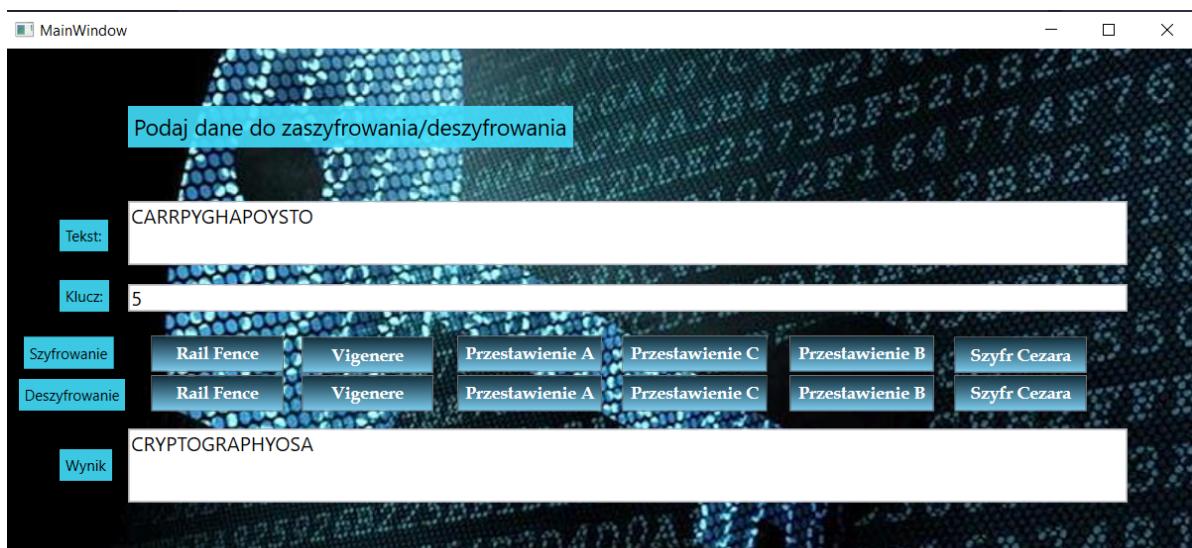
Jako tekst zaszyfrowany wprowadzono wynik z przykładu 1 z szyfrowania. Klucz podano ten sam.



Wynik jest zgodny z wartością oryginalną w przykładzie 1 szyfrowania.

Przykład 2.

Jako tekst zaszyfrowany wprowadzono wynik z przykładu 2 z szyfrowania. Klucz podano ten sam.



Wynik jest zgodny z wartością oryginalną w przykładzie 2 szyfrowania.

Przetwarzanie macierzowe. Przykład A

- **Szyfrowanie**

Kod:

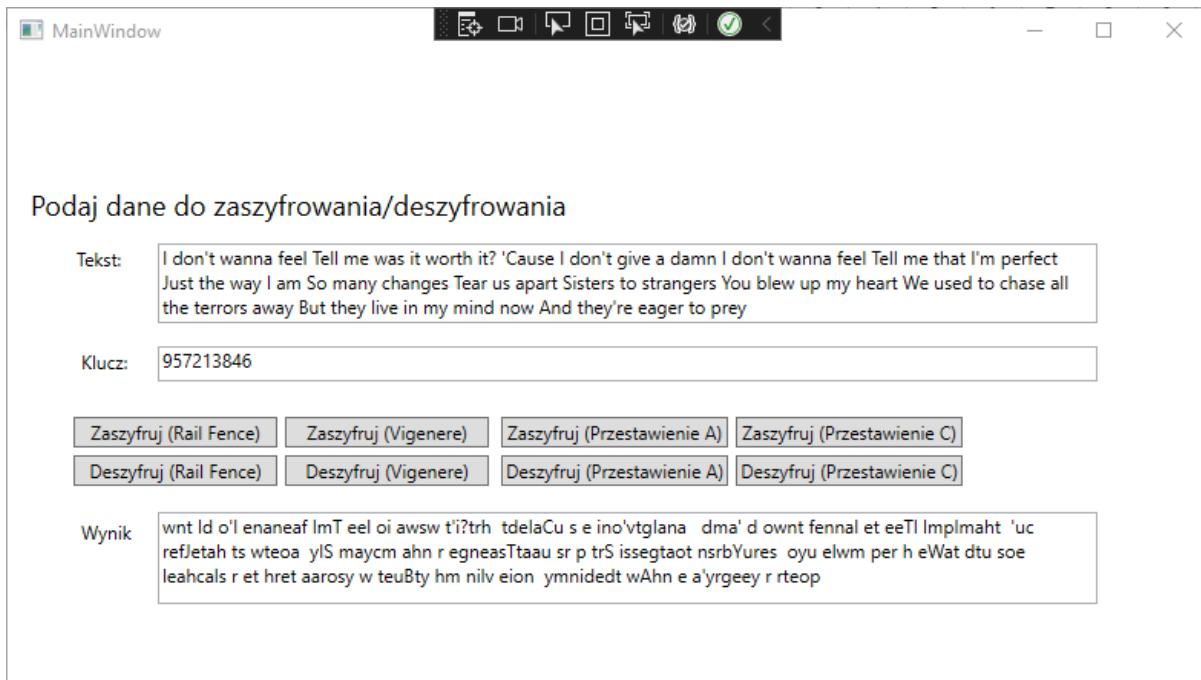
```
private void PrzetwarzanieA_button(object sender, RoutedEventArgs e)
{
    WczytajDane();
    C = null;

    for (int i = 0; i < Math.Ceiling((double)M.Length / (double)klucz.Length); i++)
        for (int j = 0; j < klucz.Length; j++)
            if ((i * klucz.Length + (int)Char.GetNumericValue(klucz[j]) - 1) < M.Length) C += M[i * klucz.Length + (int)Char.GetNumericValue(klucz[j]) - 1];

    TekstWynikowy.Text = C;
}
```

Ten algorytm nawet nie potrzebuje macierzy. Bierzemy pierwsze (długość klucza) liter i przestawiamy. Zatem drugie (długość klucza) liter i tak dalej. Co robi go prostym do deszyfrowania.

