TUGAS TAMBAHAN
DES CIPHER

Disusun sebagai salah satu tugas
mata kuliah Kriptografi

Patricia Joanne
140810160065

Dikumpulkan tanggal
22 Desember 2018

PROGRAM STUDI S-1 TEKNIK INFORMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS PADJADJARAN
2018

PENGERTIAN

DES merupakan salah satu algoritma kriptografi cipher block dengan ukuran blok 64 bit dan ukuran kuncinya 56 bit. Algoritma DES dibuat di IBM dan merupakan modifikasi dari algoritma terdahulu yang bernama Lucifer. Lucifer merupakan algoritma cipher block yang beroperasi pada blok masukan 64 bit dan kuncinya berukuran 28 bit. Pengurangan jumlah bit kunci pada DES dilakukan dengan alasan agar mekanisme algoritma ini bisa diimplementasikan dalam satu chip.

SEJARAH

DES bermula dari hasil riset Tuchman Meyer yang diajukan sebagai kandidat sandi standard Nasional yang diusulkan oleh NBS (National Bureau Standard). Konon katanya, algoritma yang dikembangkan oleh Tuchman Meyer ini merupakan algoritma terbaik dari semua kandidat Sandi Standard Nasional.

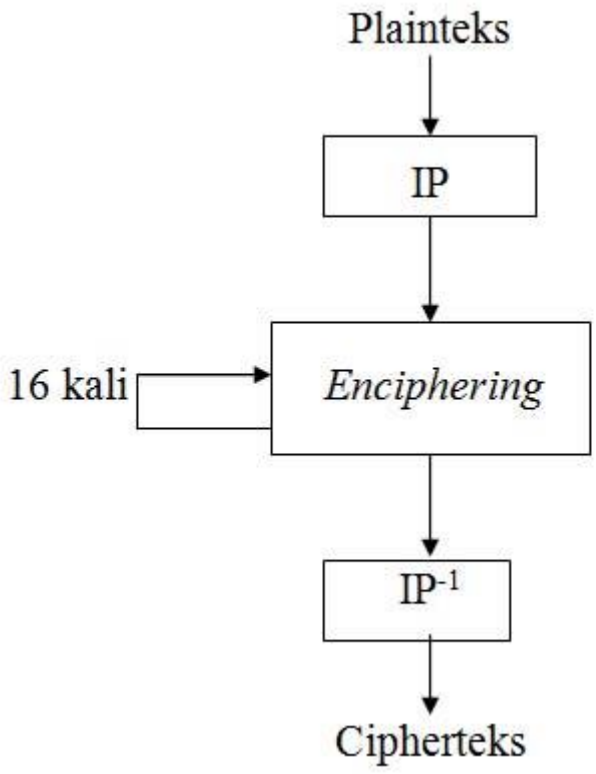
Pada mulanya, DES memiliki panjang kunci sandi 128 bit. Namun selama proses pengadopsian, NBS melibatkan NSA (National Security Agency) dan algoritma sandi ini mengalami pengurangan ukuran kunci sandi dari 128 bit menjadi 56 bit saja. Sebagian orang mungkin mengira bahwa pengurangan panjang kunci sandi ini merupakan usulan NSA untuk melemahkan algoritma Tuchman Meyer karena motif politik tertentu entah itu untuk mempermudah penyadapan atau untuk melemahkan pengamanan informasi lawan politik. Mungkin NSA menginginkan algoritma Tuchman Meyer ini “cukup aman” untuk digunakan warga sipil, tetapi mudah dipecahkan oleh organisasi besar semisal NSA dengan peralatan canggihnya. Bila dibandingkan dengan performa komputer personal pada saat itu, algoritma sandi dengan panjang kunci 56 bit dapat dikatakan cukup aman bila digunakan oleh orang-orang “biasa”, tapi dapat dengan mudah dipecahkan dengan peralatan canggih dan tentunya kepemilikan alat canggih ini hanya dapat dijangkau oleh organisasi elit seperti NSA. Dengan dukungan dana yang melimpah, pembuatan alat brute-force DES bukanlah hal yang mustahil pada saat itu.

