

MINISTERUL EDUCAȚIEI, CULTURII ȘI CERCETĂRII
UNIVERSITATEA TEHNICĂ a MOLDOVEI
FACULTATEA CALCULATOARE, INFORMATICĂ ȘI MICROELECTRONICĂ

Raport

Lucrare de laborator 1

la disciplina Securitatea Informațională

Tema: Algoritmul DES (Data Encryption Standard)

A efectuat: st. gr. TI-151 F/R

Constantinescu Nadejda

A verificat: lect. Superior

Poștaru Andrei

Chișinău 2020

Sarcina

Implementarea algoritmului Data Encryption Standard.

Indicații teoretice

Standardul de Criptare a Datelor este un cifru (o metoda de criptare a informației), selectat ca standard federal de procesare a informațiilor in Statele Unite in 1976, si care s-a bucurat ulterior de o larga utilizare pe plan internațional. Algoritmul a fost controversat inițial, având elemente secrete, lungimea cheii scurta si fiind bănuিত ca ascunde de fapt o portița pentru NSA. DES a fost analizat intens de către profesioniști in domeniu si a motivat înțelegerea cifrurilor bloc si criptanaliza lor (figura 1).

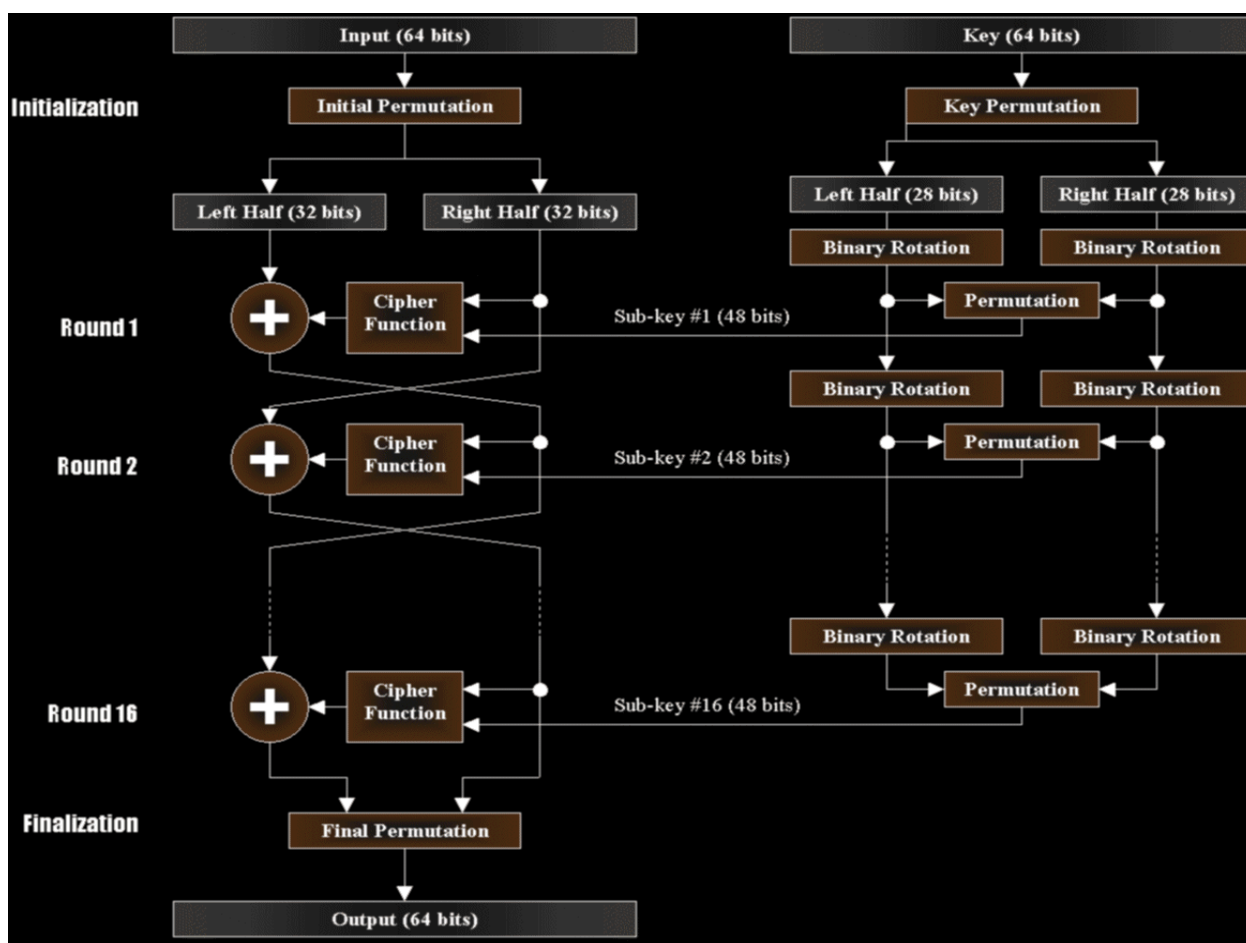


Figura 1- Descrierea generală a algoritmului DES

DES este astăzi considerat nesigur pentru multe aplicații. Acest lucru se datorează în principiu cheii de 56 de biți, considerată prea scurtă; cheile DES au fost sparte în mai puțin de 24 de ore. De asemenea, există unele rezultate analitice care demonstrează slăbiciunile teoretice ale cifrului, deși nu este fezabilă aplicarea

lor. Se crede ca algoritmul este practic sigur in forma Triplu DES, deși exista atacuri teoretice si asupra acestuia. In ultimii ani, cifrul a fost înlocuit de Advanced Encryption Standard (AES).