Przykład działania algorytmu:



- **Deszyfrowanie**

Udało się znaleźć że deszyfrowanie pracuje tym samym algorytmem z minimalnymi zmianami. Główne zrobić nowy klucz tym cyklem.

```
string swapKlucz = null;

for (int i = 0; i < klucz.Length; i++)
    for (int j = 0; j < klucz.Length; j++)
        if ((int)Char.GetNumericValue(klucz[j]) == i+1)
            swapKlucz += (j+1).ToString();
```

Przykład działania algorytmu:

MainWindow

Podaj dane do zaszyfrowania/deszyfrowania

Tekst: wnt Id o'l enaneaf lmT eel oi awsw t'i?trh tdelaCu s e ino'vtglana dma' d ownt fennal et eeTl Implmaht 'uc refleatah ts wteoa yIS maycm ahn r egeneasTtaau sr p trS issegtaot nsrbYures oyu elwm per h eWat dtu soe leahcals r et hret aarosy w teuBty hm nilv eion ymnidedt wAhn e a'yrgeey r rteop

Klucz: 957213846

Zaszyfruj (Rail Fence) Zaszyfruj (Vigenere) Zaszyfruj (Przetwarzanie A) Zaszyfruj (Przetwarzanie C)
 Deszyfruj (Rail Fence) Deszyfruj (Vigenere) Deszyfruj (Przetwarzanie A) Deszyfruj (Przetwarzanie C)

Wynik I don't wanna feel Tell me was it worth it? 'Cause I don't give a damn I don't wanna feel Tell me that I'm perfect Just the way I am So many changes Tear us apart Sisters to strangers You blew up my heart We used to chase all the terrors away But they live in my mind now And they're eager to prey

Przetwarzanie macierzowe. Przykład B

Przykład działania algorytmu:

PRZYKŁAD B

Dane wejściowe:

M = HERE IS A SECRET MESSAGE ENCIPHERED BY TRANSPOSITION

Key = CONVENIENCE



C	O	N	V	E	N	I	E	N	C	E
1	10	7	11	3	8	6	4	9	2	5
H	E	R	E	I	S	A	S	E	C	R
E	T	M	E	S	S	A	G	E	E	N
C	I	P	H	E	R	E	D	B	Y	T
R	A	N	S	P	O	S	I	T	I	O
N										

Wynik:

C = HECRN CEYI ISEP SGDI RNTO AAES RMPN SSRO EEBT ETIA EEHS

- **Szyfrowanie**

Zaczęto od realizacji szyfrowania za pomocą danego algorytmu według powyższego schematu. Kod:

```
public string SzyfrowaniePrzykladB()
{
    WczytajDane();
    string message = M;
    int kolumn = klucz.Length;
    int wiersz = Convert.ToInt32(Math.Ceiling(Convert.ToDouble(message.Length) / kolumn));
    string szyfr = "";

    char[,] message_char = new char[wiersz, kolumn];
    int value = 0;

    for (int i = 0; i < wiersz; i++)
        for (int j = 0; j < kolumn; j++)
    {
        if (value >= message.Length)
            continue;
        if (message[value] == ' ')
            value++;
        message_char[i, j] = message[value];
        value++;
    }

    for (int i = 0; i < alfabet.Length; i++)
        for (int j = 0; j < klucz.Length; j++)
            if (alfabet[i] == klucz[j])
                for (int g = 0; g <= wiersz; g++)
    {
        if (g == wiersz)
        {
            szyfr += " ";
            break;
        }
        szyfr += message_char[g, j];
    }
}

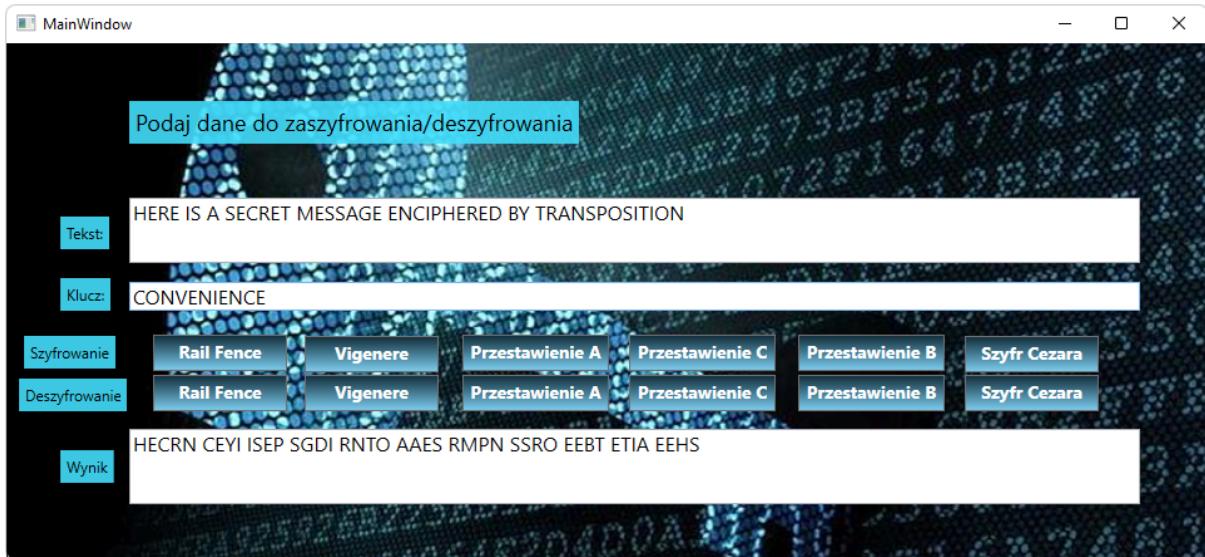
return szyfr;
}
```

Na początku wczytane są dane i tworzona jest macierz na podstawie kolumn i wierszy. Następnie macierz uzupełniamy tekstem który był podany przez użytkownika. Przechodzimy po alfabetie i szukujemy pierwszego wystąpienia kiedy litery będą zgadzały się w alfabetie i w kluczu podanego przez użytkownika. Zatem wypisujemy kolumnę pod indeksem gdzie zgadzali się litera alfabetu i klucza.

Wprowadzono przykładowe dane i sprawdzono działanie algorytmu. Wyniki przedstawiono na poniższym screenie.

Przykład:

Wprowadzono dane podane powyżej w przykładzie działania algorytmu.



Wynik jest zgodny z wynikiem w przykładzie działania.

- **Deszyfrowanie**

Następnie zrealizowano deszyfrowanie za pomocą danego algorytmu według powyższego schematu.