Kini algoritma DES sudah usang dan keamanannya pun sudah tidak dapat dipertanggungjawabkan lagi. DES telah secara resmi digantikan fungsinya oleh AES (Advanced Encryption Standard) dengan panjang kunci sandi 128, 192 dan 256 bit. Meskipun begitu, tidak ada salahnya jika kita mempelajari algoritma ini untuk tujuan hobi atau pendidikan.

IMPLEMENTASI

1. DES sudah diimplementasikan dalam bentuk chip. Setiap detik chip ini dapat mengenkripsikan 16,8 juta blok atau 1 gigabit per detik.
2. DES dapat melakukan enkripsi 32.000 blok per detik pada komputer mainframe IBM 3090.
3. DES digunakan pada enkripsi PIN (Personal Identification Numbers) mesin ATM (Automatic Teller Machine) dan transaksi perbankan lewat internet.
4. Organisasi pemerintahan di Amerika seperti Department of Energy, Justice Department, dan Federal Reserve System menggunakan DES untuk melindungi penyebaran data mereka.

SKEMA GLOBAL DES

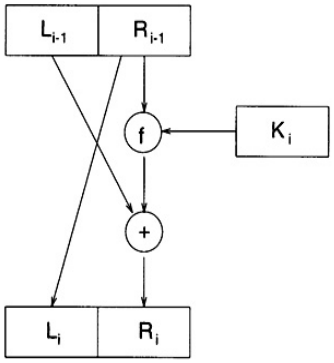


1. Blok plain text dipermutasi dengan matriks permutasi awal yang disebut juga dengan initial permutation atau IP.
2. Hasil permutasi awal kemudian di-enciphering sebanyak 16 kali (16 putaran). Setiap putaran menggunakan kunci internal yang berbeda.

- Di dalam proses enciphering, blok plainteks terbagi menjadi dua bagian, kiri (L) dan kanan (R), yang masing-masing panjangnya 32 bit.
- Pada setiap putaran i, blok R merupakan masukan untuk fungsi transformasi yang disebut f. Pada fungsi f, blok R dikombinasikan dengan kunci internal K_i. Keluaran dari fungsi f di-XOR-kan dengan blok L untuk mendapatkan blok R yang baru sedangkan blok L yang baru langsung diambil dari blok R sebelumnya. Ini adalah satu putaran DES.
- Secara matematis, satu putaran DES dinyatakan sebagai:

$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$$

Atau dapat juga ditunjukkan dalam diagram sebagai berikut:



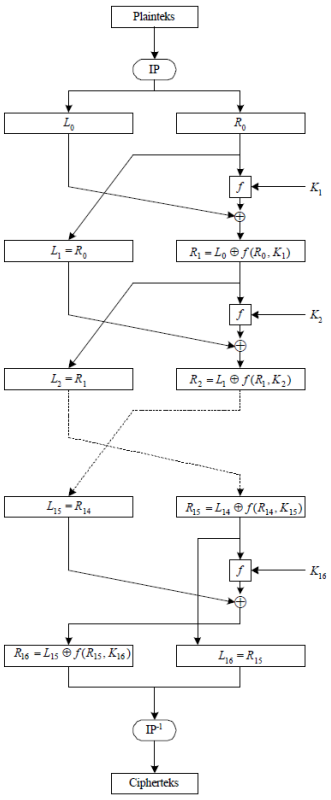
3. Hasil enciphering kemudian dipermutasi dengan matriks permutasi balikan (invers initial permutation atau IP-1) menjadi blok cipher text.

LANGKAH MENGHITUNG

DES merupakan algoritma enkripsi blok simetris karena pemrosesan data baik enkripsi maupun dekripsi diimplementasikan per blok (dalam hal ini 8 byte). Algoritma yang digunakan untuk enkripsi relatif atau bahkan sama persis dengan algoritma yang digunakan dalam proses dekripsi.

1. Enkripsi
    1. Ubah plain text dan key ke dalam bentuk biner.
    2. Lakukan Initial Permutation pada plain text sehingga menghasilkan IP(x).
    3. Pecah dua bagian bit kiri dan kanan IP(x) menjadi L_0 dan R_0.

