**PAŃSTWOWA WYŻSZA SZKOŁA ZAWODOWA W NOWYM SĄCZU**

****

**INSTYTUT TECHNICZNY**

**Kryptografia i teoria kodów**

**Szyfrowanie tekstu za pomocą algorytmu DES w języku C#**

**(Windows Forms)**

**Projekt**

Mateusz Sromek

Kierunek: Informatyka Stosowana

3 IS(s) 2018

Nr albumu: 28280, grupa P3

Rok akademicki 2020/2021

**SPIS TREŚCI**

1. Opis projektu
   1. Algorytmy symetryczne
   2. Historia algorytmu DES
   3. Zasada działania algorytmu
   4. Zalety oraz wady szyfrowania DES
2. Ogólne przedstawienie środowisk programistycznych
   1. Visual studio
   2. C (sharp)
   3. .Net framework
3. Zastosowane biblioteki
4. Przedstawienie działania programu
   1. Szyfrowanie
   2. Deszyfrowanie
5. Kod programu
6. Literatura
7. Opis

W przeciągu ostatniej dekady nastąpił duży wzrost zapotrzebowanie na cyfrową komunikacje komputerową zarówno przez firmy jak i prywatnie. Wiele z tych informacji ma znaczącą wartość i wymaga ochrony np. banki, poufne dane oraz dane do uwierzytelniana użytkowników . W takiej sytuacji powstały algorytmy szyfrujące, które umożliwiają zmianę zwykłego tekstu w zakodowaną treść. Niewłaściwe wdrożenie i użytkowanie, szyfrowanie danych może jedynie stwarzać iluzję bezpieczeństwa. Jednak przy odpowiednich kontrolach zarządczych wystarczające specyfikacje implementacji i obowiązujące wytyczne dotyczące użytkowania, szyfrowanie danych nie tylko pomoże w ochronie komunikacji danych, ale może zapewnić ochronę niezliczonej ilości określonych danych.

Przedstawiony projekt ma za zadanie zaszyfrować wprowadzony tekst oraz podane hasło przy użyciu algorytmu DES (Data Encryption Standard) oraz możliwość odszyfrowania tekstu z tym samym hasłem co zostało zaszyfrowane. Stworzona aplikacja została w Windows Forms (.NET framework) w języku programistycznym C# w środowisku IDE Visual Studio 2019.

* 1. Algorytmy symetryczne

Szyfry symetryczne, charakteryzują się tym, iż do szyfrowania i deszyfrowania wykorzystywany jest ten sam klucz . Konieczność ochrony tego klucza powoduje, że algorytmy symetryczne są również określane mianem algorytmów z tajnym/prywatnym kluczem.

Współczesne algorytmy symetryczne, ze względu na rodzaje operacji wykonywanych w trakcie szyfrowania i deszyfrowania mogą być podzielone na dwie grupy.

W większości przypadków algorytmów bazujących na sieci Feistla (Lucifer, DES) procedury szyfrowania i deszyfrowania są dokładnie takie same a różnica polega jedynie na różnej kolejności używanych kluczy rund (klucze generowane są na podstawie klucza głównego K). W pozostałych przypadkach algorytmów symetrycznych procedury szyfrowania i deszyfrowania wykorzystują różne transformacje.

Ważnym elementem każdego algorytmu symetrycznego jest procedura rozszerzania klucza głównego. Służy do zwiększenia długości klucza kryptograficznego stosowanego w algorytmie tak, aby każda runda miała swój własny klucz rundy. Nie wnikając w szczegóły takiej procedury można stwierdzić, że w większości algorytmów symetrycznych schemat generacji kluczy rund wykorzystuje te same lub bardzo podobne transformacje jak w procedurach szyfrowania oraz deszyfrowania.