Kod:

```
public string DeszyfrowaniePrzykladB()
{
    WczytajDane();
    string message = M;
    int kolumn = klucz.Length;
    int wiersz = Convert.ToInt32(Math.Ceiling(Convert.ToDouble(message.Length) / kolumn));
    string deszyfr = "";

    char[,] message_char = new char[wiersz, kolumn];

    int value = 0;

    for (int i = 0; i < alfabet.Length; i++)
        for (int j = 0; j < klucz.Length; j++)
            if (alfabet[i] == klucz[j])
                if (message[value] == ' ')
                {
                    value++;
                    for (int g = 0; g < wiersz - 1; g++)
                    {
                        message_char[g, j] = message[value];
                        value++;
                    }
                }
            else
            {
                for (int g = 0; g < wiersz; g++)
                {
                    message_char[g, j] = message[value];
                    value++;
                }
            }
}
```

```

    for (int i = 0; i < wiersz; i++)
        for (int j = 0; j < kolumn; j++)
    {
        deszyfr += message_char[i, j];
    }

    return deszyfr;
}

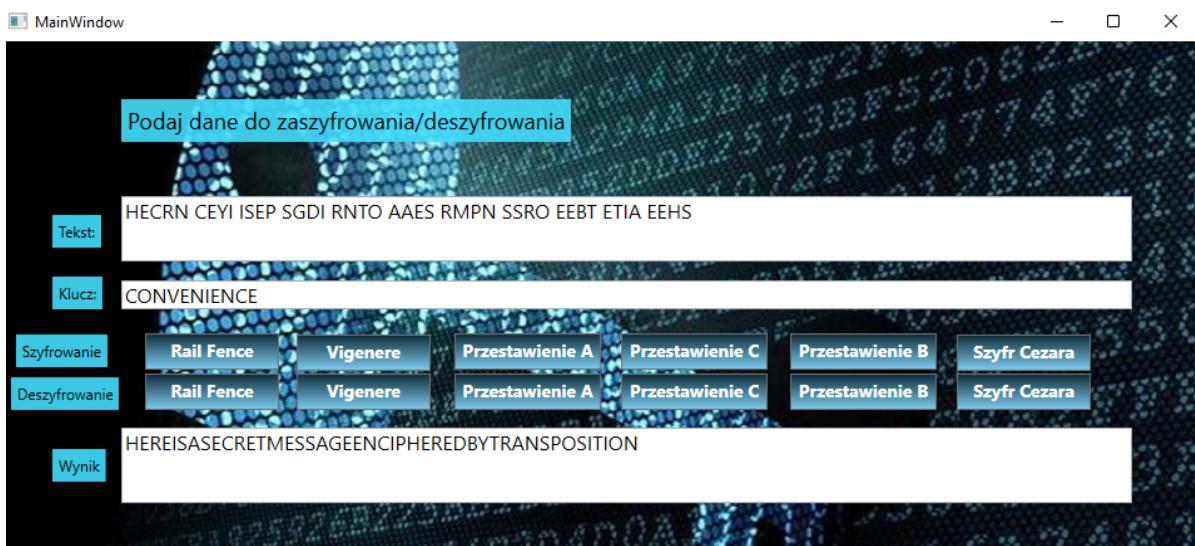
```

Na początku tak samo wczytane są dane i tworzona jest macierz na podstawie kolumn i wierszy. Następnie przechodzimy po alfabetie i szykujemy pierwszego wystąpienia kiedy litery będą zgadzały się w alfabetie i w kluczu podanego przez użytkownika. Kiedy litery będą zgadzały się, jeżeli litera będzie równała się spacji, przesuwamy o jeden znak podany przez użytkownika tekst i wpisujemy całą kolumnę. Jeżeli nie będzie równa spacji to po prostu wpisujemy całą kolumnę. Zatem zapisujemy całą zawartość macierzy po kolei w nową zmienną i zwracamy ją.

Wprowadzono przykładowe dane i sprawdzono działanie algorytmu. Wynik przedstawiono na poniższym screenie.

Przykład:

Jako tekst zaszyfrowany wprowadzono wynik z przykładu 1 z szyfrowania. Klucz podano ten sam.



Wynik jest zgodny z wartością oryginalną w przykładzie 1 szyfrowania.

Przetawienie macierzowe. Przykład C

Ten algorytm już potrzebuje tabele i jest bardziej trudny dla deszyfrowania, ale tak samo ma swoje wady. Klucz nie jest słowem, ale liczbami porządku alfabetycznego. Czyli posłanie zaszyfrowane kluczem AECDΒ może być deszyfrowane kluczem BFDEC.

- **Szyfrowanie**

Kod:

```
private void PrzestawienieC_button(object sender, RoutedEventArgs e)
{
    WczytajDane();
    //M = "HERE IS A SECRET MESSAGE ENCIPHERED BY TRANSPOSITION";
    //klucz = "CONVENIENCE";

    C = null;
    Char[,] tab = new char[M.Length + klucz.Length, M.Length + klucz.Length];
    int[] values = new int[klucz.Length];

    for (int i = 0; i <= klucz.Length; i++)
    {
        Char min = Char.MaxValue;
        int minPos = 0;
        for (int j = 0; j < klucz.Length; j++)
        {
            if ((klucz[j] < min) && (values[j] == 0)) { min = klucz[j]; minPos = j; }
        }
        values[minPos] = i;
    }

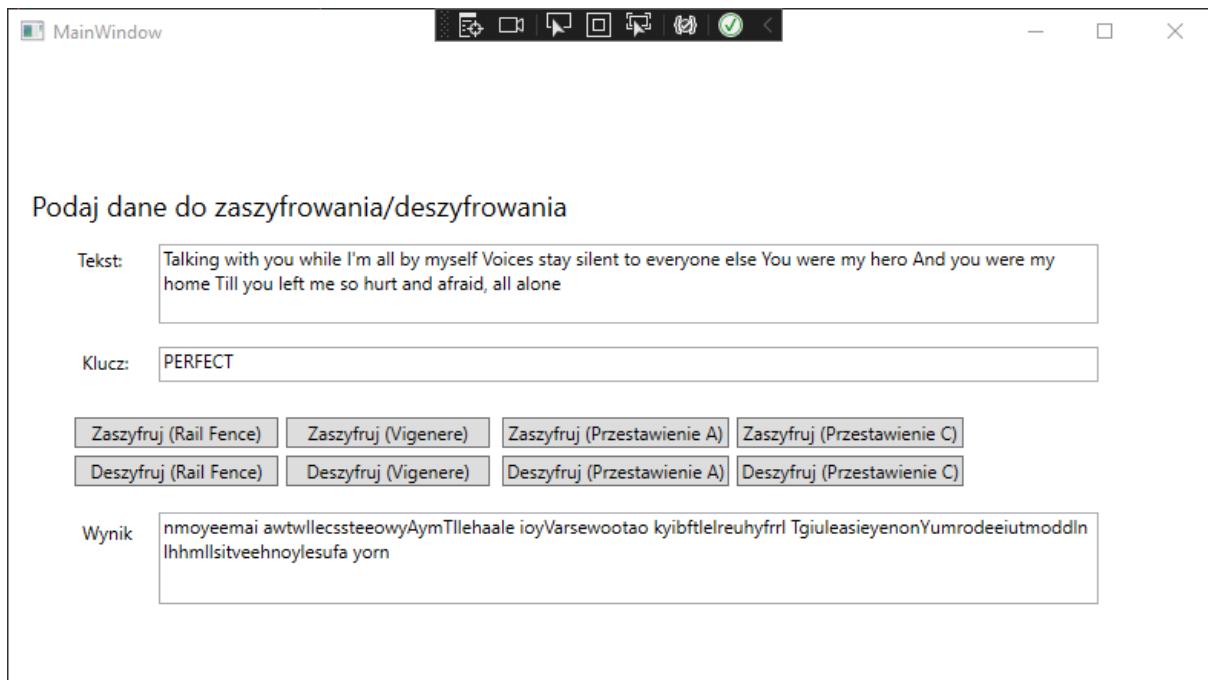
    int element = 0;
    int line = 0;
    int column = 0;
    int exceed = 0;

    while (element < M.Length)
    {
        if (Char.IsLetter(M[element]))
        {
            tab[line + klucz.Length * exceed, column] = M[element];
            element++;
            if (line == values[column]-1)
            {
                column = 0;
                line++;
            }
            else
            {
                column++;
            }
            if (line >= klucz.Length)
            {
                column = 0;
                line = 0;
                exceed++;
            }
        }
        else element++;
    }

    for (int i = 0; i < klucz.Length; i++)
    {
        for (int j = 0; j < klucz.Length; j++)
        {
            if (values[j] == i+1)
            {
                for (int k = 0; k < klucz.Length * (exceed + 1); k++)
                {
                    if (Char.IsLetter(tab[k, j])) C += tab[k, j];
                }
                C += ' ';
            }
        }
    }

    TekstWynikowy.Text = C;
}
```