Mersul lucrării

Pentru a începe elaborarea algoritmului DES trebuie să enumerăm câteva aspecte principale ale acestuia. DES este cifrul bloc arhetip - un algoritm care ia un sir de lungime fixa de biți de text normal si îl transforma print-o serie de operații complexe într-un sir de biți criptați de aceeași lungime. În cazul DES, mărimea blocului este de 64 biți. DES folosește de asemenea si o cheie pentru particularizarea transformării, astfel încât numai cei care cunosc cheia folosita sa poată efectua decriptarea. Cheia este formata din 64 de biți; totuși, numai 56 dintre ei sunt folosiți propriu-zis de algoritm. Opt biți sunt utilizați ca biți de paritate si nu sunt necesari după acest test. Deci cheia efectiva are doar 56 de biți, si așa este citata de obicei.

```
using System;
using System.Collections.Generic;

namespace SI_Lab1
{
    internal class Program
    {
        private static void Main(string[] args)
        {
            Console.WriteLine("Enter the input as a 16 character hexadecimal value:");
            var message = Console.ReadLine();

            var devidedMessages = DesImplementation.DivideMessageHexToBinaryOf64Bits(message);

            var msg = new ConvertMessage();
            var messagesInBits = msg.ConvertMessageToBinary(devidedMessages, message);

            Console.WriteLine("Enter the key as a 16 character hexadecimal value:");
            var hexKey = Console.ReadLine();

            var key = new ConvertKey();
            var keyBits = key.ConvertKeyBytes(hexKey);

            Console.WriteLine("\n+++ ENCRYPTION +++");
            var inputBitsForEncryption = new int[64];
            for (var i = 0; i < devidedMessages.Count; i++)
            {
                var temp = messagesInBits[i].ToCharArray();

                for (var j = 0; j < temp.Length; j++)
                {
                    inputBitsForEncryption[j] = (int) char.GetNumericValue(temp[j]);
                }

                DesImplementation.permute(inputBitsForEncryption, keyBits, false);
            }
        }
    }
}
```

```

        Console.WriteLine(DesImplementation.cryptedText);
        var cryptedDone =
DesImplementation.DivideMessageHexToBinaryOf64Bits(DesImplementation.cryptedText);
        IList<string> cryptedDoneInBits = msg.ConvertMessageToBinary(cryptedDone,
DesImplementation.cryptedText);

        Console.WriteLine();
        Console.WriteLine("\n+++ DECRYPTION +++");
        var inputBitsForDecryption = new int[64];
        for (var i = 0; i < cryptedDoneInBits.Count; i++)
        {
            var temp1 = cryptedDoneInBits[i].ToCharArray();
            for (var j = 0; j < temp1.Length; j++)
            {
                inputBitsForDecryption[j] = (int) char.GetNumericValue(temp1[j]);
            }
            var binaryTextTemp = DesImplementation.permute(inputBitsForDecryption, keyBits,
true);

            string stringTextTemp = null;
            foreach (var item in binaryTextTemp)
            {
                stringTextTemp += item;
            }

            var myBytesTemp = DesImplementation.GetBytesFromBinaryString(stringTextTemp);
            var str = DesImplementation.ByteArrayToString(myBytesTemp);
            Console.WriteLine(str);
        }
        Console.ReadKey();
    }
}

```

Formatul mesajului de intrare este în Hexazecimal, pentru a transforma mesajul din hexazecimal în binar se aplică următoarea secvență de cod:

```
public static IList<string> DivideMessageHexToBinaryOf64Bits(string str)
{
    var list = new List<string>();
    for (int i = 0; i < str.Length; i += 16)
    {if (str.Length > (list.Count + 1) * 16)
        list.Add(str.Substring(i, 16));
        else
            list.Add(str.Substring(i, str.Length - list.Count * 16));
    }return list;}
public List<string> ConvertMessageToBinary(IList<string> message, string originalmessage)
{
    var inputBits = new int[64];
    var messagesInBits = new List<string>();
    for (var messagCount = 0; messagCount < message.Count; messagCount++)
    {
        string binaryMessage = null;
        for (var i = 0; i < 16; i++)
        {string hexMessage = null;
        hexMessage = originalmessage;
        var temp = int.Parse(hexMessage.ToCharArray().ElementAt(i).ToString(), NumberStyles.HexNumber);
        var s2 = Convert.ToString(temp, 2);
        while (s2.Length < 4)
        {s2 = "0" + s2; }
        for (var j = 0; j < 4; j++)
        {inputBits[(4*i) + j] = int.Parse(s2.ElementAt(j) + "");}
        binaryMessage = null;
        foreach (var digit in inputBits)
        {binaryMessage += digit; }
        }messagesInBits.Add(binaryMessage);} return messagesInBits;}
```