4. Generate key dengan permutasi kompresi PC-1 (buang 1 bit blok kunci dari 64 bit menjadi 56 bit) sehingga menjadi CD(k).
5. Pecah dua bagian bit kiri dan kanan CD(k) menjadi C_0 dan D_0.
6. Lakukan left shift C_0 dan D_0 sebanyak 1 atau 2 kali berdasarkan putaran ke-inya.
7. Setiap hasil putaran digabungkan kembali menjadi C_iD_i dan diinput ke dalam tabel permutasi kompresi 2 (PC-2) sehingga C_iD_i 56 bit menjadi C_iD_i 48 bit.
8. Ekspansi data R_{i-1} 32 bit menjadi R_i 48 bit sebanyak 16 kali putaran dengan nilai perputaran $1 \leq i \leq 16$ menggunakan Tabel Ekspansi (E).
9. Hasil $E(R_{i-1})$ kemudian di XOR dengan K_i dan menghasilkan Vektor Matriks A_i.
10. Substitusikan Vektor A_i ke delapan buah S-Box (Substitution Box) dan menghasilkan output vektor B_i 32 bit menjadi nilai vektor B_i.
11. Mutasikan bit vektor B_i menggunakan tabel P-Box kemudian dikelompokkan menjadi 4 blok dimana tiap-tiap blok memiliki 32 bit data.
12. Hasil $P(B_i)$ kemudian di XOR dengan L_{i-1} untuk mendapatkan nilai R_i. Sedangkan nilai Li sendiri diperoleh dari Nilai Ri-1 untuk nilai $1 \leq i \leq 16$.
13. Gabungkan R_{16} dengan L_{16} kemudian dipermutasikan untuk terakhir kali dengan tabel Invers Initial Permutasi (IP^{-1}) menghasilkan output cipher text.



2. Dekripsi

   Proses dekripsi terhadap cipherteks merupakan kebalikan dari proses enkripsi. Jika pada proses enkripsi urutan kunci internal yang digunakan adalah K_1, K_2, …, K_{16}, maka pada proses dekripsi urutan kunci yang digunakanadalah K_{16}, K_{15}, …, K_1.

   Cara untuk mendapatkan plain text kembali yaitu:

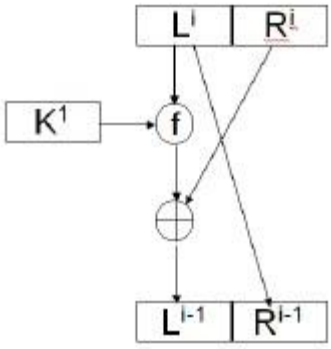
$$x = IP^{-1} (RD^0 LD^0)$$

   Secara matematis, satu putarannya dinyatakan sebagai:

$$L^{i-1} = R^i \oplus f(L^i, K^i)$$

$$R^{i-1} = L^i$$

   Atau dapat juga ditunjukkan dalam diagram sebagai berikut:



CONTOH SOAL

Diberikan:

Plain text : 0123456789ABCDEF (Hexadesimal)

Key : 133457799BBCDFF1 (Hexadesimal)

Ditanya: Cipher text?

Jawab:

- Ubah plain text ke dalam bentuk biner:

  0000 0001 0010 0011

  0100 0101 0110 0111

  1000 1001 1010 1011

  1100 1101 1110 1111

- Lakukan Initial Permutation pada plain text sehingga menghasilkan IP(x):

  1100 1100

  0000 0010

  1100 1100

  1111 1111

  1111 0000

  1010 1010

  1111 0000

  1010 1010

- Didapatkan L_0 dan R_0 dimana:

  L_0 = 1100 1100 0000 0010 1100 1100 1111 1111

  R_0 = 1111 0000 1010 1010 1111 0000 1010 1010

- Ubah key ke dalam bentuk biner:

  0001 0011 0011 0100

  0101 0111 0111 1001

  1001 1011 1011 1100

  1101 1111 1111 0001

- Generate key ke dalam PC-1

1	1	1	1	0	0	0
0	1	1	0	0	1	1
0	0	1	0	1	0	1
0	1	0	1	1	1	1
0	1	0	1	0	1	0
1	0	1	1	0	0	1
1	0	0	1	1	1	1
0	0	0	1	1	1	1

- Didapatkan C_0 dan D_0 dimana C_0 adalah 28 bit pertama dan D_0 adalah 28 bit terakhir.
- Iterasi C_0 dan D_0 sebanyak 16 kali:

  $C_i = LS_i(C_{i-1})$

  $D_i = LS_i(D_{i-1})$

  Misal iterasi pertama:

  C_1 = 1110000 1100110 0101010 1011111

D_1 = 1010101 0110011 0011110 0011110

- $K_i = PC\text{-}2(C_iD^i)$

  Lakukan iterasi hingga didapatkan K_1-K_{16}

- Mencari $E(R_0)$

0	1	1	1	1	0
1	0	0	0	0	1
0	1	0	1	0	1
0	1	0	1	0	1
0	1	1	1	1	0
1	0	0	0	0	1
0	1	0	1	0	1
0	1	0	1	0	1

- Mencari $E(R_0)$ XOR K_1

0	1	1	0	0	0
0	1	0	0	0	1
0	1	1	1	1	0
1	1	1	0	1	0
1	0	0	0	0	1
1	0	0	1	1	0
0	1	0	1	0	0
1	0	0	1	1	1

- S-Box

  B1 = 011000

  Row: 1100

  Column: 00

  C1 = 0101

S_1															
14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

Lakukan sampai C8.

C = C1C2C3C4C5C6C7C8

- Permutasi tetap C

0	0	1	0
0	0	1	1
0	1	0	0
1	0	1	0
1	0	1	0
1	0	0	1
1	0	1	1
1	0	1	1

- $F(R_0, K_1)$ atau R_0 XOR K_1

  L2 = R1 = 1110 1111 0100 1010 0100 0101 0100 0100

- Lakukan iterasi sebanyak 16 kali sehingga didapat L16 dan R16

$E(R_0)$	=	011110100001010101010101011110100001010101010101
K_1	=	000110110000000101110111111111110001110000001110010
$E(R_0) \oplus K_1$	=	011000010001011110111010100001100110010100100111
S-box outputs		01011100100000101011010110010111
$f(R_0, K_1)$	=	00100011010010101010100110111011
$L_2 = R_1$	=	11101111010010100110010101000100

$E(R_1)$	=	011101011111010100101010000110000101010100000001001
K_2	=	011110011010111011011001110110111100100111100101
$E(R_1) \oplus K_2$	=	000011000100010010001101111010110110001111101100
S-box outputs		11111000110100000011101010101110
$f(R_1, K_2)$	=	00111100101010111000011110100011
$L_3 = R_2$	=	11001100000000010111011100001001

$E(R_2)$	=	111001011000000000000010101110101110100001010011
K_3	=	010101011111111001000101001000010110011111100111001
$E(R_2) \oplus K_3$	=	101100000111110010001000111110000010011111001010
S-box outputs		00100111000100001110000101101111
$f(R_2, K_3)$	=	01001101000101100110111010110000
$L_4 = R_3$	=	10100010010111000000101111110100

$E(R_3)$	=	010100000100000101111100000000101010111111110101001
K_4	=	011100101010110111010110110110110011010100011101
$E(R_3) \oplus K_4$	=	001000101110111110010111011011111001001010110100
S-box outputs		00100001111011011001111100111010
$f(R_3, K_4)$	=	10111011001000110111011101001100
$L_5 = R_4$	=	01110111001000100000000001000101

$E(R_4)$	=	101110101110100100000100000000000000001000001010
K_5	=	011111001110110000000111111010110101001110101000
$E(R_4) \oplus K_5$	=	110001100000010100000011111010110101000110100010
S-box outputs		01010000110010000011000111101011
$f(R_4, K_5)$	=	00101000000100111010110111000011
$L_6 = R_5$	=	10001010010011111010011000110111

$E(R_5)$	=	110001010100001001011111110100001100000110101111
K_6	=	011000111010010100111110010100000111101100101111
$E(R_5) \oplus K_6$	=	101001101110011101100001100000001011101010000000
S-box outputs		01000001111100110100110000111101
$f(R_5, K_6)$	=	10011110010001011100110100101100
$L_7 = R_6$	=	11101001011001111100110101101001

$E(R_6)$	=	11110101001010110000111111100101101010110101010011
K_7	=	11101100100001001011011111111011000011000101111100
$E(R_6) \oplus K_7$	=	00011001101011111011100000010011101100111111101111
S-box outputs		00010000011101010100000010101101
$f(R_6, K_7)$	=	10001100000001010001110000100111
$L_8 = R_7$	=	00000110010010101011101000010000