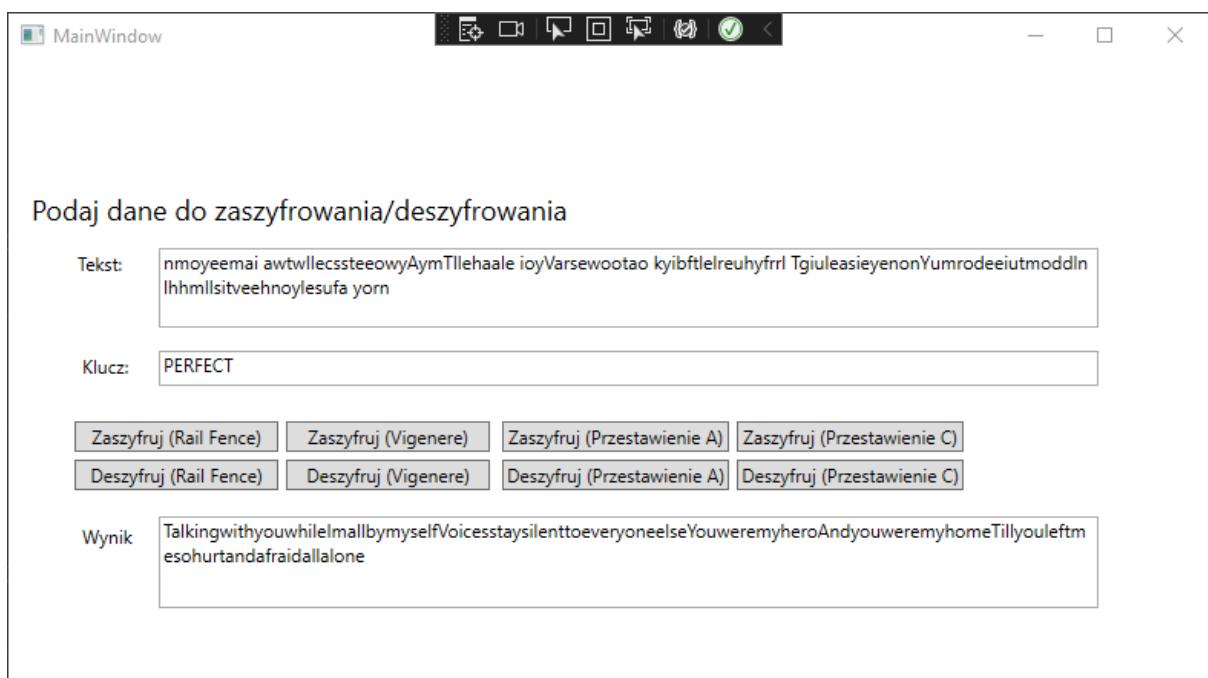
Przykład działania algorytmu:



- **Deszyfrowanie**

Najtrudniejsze było to że dla deszyfrowania potrzebna ta sama struktura tabeli jaką była wykorzystana przy szyfrowaniu. Prostym rozwiązaniem stało ponownie uruchomienie algorytmu szyfrowania dla otrzymania tej struktury.

Przykład działania algorytmu:



Szyfr Cezara

Przykład działania algorytmu:

Szyfrowanie: $c = (a + k) \bmod n$

Deszyfrowanie: $a = [c + (n - k)] \bmod n$

gdzie:

n – liczba znaków w alfabetie

k – klucz

a – znak do zaszyfrowania (test jawny)

c – znak zaszyfrowany (szyfrogram)

Dane wejściowe:

M = CRYPTOGRAPHY

K = 3

Wynik:

C = FUBSWRJUDSKB

- **Szyfrowanie**

Zaczęto od realizacji szyfrowania za pomocą danego algorytmu według powyższego schematu.

Kod:

```
public string Szyfrowanie(string inp)
{
    WczytajDane();
    StringBuilder code = new StringBuilder();
    string message = M;

    for (int i = 0; i < message.Length; i++)
        for (int j = 0; j < alfabet.Length; j++)
            if (message[i] == alfabet[j])
                code.Append(alfabet[(j + Convert.ToInt32(klucz)) % alfabet.Length]);

    return code.ToString();
}
```