* 1. Historia algorytmu DES

W roku 1973 NIST(Narodowy Instytut Standaryzacji i Technologii) opublikował zapytanie ofertowe dotyczące krajowego systemu kryptograficznego stworzenie algorytmu szyfrującego z kluczem symetrycznym. Zleceniem się projektu podjęła firma IBM, ulepszyli projekt o nazwie „Lucifer” i została przyjęta jako DES (Data Encryption Standard). DES został opublikowany w Rejestr Federalny USA w marcu 1975 r. Jako projekt FIPS (Federalnego Standardu Przetwarzania Informacji).

Po opublikowaniu projektu został on mocno skrytykowany, ponieważ została użyta krótka długość klucza (tylko 56 bitów), co może uczynić szyfr podatnym na atak siłowy.

DES został ostatecznie opublikowany jako FIPS 46 w Rejestrze Federalnym w styczniu 1977 roku. Jednak NIST definiuje DES jako standard do użytku w niesklasyfikowanych aplikacjach. Data Encryption Standard był najczęściej używanym szyfrem blokowy z kluczem symetrycznym od czasu jego publikacji do 2001 roku , następnie NIST wydał nowy standard 3DES (Triple Data Encryption Standard), a od 2001 roku USA został zastąpiony w ramach standardu federalnego przez AES. Jest jednym z najlepiej przeanalizowanych algorytmów szyfrujących.

* 1. Zasada działania algorytmu

Pracuje na 64-bitowych blokach tekstu jawnego. Po początkowej permutacji blok wejściowy jest dzielony na lewą i prawą połowę, każda o długości 32 bitów. Następnie jest wykonywanych 16 cykli jednakowych operacji, nazywanych funkcjami f, w czasie których dane są łączone z kluczem. Po szesnastym cyklu lewa i prawa połowa są łączone z kluczem. Następnie są one łączone i końcowa permutacja (będąca odwrotnością permutacji początkowej) kończy przebieg algorytmu.

Klucz ma długość 56 bitów. (Zwykle klucz jest zapisany za pomocą 64 bitów, przy czym każdy co ósmy jest bitem parzystości, który jest pomijany). Kluczem może być dowolna liczba o długości 56 bitów, która może być zmieniona w dowolnej chwili. Kilka z tych liczb jest uważane za klucze słabe, lecz mogą one być pominięte. Całe bezpieczeństwo spoczywa na kluczu.

W algorytmie DES do szyfrowania i deszyfrowania danych wykorzystywany jest klucz o długości 56 bitów. Zazwyczaj klucz jest zapisywany w postaci 64 bitowego ciągu, przy czym każdy co ósmy jest bitem parzystości, który jest pomijany. Całe bezpieczeństwo polega na kluczu.

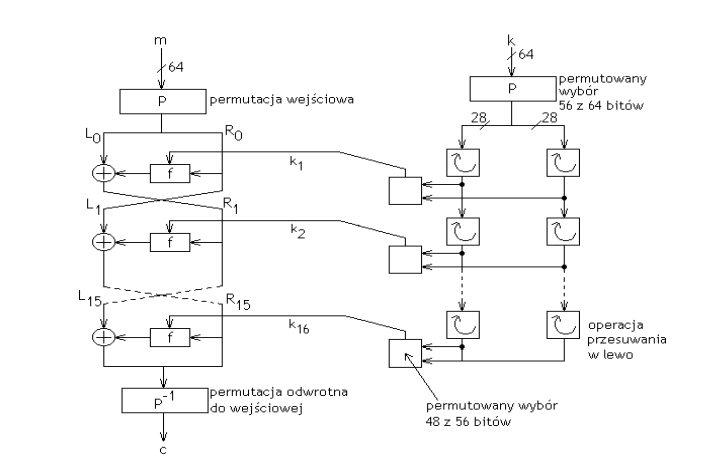
W procesie szyfrowania dane wejściowe algorytmu pracują na 64-bitowych blokach tekstu jawnego. Po początkowej permutacji blok wejściowy jest dzielony na lewą i prawą połowę, każda o długości 32 bitów. Następnie dla każdego bloku wykonywane są następujące operacje:

1. Podzielenie bloku wejściowego danych na dwie 32-bitowe części: lewą oraz prawą.
2. Wybranie 56 bitów z 64-bitowego sekretnego klucza - permutacja, a następnie podzielenie tych bitów na dwie 28-bitowe części.
3. Powtórzenie 16 razy tych samych operacji, nazywanych funkcjami Feistela, przy użyciu przekazanego klucza:
   1. Bity klucza są przestawiane w połówkach w lewo o jedną lub dwie pozycje, a następnie w Permutacjach wybieranych jest 48 z 56 bitów klucza.
   2. Prawa część danych rozszerzana jest do 48-bitów za pomocą Permutacji Rozszerzającej.
   3. Prawa połowa bloku danych jest rozszerzona do 48 bitów za pomocą operacji XOR z wybranymi wcześniej 48 bitami klucza.
   4. Zsumowane dane dzielone są na osiem 6-bitowych bloków i każdy blok podawany jest na wejście jednego z S-bloków (pierwszy 6-bitowy blok na wejście pierwszego S-bloku, drugi 6-bitowy blok na wejście drugiego S-bloku, itd.). Pierwszy i ostatni bit danych określa wiersz, a pozostałe cztery bity wskazują kolumnę w tabeli S-Bloku. Po wyznaczeniu właściwej pozycji w tabeli, odczytuje się wynikową wartość i zamienia liczbę na zapis dwójkowy. Wynikiem działania każdego S-Bloku są 4 bity wyjściowe. Wszystkie wyjścia z S-Bloków tworzą 32-bitową tablicę. Każdy S-Blok ma inną strukturę.
   5. Dane wyjściowe z S-Bloków są łączone i poddawane permutacji w P-Blokach.
   6. Bity tak przekształconej prawej połowy danych dodawane są do bitów lewej połowy danych.
   7. Zmieniona lewa połowa danych staje się nową prawą połową, natomiast poprzednia prawa połowa staje się nową lewą połową.
4. Po wykonaniu wszystkich 16 powtórzeń ciąg wyjściowy funkcji feistela jest dalej łączony z lewą połową za pomocą poelementowej sumy XOR.
5. Wynikiem tych operacji jest nowa prawa połowa bloku; stara prawa połowa staje się nową lewą.

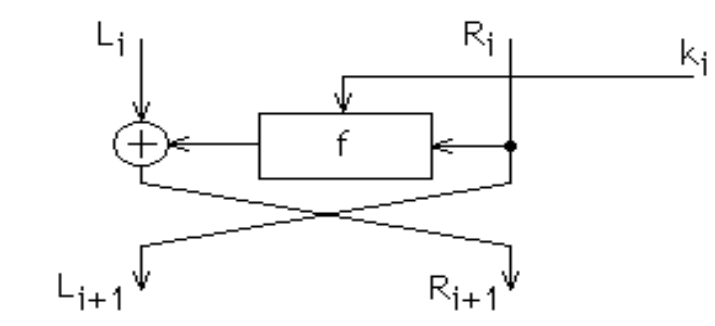
**Deszyfrowanie** w DES polega na zastosowaniu tych samych operacji w odwrotnej kolejności. Jedyną różnicą jest to, że podklucze, które teraz należy wybierać od końca.

(Funkcja feistela składa się z s-bloków o tajnej strukturze)

Schemat blokowy algorytmu:



Pojedyncza iteracja (w uproszczeniu)



* 1. Zalety oraz wady szyfrowania DES

**Zalety**

* 1. każdy bit szyfru jest oparty na wielu bitach klucza.
  2. każdy bit szyfru zależy od wielu bitów zwykłego tekstu i niewielka zmiana zwykłego tekstu drastycznie zmieni tekst szyfru.
  3. Jest to bardzo wygodne dla wymagań oprogramowania i sprzętu, jak szyfrowanie i odszyfrowywanie ma ten sam algorytm. Tylko funkcja szyfrowania musi zostać odwrócona, a klucze okrągłe należy podjąć w odwrotnej kolejności.