Ca obiectiv în această lucrare am avut de implementat criptarea și decriptarea. Astfel la aceste 2 operații se efectuează aceeași pași, dar pentru decriptare pașii se efectuează în ordine opusă operații de criptare. Având mesajul nostru în biți, trebuie de lansat algoritmul de criptare.

```
var inputBitsForEncryption = new int[64];
for (var i = 0; i < dividedMessages.Count; i++)
{var temp = messagesInBits[i].ToCharArray();
for (var j = 0; j < temp.Length; j++)
{inputBitsForEncryption[j] = (int) char.GetNumericValue(temp[j]);
}DesImplementation.permute(inputBitsForEncryption, keyBits, false); }
```

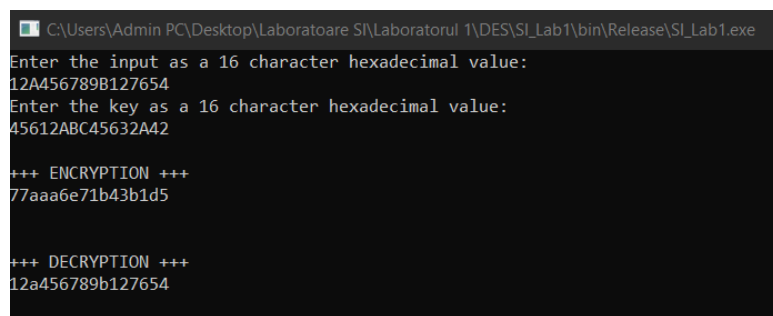
Decriptarea are loc în modul următor:

```
for (var i = 0; i < cryptedDoneInBits.Count; i++)
{var temp1 = cryptedDoneInBits[i].ToCharArray();
for (var j = 0; j < temp1.Length; j++)
{inputBitsForDecryption[j] = (int) char.GetNumericValue(temp1[j]);}
var binaryTextTemp = DesImplementation.permute(inputBitsForDecryption, keyBits, true);
string stringTextTemp = null;
foreach (var item in binaryTextTemp)
{stringTextTemp += item;}
var myBytesTemp = DesImplementation.GetBytesFromBinaryString(stringTextTemp);
var str = DesImplementation.ByteArrayToString(myBytesTemp);
Console.WriteLine(str);
}
```

Un rol primordial în DES îl are cheia care este transmisă algoritmului. Cu ajutorul ei se realizează criptarea și decriptarea mesajelor. Cheia trebuie să fie unică pentru ambele operații. Aceasta necesită la rândul ei să fie transformată în biți.

```
public int[] ConvertKeyBytes(string key)
{
    var tempKey = key.ToCharArray();
    var keyBits = new int[64];
    for (var i = 0; i < 16; i++)
    {
        var temp = int.Parse(tempKey.ElementAt(i).ToString(), NumberStyles.HexNumber);
        var s2 = Convert.ToString(temp, 2);
        while (s2.Length < 4)
        {
            s2 = "0" + s2;
        }
        for (var j = 0; j < 4; j++)
        {
            keyBits[(4*i) + j] = int.Parse(s2.ElementAt(j) + "");
        }
    }
    return keyBits;
}
```

La executarea algoritmului DES s-au obținut următoarele rezultate reprezentate în imaginea următoare.



```
C:\Users\Admin PC\Desktop\Laboratoare SI\Laboratorul 1\DES\SI_Lab1\bin\Release\SI_Lab1.exe
Enter the input as a 16 character hexadecimal value:
12A456789B127654
Enter the key as a 16 character hexadecimal value:
45612ABC45632A42

+++ ENCRYPTION +++
77aaa6e71b43b1d5

+++ DECRYPTION +++
12a456789b127654
```

Figura 2- Rezultatele execuției programului

Concluzie

Am efectuat aceasta lucrare de laborator am insusit algoritmul DES, care depinde de utilizarea cheii care are 56 de biti, din aceasta cauza se utilizeaza o singura chei. Un dezavanta in acest algortm este cheia de 56 biti care poate fi sparta, la fel daca are loc pierderea cheiei informatia criptata.

Bibliografie

1 Algoritmul DES. (n.d.). Resrsă electronică:

http://www.copiatac.3x.ro/Proiecte_AC/MoldovanRadu/index.htm

2 Copyright © 2010 - 2016 Securitatea-Informatiilor.ro. (n.d.). Algoritmul de criptografie DES. Resrsă electronică:

<http://www.securitatea-informatiilor.ro/solutii-de-securitate-it/algoritmul-de-criptografie-des/>

3 Grabbe, J. O. (n.d.). The DES Algorithm Illustrated. Resrsă electronică:

<http://page.math.tu-berlin.de/~kant/teaching/hess/krypto-ws2006/des.html>