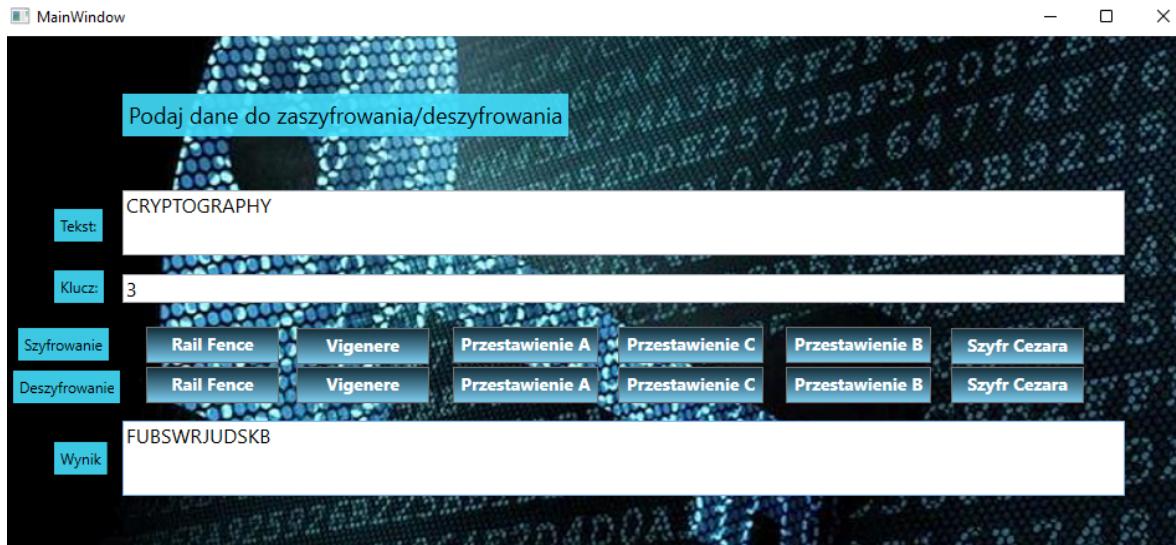
Na początku wczytano podane przez użytkownika dane: tekst do zaszyfrowania M i klucz. Następnie sprawdzono, czy wprowadzony klucz jest liczbą. Jeśli nie, wyświetlany jest błąd. Natomiast jeśli klucz jest liczbą, to zaczyna się algorytm szyfrowania. Algorytm działa na podstawie algorytmu danego na screenie. Szyfrowanie:

$$c = (a + k) \bmod n.$$

Wprowadzono przykładowe dane i sprawdzono działanie algorytmu. Wyniki przedstawiono na poniższym screenie.

Przykład:

Wprowadzono dane podane powyżej w przykładzie działania algorytmu.



Wynik jest zgodny z wynikiem w przykładzie działania.

- **Deszyfrowanie**

Następnie zrealizowano deszyfrowanie za pomocą danego algorytmu według powyższego schematu.

Kod:

```
public string Deszyfrowanie(string inp)
{
    WczytajDane();
    StringBuilder code = new StringBuilder();
    string message = M;
    int k = Convert.ToInt32(klucz);

    for (int i = 0; i < message.Length; i++)
        for (int j = 0; j < alfabet.Length; j++)
            if (message[i] == alfabet[j])
                code.Append(alfabet[(j - k + alfabet.Length) % alfabet.Length]);

    return code.ToString();
}
```

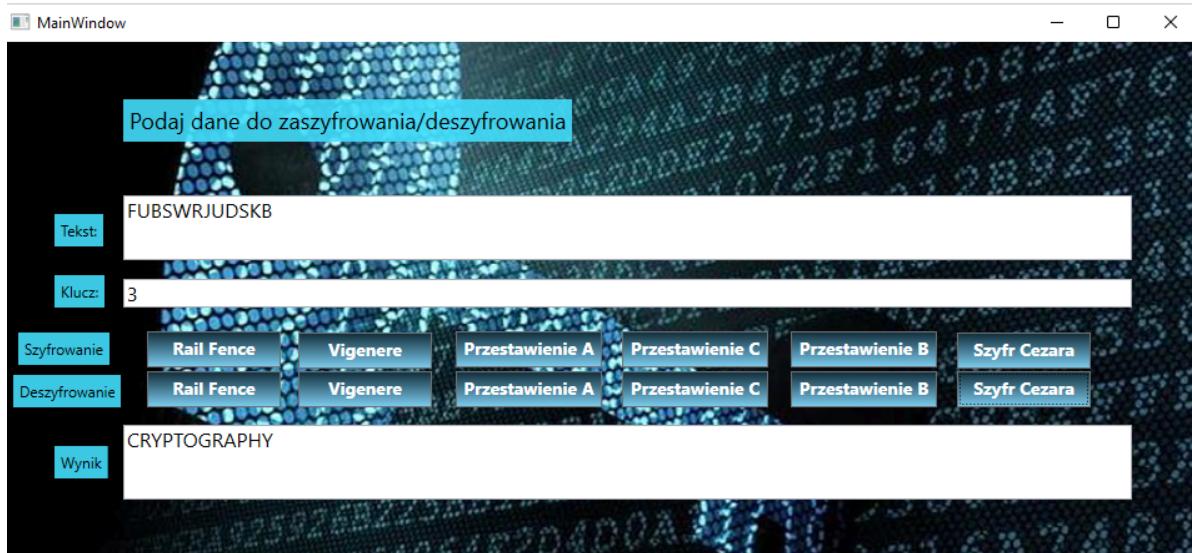
Na początku wczytano podane przez użytkownika dane: tekst do zaszyfrowania M i klucz. Następnie sprawdzono, czy wprowadzony klucz jest liczbą. Jeśli nie, wyświetlany jest błąd. Natomiast jeśli klucz jest liczbą, to zaczyna się algorytm szyfrowania. Algorytm działa na podstawie algorytmu danego na screenie. Deszyfrowanie:

$$a = [c + (n - k)] \bmod n.$$

Wprowadzono przykładowe dane i sprawdzono działanie algorytmu. Wyniki przedstawiono na poniższym screenie.

Przykład:

Wprowadzono dane podane powyżej w przykładzie działania algorytmu.



Wynik jest zgodny z wartością oryginalną w przykładzie 1 szyfrowania.

Szyfrowanie Vigenere'a

Przykład działania algorytmu:

Klucz	Tekst																									
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Dla litery tekstu jawnego a i klucza k, zaszyfrowany tekst c jest literą w kolumnie a i wierszu k.

Dla szyfrogramu c, plaintext a jest kolumną zawierającą c w wierszu k.

Dane wejściowe:

M = CRYPTOGRAPHY

K = BREAKBREAKBR

Wynik:

C = DICPDGXVAZIP

• Szyfrowanie

Zaczęto od realizacji szyfrowania za pomocą danego algorytmu według powyższego schematu. Kod:

```
private void Vigenere_button(object sender, RoutedEventArgs e)
{
    WczytajDane();

    C = null;

    if (int.TryParse(klucz, out k)) //sprawdzenie, czy podany klucz nie jest liczbą
    {
        Klucz.Background = Brushes.Red;
        Klucz.ToolTip = "Niepoprawnie podano klucz! Podaj napis.";
    }
    else
    {
        Klucz.Background = Brushes.White;
        Klucz.ToolTip = "";
        for (int i = 0; i < M.Length; i++) //przechodząc po wartościach tekstu oryginalnego podmieniamy litery
        {
            a = alfabet.IndexOf(char.ToUpper(M.ElementAt(i)));
            b = alfabet.IndexOf(char.ToUpper(klucz.ElementAt(i % klucz.Length)));

            C += alfabet.ElementAt((a + b) % alfabet.Length); //reszta z dzielenia sum indeksów liter tekstu i klucza daje indeks litery
                                                               //na którą podmieniamy
        }

        TekstWyjnikowy.Text = C;
    }
}
```

Na początku wczytano podane przez użytkownika dane: tekst do rozszyfrowania M i klucz klucz. Następnie sprawdzono, czy wprowadzony klucz nie jest liczbą. Jeśli jest liczbą - wyświetlany jest błąd. Natomiast jeśli klucz nie jest liczbą, to zaczyna się algorytm szyfrowania. Zamiast tworzyć tabele i zaszyfrować tekst za pomocą niej, zauważono, że reszta z dzielenia sumy indeksów odpowiednich liter tekstu oryginalnego i klucza daje indeks litery zaszyfrowanej. Dlatego rozwiązano zadanie za pomocą tego wzoru.