$E(R_7)$	=	00000000110000100101010101011111010000001010000
K_8	=	11110111100010100011101011000001001110111111011
$E(R_7) \oplus K_8$	=	11110111010010000110111110011110011110110101011
S-box outputs		01101100000110000111110010101110
$f(R_7, K_8)$	=	00111100000011101000011011111001
$L_9 = R_8$	=	11010101011010010100101110010000

$E(R_8)$	=	01101010101010110101001010100101011111100101000001
K_9	=	11100000110110111110101111101101111001111000000001
$E(R_8) \oplus K_9$	=	10001010011100001011100101001000100110110010000
S-box outputs		00010001000011000101011101110111
$f(R_8, K_9)$	=	00100010001101100111110001101010
$L_{10} = R_9$	=	00100100011111001100011001111010

$E(R_9)$	=	00010000100000111111100101100000110000111111010
K_{10}	=	10110001111100110100011110111010010001100100111
$E(R_9) \oplus K_{10}$	=	10100001011100001011111010110110100000101101111011
S-box outputs		11011010000001000101001001110101
$f(R_9, K_{10})$	=	01100010101111001001110000100010
$L_{11} = R_{10}$	=	10110111110101011101011110110010

$E(R_{10})$	=	01011010111111101010101111101010111111010110100101
K_{11}	=	00100001010111111101001111011110110100111000011 0
$E(R_{10}) \oplus K_{11}$	=	01111011101000010111100000110100001011100010001 1
S-box outputs		01110011000001011101000100000001
$f(R_{10}, K_{11})$	=	11100001000001001111101000000010
$L_{12} = R_{11}$	=	11000101011110000011110001111000

$E(R_{11})$	=	01100000101010111111000000011111100000111111 0001
K_{12}	=	01110101011100011111010110010100011001111110 1001
$E(R_{11}) \oplus K_{12}$	=	00010101110110100000010110001011111001000001 1000
S-box outputs		01111011100010110010011000110101
$f(R_{11}, K_{12})$	=	11000010011010001100111111101010
$L_{13} = R_{12}$	=	01110101101111010001100001011000

$E(R_{12})$	=	00111010101111011111110101000111100000010111 10000
K_{13}	=	10010111110001011101000111111010101110100100000 1
$E(R_{12}) \oplus K_{13}$	=	10101101011110000010101101110101101110001011000 1
S-box outputs		10011010110100011000101101001111
$f(R_{12}, K_{13})$	=	11011101101110110010100100100010
$L_{14} = R_{13}$	=	00011000110000110001010101011010

$E(R_{13})$	=	000011110001011000000110100010101010101011110100
K_{14}	=	010111101000011101101111111001011100111001111010
$E(R_{13}) \oplus K_{14}$	=	010100000101010110110001011110000100110111001110
S-box outputs		01100100011110011001101011110001
$f(R_{13}, K_{14})$	=	10110111001100011000111001010101
$L_{15} = R_{14}$	=	11000010100011001001011000001101

$E(R_{14})$	=	11100000010101000101100101001010110000000101101 1
K_{15}	=	10111111100100011000110100111101001111110000101 0
$E(R_{14}) \oplus K_{15}$	=	01011111110001011101010001110111111111110101000 1
S-box outputs		10110010111010001000110100111100
$f(R_{14}, K_{15})$	=	01011011100000010010011101101110
$L_{16} = R_{15}$	=	01000011010000100011001000110100

$E(R_{15})$	=	00100000011010100000010000011010010000011010100 0
K_{16}	=	11001011001111011000101100001110000101111111010 1
$E(R_{15}) \oplus K_{16}$	=	11101011010101111000111100010100010101100101110 1
S-box outputs		10100111100000110010010000101001
$f(R_{15}, K_{16})$	=	11001000110000000100111110011000
R_{16}	=	00001010010011001101100110010101

- Lakukan Inverse inisial permutasi terbalik

  Cipher text = IP^{-1} (R_{16} L_{16})

1	0	0	0	0	1	0	1
1	1	1	0	1	0	0	0
0	0	0	1	0	0	1	1
0	1	0	1	0	1	0	0
0	0	0	0	1	1	1	1
0	0	0	0	1	0	1	0
1	0	1	1	0	1	0	0
0	0	0	0	0	1	0	1

- Sehingga cipher text: 85E813540F0AB405 (Hexadesimal)