**Wady:**

1. Klucz wybrany na rundach jest problemem. Podczas dzielenia kluczy na dwie połówki i zamiany mogą rzucać ten sam wynik, jeśli mają ciągłe 1 i 0. Rzeczy mogą skończyć się przy użyciu tego samego klucza w ciągu 16 cykli.
2. niektóre systemy haseł zabezpieczone DES były ograniczone do 8 znaków i po cichu obcinały hasła (pasują tylko do pierwszych 8 znaków).
3. Kiedy DES został opracowany, nie można było złamać tego szyfru przez brutalny atak siłowy. Obecnie dzięki współczesnym komputerów, łamanie DES stało się stosunkowo łatwe.
4. Ogólne przedstawienie środowisk programistycznych
   1. Visual studio

Zintegrowane środowisko programistyczne (IDE), umożliwia tworzenie samodzielnych aplikacji, aplikacji sieciowych, usług sieciowych oraz serwisów internetowych, założycielem jest firma Microsoft z licencją: komercyjna / freeware. Jest używane do tworzenia oprogramowania konsolowego oraz z graficznym interfejsem użytkownika, w tym aplikacji Windows Forms, WPF, Web Sites, Web Applications i inne. Aplikacje mogą być pisane na platformy: Microsoft Windows, Windows Phone, Windows CE, .NET Framework, Microsoft Silverlight, Linux, MacOS oraz konsole XBOX.

* 1. C (sharp)

Nowoczesny język programowania stworzony przez firmę Microsoft. Został stworzony przez Andersa Hejlsberga, który również był zaangażowany w tworzenie takich języków jak Delphi czy Turbo Pascal. Powstał na bazie języków C++ oraz C. Obecnie znajduje się w czołówce najpopularniejszych języków programowania na świecie. Używany jest głównie do tworzenia oprogramowania z wykorzystaniem platformy .NET.

Największą zaletą i zarazem wadą C# jest względna zamkniętość. Prawie wszystkie narzędzia i rozwiązania pochodzą od Microsoft. Na plus działa wsteczna kompatybilność i wszystko trzymane jest w bardzo podobnym schemacie. Niestety, wszystkie nowości i plusy płynące z Open Source trafiają z opóźnieniem.

* 1. .Net framework

To platforma programistyczna opracowana przez Microsoft, wydana w roku 2002. Obejmuje Common Language Runtime (CLR), czyli środowisko uruchomieniowe na których działają napisane programy, a także biblioteki klas, które implementują typowe funkcjonalności, wykorzystywane w aplikacjach .NET. Wspomniane biblioteki to zbiór funkcjonalności i metod, które ułatwiają pracę programistów. Dzięki którym możesz użyć gotowych już implementacji. Zadaniem platformy .NET jest między innymi zarządzanie kodem aplikacji, pamięcią i zabezpieczeniami. Programy na platformę .NET mogą być pisane w różnych językach, takich jak C#, F#, Delphi 8, Visual Basic .NET, COBOL,

1. Zastosowane biblioteki
2. using System.Security.Cryptography

Zapewnia usługi kryptograficzne, w tym bezpieczne kodowanie i dekodowanie danych, a także wiele innych operacji, takich jak tworzenie skrótów, generowanie losowe i uwierzytelnianie komunikatów.

Obszar nazw System.Security.Cryptography zawiera obsługę najbardziej typowych algorytmów symetrycznych (m.in. DES, 3DES, RC2, Rijndael), asymetrycznych (RSA, DSA) i haszyszu (MD5, SHA-1, SHA-256, SHA- 384, SHA-512) algorytmów krypty. Zawiera również pomocne klasy do szyfrowania i odszyfrowywania strumieni.