Wprowadzono przykładowe dane i sprawdzono działanie algorytmu. Wynik przedstawiono na poniższym screenie.

Przykład 1.

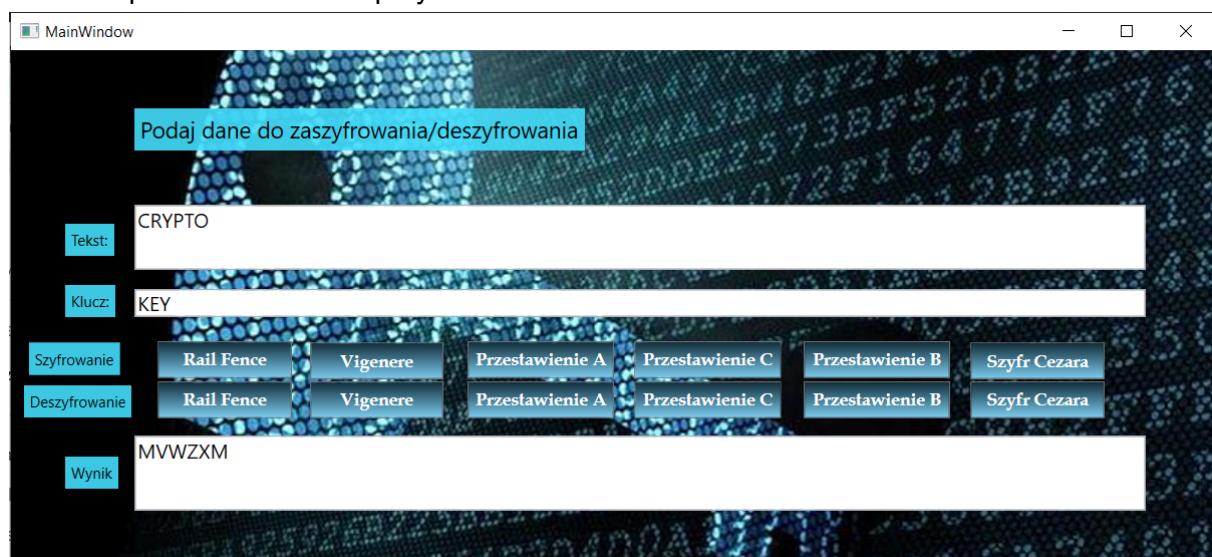
Wprowadzono dane podane powyżej w przykładzie działania algorytmu.



Wynik jest zgodny z wynikiem w przykładzie działania.

Przykład 2.

Wprowadzono nowe przykładowe dane M = CRYPTO i k = KEY.



Aby sprawdzić, czy wynik jest poprawny, obliczono go teoretycznie:

Klucz	Tekst																									
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Wyniki są zgodne.

• Deszyfrowanie

Następnie zrealizowano deszyfrowanie za pomocą danego algorytmu według powyższego schematu. Kod:

```
private void VigenereDeszyfrowanie_button(object sender, RoutedEventArgs e)
{
    WczytajDane();

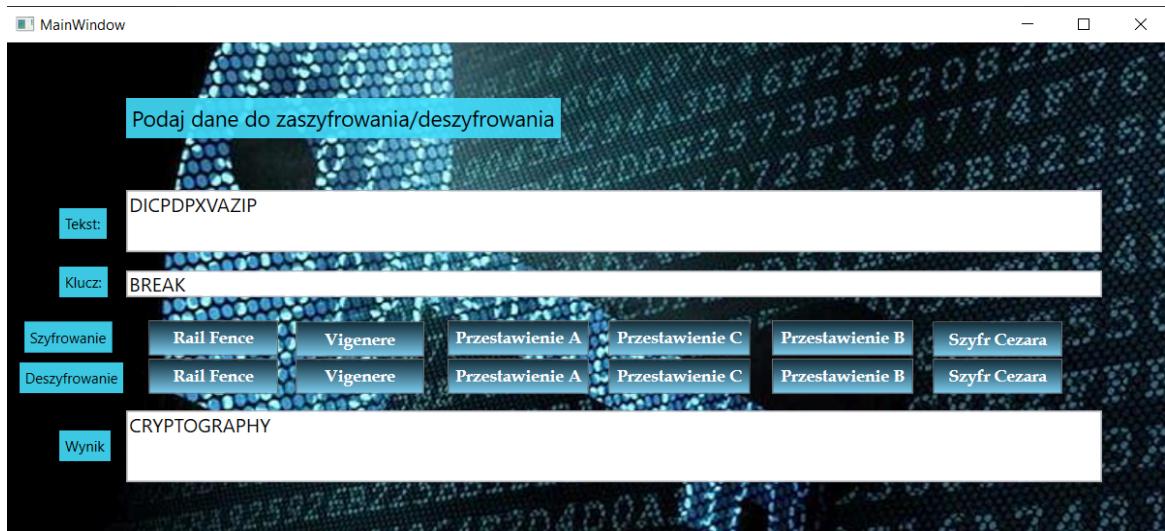
    C = null;
    int r;
    if (int.TryParse(klucz, out k)) //sprawdzenie, czy podany klucz nie jest liczbą
    {
        Klucz.Background = Brushes.Red;
        Klucz.ToolTip = "Niepoprawnie podano klucz! Podaj napis.";
    }
    else
    {
        Klucz.Background = Brushes.White;
        Klucz.ToolTip = "";
        for (int i = 0; i < M.Length; i++) //przechodząc po wartościach zasyfrowanego tekstu podmieniamy litery
        {
            a = alfabet.IndexOf(char.ToUpper(M.ElementAt(i)));
            b = alfabet.IndexOf(char.ToUpper(klucz.ElementAt(i % klucz.Length)));
            r = a - b;
            if (r < 0) //w zależności od znaku różnicy indeksów stosujemy tą albo inną formułę
            {
                C += alfabet.ElementAt((r + alfabet.Length) % alfabet.Length);
            }
            else
            {
                C += alfabet.ElementAt(r % alfabet.Length);
            }
        }
        TekstWynikowy.Text = C;
    }
}
```

Kod deszyfrowania jest bardzo podobny do szyfrowania, jedyną zmianą jest to, że zamiast sumy obliczana teraz jest różnica indeksów odpowiednich liter tekstu i klucza. Gdy jest ona ujemna dodaje się do niej ilość liter alfabetu.