PROGRAM

Program DES Cipher ini dibuat menggunakan Visual Basic.

```
Module Module1
    Sub Main()
        Console.WriteLine("Algoritma DES (Data Encryption Standard)")

        '1. Tentukan kalimat yang akan dienkrip
        Console.WriteLine("Masukkan kalimat yang akan dienkrip: ")
        Dim input As String = Console.ReadLine
        Console.WriteLine("")
```

```
        '2. Tentukan kata kunci enkripsi yang digunakan
        Console.WriteLine("Masukkan kata kunci enkripsi: ")
        Dim kataKunci As String = Console.ReadLine
        Console.WriteLine("")

        '3. Lakukan inisialisasi variabel yang digunakan oleh metode ini
        Dim xc As New CryptCore()
        xc.InitCore()
        xc.Key = kataKunci

        '4. Lakukan enkripsi kalimat awal menggunakan algoritma ini
        Dim hasilEnkripsi As String = xc.Encrypt(input)
        Console.WriteLine("Hasil enkripsi kalimat input adalah: " &
vbCrLf & hasilEnkripsi.ToString & vbCrLf)

        '5. Lakukan dekripsi dari kalimat yang telah terenkripsi
        Dim hasilDekripsi As String = xc.Decrypt(hasilEnkripsi)
        Console.WriteLine("Hasil dekripsi dari kalimat terenkripsi
adalah: " & vbCrLf & hasilDekripsi & vbCrLf)

        Console.ReadLine()
    End Sub
End Module

'Class CryptCore adalah inti class untuk pemanggilan fungsi enkripsi dan
dekripsi pada algoritma DES (Data Encryption Standard)
Public Class CryptCore
    Private _key As String = Nothing
    Public Property Key() As String
        Get
            Return _key
        End Get
        Set(value As String)
            _key = Me.formatKey(value)
        End Set
    End Property

    Private Function formatKey(key As String) As String
```

```
    If key Is Nothing OrElse key.Length = 0 Then
        Return Nothing
    End If
    Return key.Trim()
End Function

Private DefaultKey As String = ""

Public Sub New()
    DefaultKey = "enkripsi"
End Sub

Private _coreSymmetric As CoreAlgoritmaSymmetric

Public Function InitCore() As Boolean
    _coreSymmetric = New CoreAlgoritmaSymmetric()
    Return True
End Function

Public Function Decrypt(src As String) As String
    Dim hasil As String = ""

    If _key Is Nothing Then
        hasil = _coreSymmetric.ProsesDecrypt(src, DefaultKey)
    Else
        hasil = _coreSymmetric.ProsesDecrypt(src, _key)
    End If

    Return hasil
End Function

Public Function Decrypt(src As String, key As String) As String
    Dim hasil As String = ""

    hasil = _coreSymmetric.ProsesDecrypt(src, key)

    Return hasil
End Function
```

```
    Public Function Encrypt(src As String) As String
        Dim hasil As String = ""

        If _key Is Nothing Then
            hasil = _coreSymmetric.ProsesEncrypt(src, DefaultKey)
        Else
            hasil = _coreSymmetric.ProsesEncrypt(src, _key)
        End If

        Return hasil
    End Function

    Public Function Encrypt(src As String, key As String) As String
        Dim hasil As String = ""

        hasil = _coreSymmetric.ProsesEncrypt(src, key)

        Return hasil
    End Function

    'Pada saat melakukan proses InitCore, maka proses tersebut akan
melakukan inisialisasi pada Class CoreAlgoritmaSymmetric
    Public Class CoreAlgoritmaSymmetric
        Private metodeEncode As System.Security.Cryptography.
SymmetricAlgorithm

        Public Sub New()
            metodeEncode = New System.Security.Cryptography.
DESCryptoServiceProvider()
        End Sub

        Private Function GetValidKey(Key As String) As Byte()
            Dim sTemp As String
            If metodeEncode.LegalKeySizes.Length > 0 Then
                Dim lessSize As Integer = 0, moreSize As Integer =
metodeEncode.LegalKeySizes(0).MinSize
```