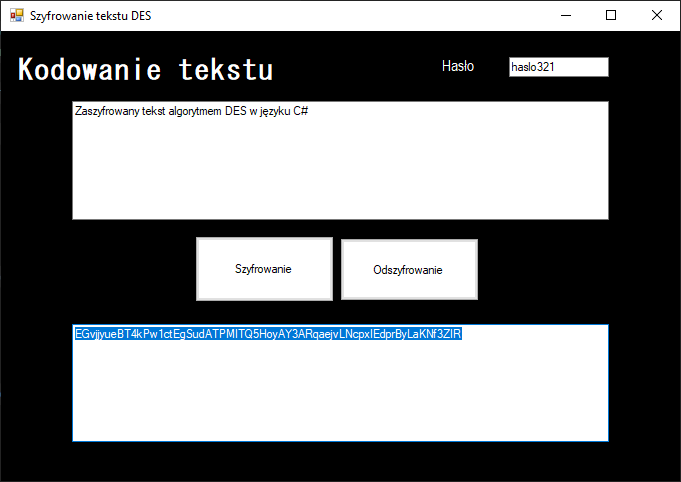
1. using System.IO

Zawiera typy, które umożliwiają odczytywanie i zapisywanie plików oraz strumieni danych oraz typy zapewniające podstawową obsługę plików i katalogów.

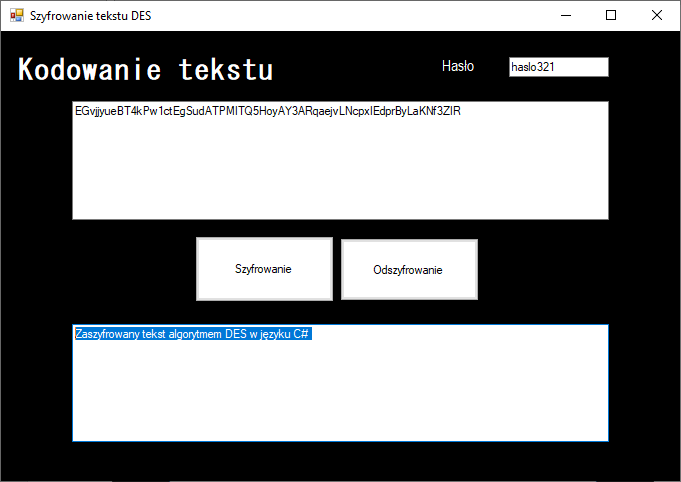
1. using System.Text
2. using System.Windows.Forms
3. Przedstawienie działania programu
   1. Szyfrowanie

Wprowadzamy tekst który chcemy zaszyfrować, następnie wpisujemy hasło dzięki któremu możemy odkodować zaszyfrowany tekst.

Klikamy przycisk „Szyfrowanie” i zostaje nam wyświetlony zaszyfrowany tekst.



* 1. Deszyfrowanie  
     Wprowadzamy zaszyfrowany tekst wpisujemy poprawne hasło i naciskamy ‘Odszyfrowanie’ tekst zostaje odszyfrowany.



1. Kod programu

using System;

using System.IO;

using System.Security.Cryptography;

using System.Text;

using System.Windows.Forms;

namespace WindowsFormsApp1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void label1\_Click(object sender, EventArgs e)

{

}

private void Form1\_Load(object sender, EventArgs e)

{

}

private void button1\_Click(object sender, EventArgs e)

{

}

private void button2\_Click(object sender, EventArgs e)

{

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

}

private void textBox2\_TextChanged(object sender, EventArgs e)

{

}

private void btn\_odsz(object sender, EventArgs e)

{

try

{

if (textBox3.Text.Length < 8)

{

MessageBox.Show("Klucz musi się składać z 8 znaków");

textBox3.Focus(); //przeniesienie kursora na klucz

return;

}

string password = textBox3.Text;

string cipheredText = textBox1.Text;

byte[] cipheredBytes = Convert.FromBase64String(cipheredText); //Konwertuje tekst z textBox1, gdzie koduje dane binarne jako cyfry Base64, do równoważnej 8-bitowej nieoznaczonej liczby całkowitej