Wprowadzono przykładowe dane i sprawdzono działanie algorytmu. Wynik przedstawiono na poniższym screenie.

Przykład 1.

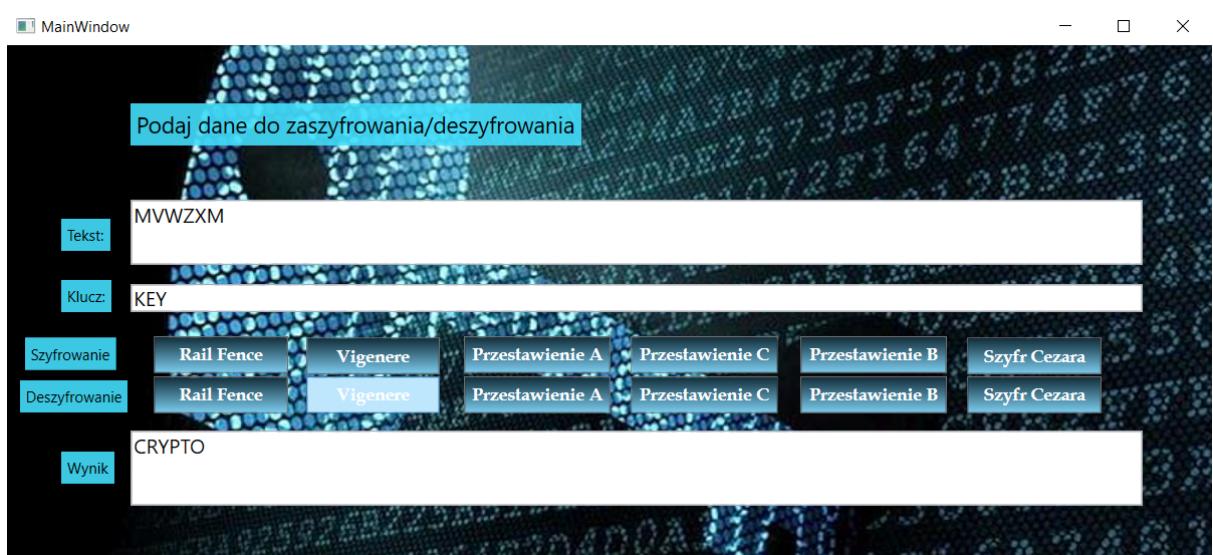
Jako tekst zaszyfrowany wprowadzono wynik z przykładu 1 z szyfrowania. Klucz podano ten sam.



Wynik jest zgodny z tekstem oryginalnym w przykładzie 1 szyfrowania.

Przykład 2.

Jako tekst zaszyfrowany wprowadzono wynik z przykładu 2 z szyfrowania. Klucz podano ten sam.



Wynik jest zgodny z tekstem oryginalnym w przykładzie 2 szyfrowania.

Walidacja danych

Podczas wczytywania danych podanych przez użytkownika zaimplementowano sprawdzenie, czy użytkownik nie zapomniał podać dane:

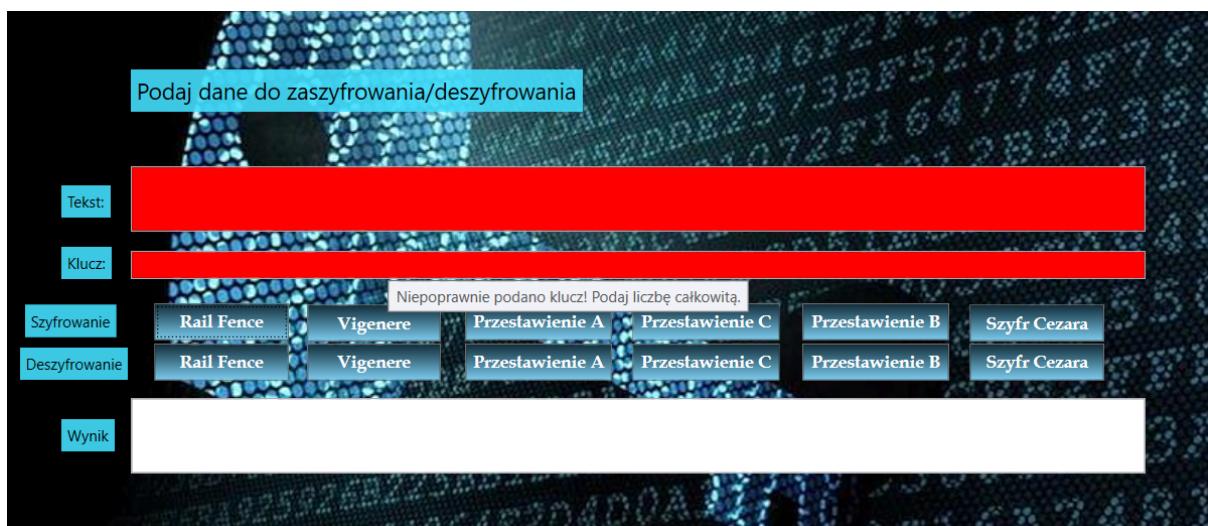
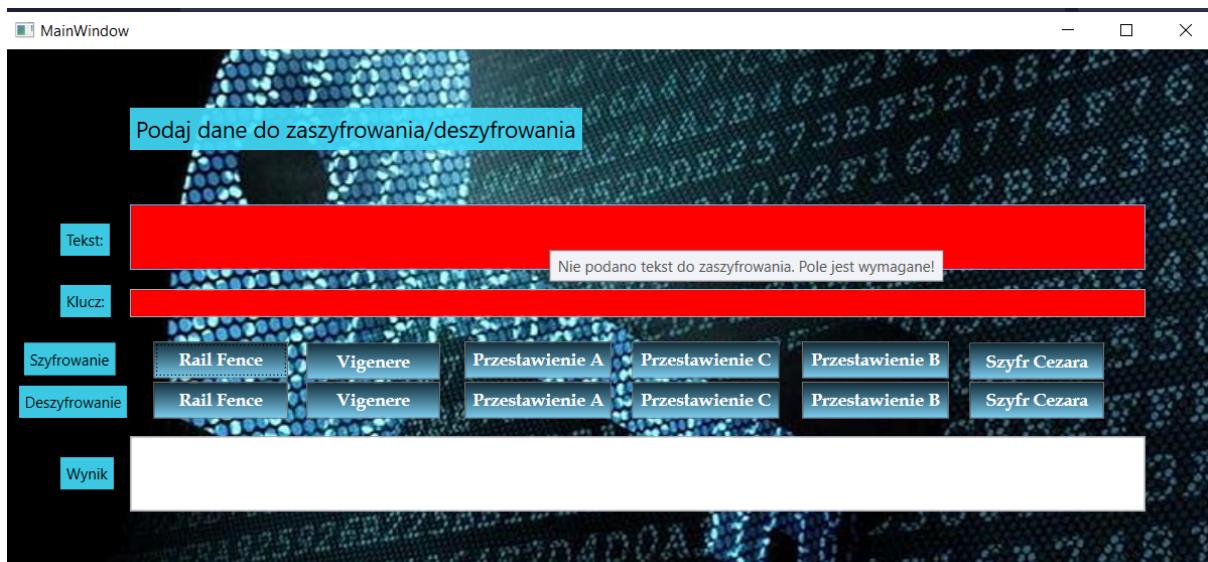
```
Odwzorowania: 12
private void WczytajDane()
{
    if (TekstPodany.Text.Length < 1)
    {
        TekstPodany.Background = Brushes.Red;
        TekstPodany.ToolTip = "Nie podano tekstu do zaszyfrowania. Pole jest wymagane!";
    }
    else if (Klucz.Text.Length < 1)
    {
        Klucz.Background = Brushes.Red;
        Klucz.ToolTip = "Niepoprawnie podano wzrost! Pole jest wymagane!";
    }
    else
    {
        TekstPodany.Background = Brushes.White;
        TekstPodany.ToolTip = "";
    }
    M = TekstPodany.Text;
    Klucz = Klucz.Text;
}
```