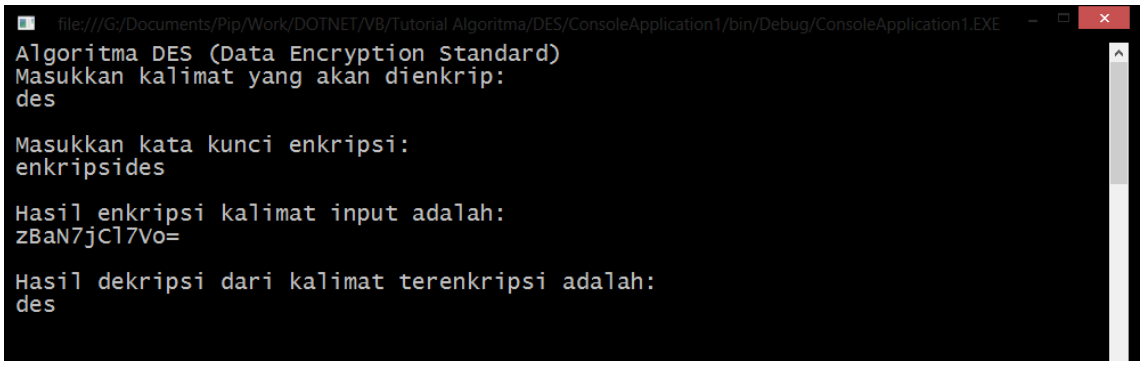
```
                While Key.Length * 8 > moreSize AndAlso metodeEncode.
LegalKeySizes(0).SkipSize > 0 AndAlso moreSize < metodeEncode.
LegalKeySizes(0).MaxSize
                    lessSize = moreSize
                    moreSize += metodeEncode.LegalKeySizes(0).SkipSize
                End While

                If Key.Length * 8 > moreSize Then
                    sTemp = Key.Substring(0, (moreSize / 8))
                Else
                    sTemp = Key.PadRight(moreSize / 8, " "c)
                End If
            Else
                sTemp = Key
            End If

            'Konversi kata kunci menjadi byte array
            Return System.Text.ASCIIEncoding.ASCII.GetBytes(sTemp)
        End Function

        Private Function GetValidIV(InitVector As [String], panjangValid
As Integer) As Byte()
            If InitVector.Length > panjangValid Then
                Return          System.Text.ASCIIEncoding.ASCII.GetBytes
(InitVector.Substring(0, panjangValid))
            Else
                Return          System.Text.ASCIIEncoding.ASCII.GetBytes
(InitVector.PadRight(panjangValid, " "c))
            End If
        End Function

        'Skrip ini akan menjalankan proses enkripsi yang terdapat dalam
Class CoreAlgoritmaSymmetric
        Public Function ProsesEncrypt(Source As String, Key As String)
As String
            If Source Is Nothing OrElse Key Is Nothing OrElse
Source.Length = 0 OrElse Key.Length = 0 Then
                Return Nothing
            End If
```