DESCryptoServiceProvider Alg = new DESCryptoServiceProvider();

// 8 bitowy tajny klucz DES

//GetBytes - koduje znaki w bajty

Alg.Key = ASCIIEncoding.ASCII.GetBytes(password); //ASCII - kodowanie znaków Unicode aby nie stracic danych które występują poza zakresem ASCII

Alg.IV = ASCIIEncoding.ASCII.GetBytes(password); //wektor inicjalizacji ( IV )

MemoryStream strDeciphered = new MemoryStream(); // Przechowywanie odszyfrowanych danych

CryptoStream strCrypto = new CryptoStream(strDeciphered, Alg.CreateDecryptor(), CryptoStreamMode.Write); //łączenie strumieni danych z transformacjami kryptograficznymi

strCrypto.Write(cipheredBytes, 0, cipheredBytes.Length);

strCrypto.Close();

textBox2.Text = Encoding.UTF8.GetString(strDeciphered.ToArray()); //dekodowanie sekwencji bajtów na ciąg i zapisanie zawartości strumienia w tablicy bajtów

strDeciphered.Close();

textBox2.Focus();

textBox2.Focus();

}

catch (Exception ex)

{

MessageBox.Show("Niepoprawny klucz zabezpieczający !" + ex.Message);

}

}

private void btn\_szyf\_Click(object sender, EventArgs e)

{

if (textBox3.Text.Length < 8)

{

MessageBox.Show("Klucz musi się składać z 8 znaków");

textBox3.Focus(); //przeniesienie kursora na klucz

return;

}

try

{

string password = textBox3.Text;

string plainText = textBox1.Text;

byte[] plainBytes = Encoding.UTF8.GetBytes(plainText);

DESCryptoServiceProvider Alg = new DESCryptoServiceProvider();

// 8 bitowy tajny klucz DES

Alg.Key = ASCIIEncoding.ASCII.GetBytes(password);

Alg.IV = ASCIIEncoding.ASCII.GetBytes(password);

MemoryStream strCiphered = new MemoryStream(); //To Store Encrypted Data

CryptoStream strCrypto = new CryptoStream(strCiphered,

Alg.CreateEncryptor(), CryptoStreamMode.Write);

strCrypto.Write(plainBytes, 0, plainBytes.Length);

strCrypto.Close();

textBox2.Text = Convert.ToBase64String(strCiphered.ToArray());

strCiphered.Close();

textBox2.Focus();

}

catch (Exception ex)

{

MessageBox.Show("Niepoprawny klucz zabezpieczający !" + ex.Message);

}

}

private void key\_txt(object sender, EventArgs e)

{

}

private void textBox3\_TextChanged(object sender, EventArgs e)

{

}

}

}

1. Literatura

[opis\_DES.pdf (polsl.pl)](http://sun.aei.polsl.pl/~kfrancik/bsk/dokumenty/opis_DES.pdf)

[DES Szyfr Symetryczny | Kryptografia | Crypto-IT (crypto-it.net)](http://www.crypto-it.net/pl/symetryczne/des.html)

[10-Minutowy Przewodnik Po C# i .NET - Modest Programmer](https://www.modestprogrammer.pl/10-minutowy-przewodnik-po-csharp-i-dotnet)

<https://nvlpubs.nist.gov/nistpubs/Legacy/FIPS/fipspub74.pdf>

<https://csrc.nist.gov/csrc/media/publications/fips/46/3/archive/1999-10-25/documents/fips46-3.pdf>

<https://www.irjet.net/archives/V4/i3/IRJET-V4I3489.pdf>

http://ii.uwb.edu.pl/rybnik/BDiSI/BDiSI%20W5.pdf

https://academic.csuohio.edu/yuc/security/Chapter\_06\_Data\_Encription\_Standard.pdf