Przykłady

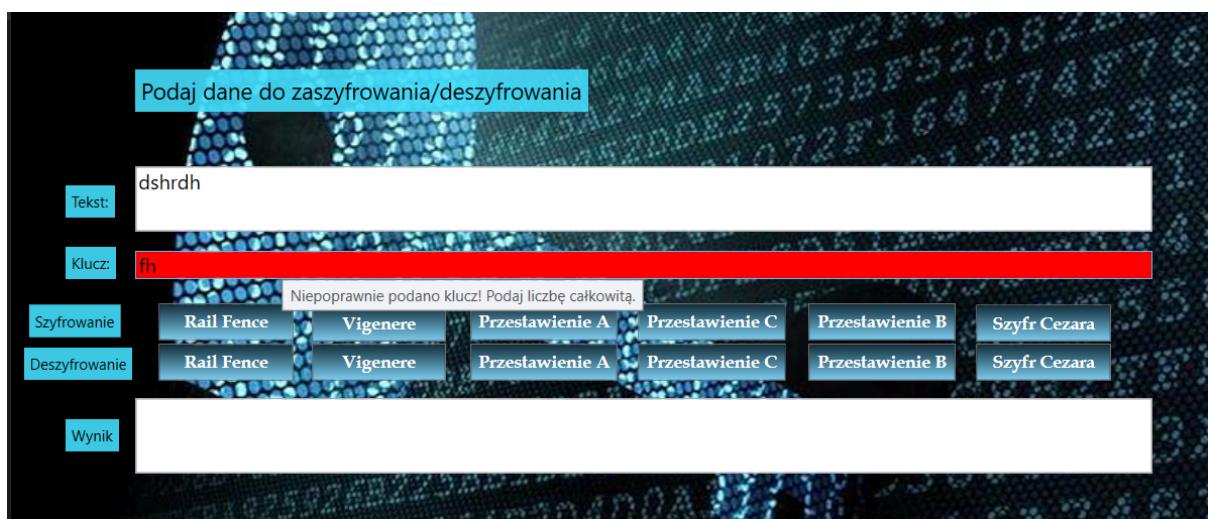
- Nie podano danych:



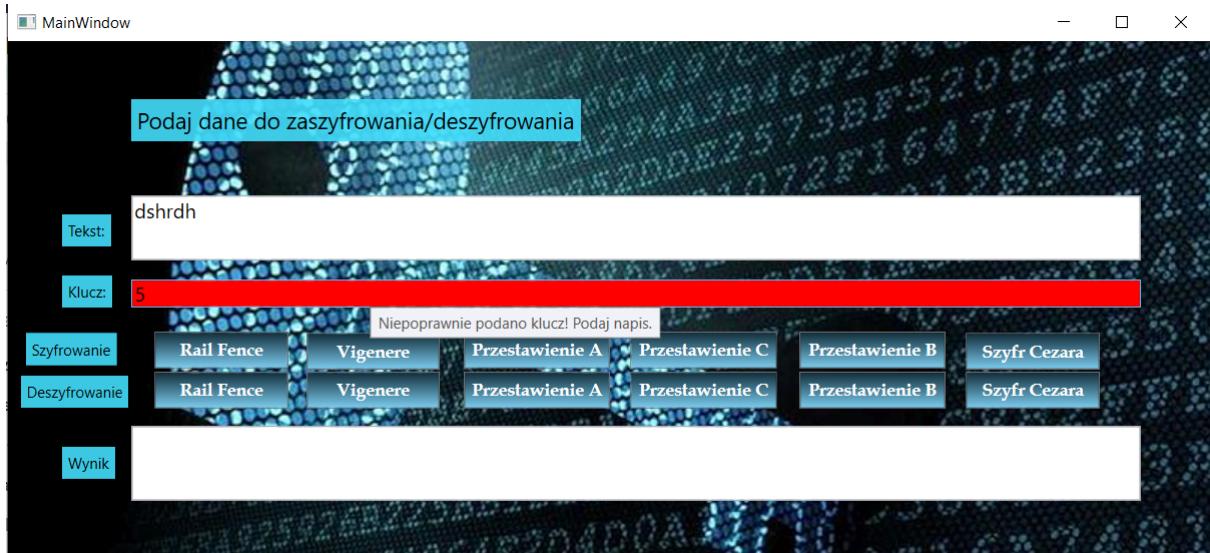
Wyświetlające się komunikaty:



- Do szyfrowania danych za pomocą Rail Fence podano napis zamiast liczby jako klucz:



- Do szyfrowania danych za pomocą szyfru Vigenere'a podano liczbę zamiast napisu jako klucz:



Wnioski

- Wszystkie algorytmy zostały wykonane zgodnie z instrukcją i otrzymano wynik którego oczekiwaliśmy.
- Żaden z zaimplementowanych szyfrów nie jest na 100% bezpiecznym. Każdy da się złamać.
- Szyfr jest nie do złamania, jeśli znajdziemy równie dobrą metodę zaszyfrowania klucza szyfrującego.
- Łamanie Szyfru Cezara jest bardzo proste, ponieważ można sprawdzić 26 wszystkich przesunięć.