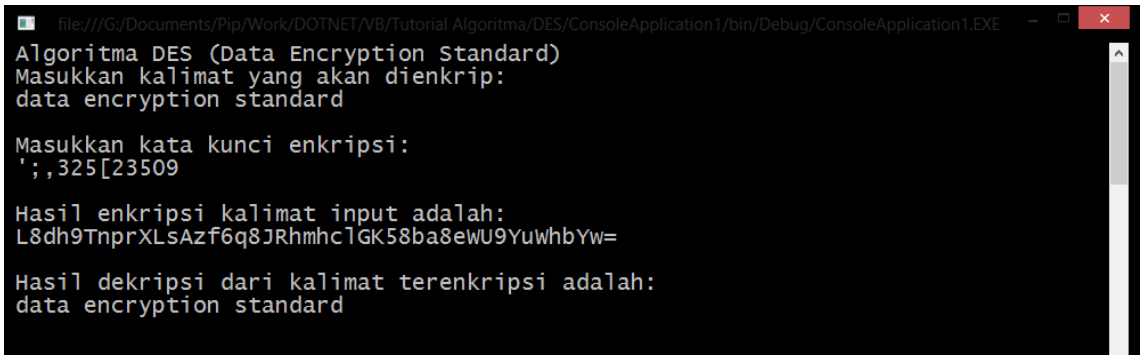
```
            If metodeEncode Is Nothing Then
                Return Nothing
            End If

            Dim lPanjangStream As Long
            Dim jumlahBufferTerbaca As Integer
            Dim byteBuffer As Byte() = New Byte(2) {}
            Dim srcData As Byte() = System.Text.ASCIIEncoding.ASCII.
GetBytes(Source)
            Dim encData As Byte()
            Dim streamInput As New System.IO.MemoryStream()
            streamInput.Write(srcData, 0, srcData.Length)
            streamInput.Position = 0
            Dim streamOutput As New System.IO.MemoryStream()
            Dim streamEncrypt As System.Security.Cryptography.
CryptoStream

            metodeEncode.Key = GetValidKey(Key)
            metodeEncode.IV = GetValidIV(Key, metodeEncode.IV.Length)

            streamEncrypt = New System.Security.Cryptography.
CryptoStream(streamOutput, metodeEncode.CreateEncryptor(), System.
Security.Cryptography.CryptoStreamMode.Write)
            lPanjangStream = streamInput.Length

            Dim totalBufferTerbaca As Integer = 0
            While totalBufferTerbaca < lPanjangStream
                jumlahBufferTerbaca = streamInput.Read(byteBuffer, 0,
byteBuffer.Length)
                streamEncrypt.Write(byteBuffer, 0, jumlahBufferTerbaca)
                totalBufferTerbaca += jumlahBufferTerbaca
            End While
            streamEncrypt.Close()

            encData = streamOutput.ToArray()

            'Konversi menjadi base64 agar dapat digunakan dalam xml
            Return Convert.ToBase64String(encData)
```

```
        End Function

        'Skrip ini akan menjalankan proses dekripsi yang terdapat dalam
Class CoreAlgoritmaSymmetric
        Public Function ProsesDecrypt(Source As String, Key As String)
As String
            If Source Is Nothing OrElse Key Is Nothing OrElse
Source.Length = 0 OrElse Key.Length = 0 Then
                Return Nothing
            End If

            If metodeEncode Is Nothing Then
                Return Nothing
            End If

            Dim lPanjangStream As Long
            Dim jumlahBufferTerbaca As Integer
            Dim byteBuffer As Byte() = New Byte(2) {}
            Dim encData As Byte() = Convert.FromBase64String(Source)
            Dim decData As Byte()
            Dim streamInput As New System.IO.MemoryStream(encData)
            Dim streamOutput As New System.IO.MemoryStream()
            Dim streamDecrypt As System.Security.Cryptography.
CryptoStream

            metodeEncode.Key = GetValidKey(Key)
            metodeEncode.IV = GetValidIV(Key, metodeEncode.IV.Length)

            streamDecrypt = New System.Security.Cryptography.
CryptoStream(streamInput, metodeEncode.CreateDecryptor(), System.
Security.Cryptography.CryptoStreamMode.Read)
            lPanjangStream = streamInput.Length

            Dim totalBufferTerbaca As Integer = 0
            While totalBufferTerbaca < lPanjangStream
                jumlahBufferTerbaca = streamDecrypt.Read(byteBuffer, 0,
byteBuffer.Length)
                If 0 = jumlahBufferTerbaca Then
                    Exit While
```

```
                End If

                streamOutput.Write(byteBuffer, 0, jumlahBufferTerbaca)
                totalBufferTerbaca += jumlahBufferTerbaca
            End While
            streamDecrypt.Close()

            decData = streamOutput.ToArray()
            For i As Integer = 0 To decData.Length - 1
                If decData(i) < 8 Then decData(i) = 0
            Next

            Dim encodeASCII As New System.Text.ASCIIEncoding()
            Return encodeASCII.GetString(decData)
        End Function
    End Class
End Class
```

Hasil screenshot program yang dijalankan:





DAFTAR PUSTAKA

Cryptography: Theory and Practice by Douglas Stinson

http://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/Data%20Encryption%20Standard%20(DES).pdf

http://itrezasaputra.blogspot.com/2016/03/contoh-soal-data-encryption-standard.html

http://octarapribadi.blogspot.com/2012/10/contoh-enkripsi-dengan-algoritma-des.html

http://studyinformatics.blogspot.com/2012/07/des-data-encryption-standard.html

https://makalah-update.blogspot.com/2012/11/makalah-pengertian-dan-sejarah-des-data.html

https://piptools.net/algoritma-des-data-encryption-standard/

https://www.academia.edu/12741838/Makalah-algoritma-kriptografi-des