IMPLEMENTASI KOMBINASI ALGORITMA GOST PADA KEAMANAN DATA TEXT DAN KOMPRESI DATA MENGGUNAKAN ALGORITMA DMC

Devi Afnita, Efori Buulolo, Noferianto Sitompul

Program Studi Teknik Informatika, STMIK Budi Darma, Medan, Indonesia
Jl. Sisingamangaraja No. 338 Simpang Limun, Medan, Indonesia
E-mail: devi.afnita95@gmail.com

Abstrak

Kriptografi berasal dari kata crypto yang berarti rahasia dan graphy yang berarti tulisan. Jadi kriptografi dapat diartikan sebagai tulisan rahasia. Secara istilah dapat didefinisikan sebagai studi tentang teknik-teknik matematika yang berhubungan dengan keamanan informasi. Untuk mengirimkan informasi secara cepat dengan ukuran *file* yang cukup besar pastinya akan membutuhkan waktu yang cukup lama. Kecepatan pengiriman ini tergantung dari ukuran informasi yang dikirimkan. Salah satu solusi untuk mengatasi masalah tersebut adalah dengan melakukan penempatan (kompresi) data sebelum melakukan pengiriman menggunakan algoritma kompresi dengan prinsip untuk memperkecil ukuran *file* tersebut, agar ketika proses pengiriman dapat dilakukan dengan cepat. Tujuan dari penelitian ini adalah untuk menggabungkan algoritma enkripsi *Government Standart* (Gost) dan algoritma kompresi *Dynamic Markov Compresion* (DMC).

Kata Kunci: *Kriptografi, Government Standart (Gost), Dynamic Markov Compresion (DMC).*

Abstract

Cryptography comes from the word crypto which means secret and graphy which means writing. So cryptography can be interpreted as a secret writing. The term can be defined as the study of mathematical techniques related to information security. To send information quickly with a large enough file size, it will certainly take a long time. This delivery speed depends on the size of the information sent. One solution to overcome this problem is by placing (compression) data before sending using a compression algorithm with the principle to reduce the size of the file, so that when the delivery process can be done quickly. The purpose of this study was to combine Government Standard (Gost) encryption algorithms and Dynamic Markov Compression (DMC) compression algorithms.

Keywords: *Cryptography, Government Standard (Gost), Dynamic Markov Compression (DMC).*

1. PENDAHULUAN

Kriptografi adalah suatu ilmu yang mempelajari bagaimana cara menjaga agar data atau informasi tetap aman, tanpa mengalami gangguan dari pihak ketiga. Kriptografi yang digunakan adalah Algoritma Gost. Algoritma Gost merupakan singkatan dari "*Gosudarstvennyi Standard*" atau "*Government Standard*". Algoritma ini merupakan suatu algoritma jumlah proses sebanyak 32 *round* (putaran) dan menggunakan 64 bit *block cipher* dengan 256 bit *key*. Metode GOST juga menggunakan 8 buah *S-Box* yang berbeda-beda dan operasi *XOR* serta *Left Circular Shift*. Untuk mengirimkan data text yang ber tife longtext secara cepat dengan ukuran *file* yang cukup besar pastinya akan membutuhkan waktu yang cukup lama. Kecepatan pengiriman ini tergantung dari ukuran informasi yang dikirimkan. Kompresi data adalah proses mengkodekan informasi menggunakan bit atau information-bearing unit yang lain yang lebih rendah daripada representasi data yang tidak terkodekan dengan suatu sistem enkoding tertentu. Algoritma yang digunakan yaitu algoritma *Dynamic Markov Compression* atau disingkat dengan DMC adalah salah satu metode kompresi *lossless*.

2. TEORITIS

2.1 ALGORITMA GOVERNMENT STANDARD (GOST)

GOST (*Government Standard)* merupakan algoritma *block cipher* yang dikembangkan oleh seseorang yang berkebangsaan Uni Soviet. Kemudian, metode ini dikembangkan oleh pemerintah Uni Soviet pada masa perang dingin untuk menyembunyikan data atau informasi yang bersifat rahasia pada saat komunikasi. Algoritma ini merupakan suatu algoritma jumlah proses sebanyak 32 *round* (putaran) dan menggunakan 64 bit *block cipher* dengan 256 bit *key*. Metode GOST juga menggunakan 8 buah *S-Box* yang berbeda-beda dan operasi *XOR* serta *Left Circular Shift. Algoritma Gost* merupakan blok *cipher* 64 bit dengan panjang kunci 256 bit (Saarinen, 1998). Algoritma ini mengiterasi algoritma enkripsi sederhana sebanyak 32 putaran (*round*) (Saarinen, 1998). Untuk

mengenkripsi pertama-tama plaintext 64 bit dipecah menjadi 32 bit bagian kiri, L dan 32 bit bagian kanan, R. Subkunci (*subkey*) untuk putaran *i* adalah *Ki*.

1. Proses Pembangkitan Kunci Internal
   Kunci internal pada algoritma GOST dibangkitkan dari kunci eksternal yang diberikan oleh pengguna. Setelah pengguna memberikan kunci eksternal, ubah kunci menjadi bit biner. Pembangkitan kunci internal dilakukan dengan membagi kunci eksternal 256 bit (k_1, k_2, k_3, k_4, …, k_{256}) ke dalam delapan bagian yang masing-masing panjangnya 32 bit. Pembagiannya adalah sebagai berikut [11]:
   K0 = (k_{32}, …, k_1)
   K1 = (k_{64}, …, k_{33})
   K2 = (k_{96}, …, k_{65})
   K3 = (k_{128}, …, k_{97})
   K4 = (k_{160}, …, k_{129})
   K5 = (k_{192}, …, k_{161})
   K6 = (k_{224}, …, k_{193})
   K7 = (k_{256}, …, k_{225})
2. Proses Enkripsi
   a. 64 bit *plaintext* dibagi menjadi 2 buah bagian 32 bit, yaitu L_i dan R_i.
      Caranya: *Input* a1(0), a2(0), …, a32(0), b1(0), …, b32(0)
   b. R0 = a32(0), a31(0), …, a1(0)
      L0 = b32(0), b31(0), …, b1(0)
   c. (R_i + K_i) mod 2^{32}. Hasil dari penjumlahan modulo 2^{32} (4294967296) berupa 32 bit
   d. Hasil dari penjumlahan modulo 2^{32} dibagi menjadi 8 bagian, dimana masing-masing bagian terdiri dari 4 bit. Setiap bagian dimasukkan ke dalam tabel *S-Box* yang berbeda, 4 bit pertama menjadi input dari *S-Box* 0, 4 bit kedua menjadi *S-Box* 1, dan seterusnya.

Tabel 1. *S-Box* Algoritma GOST

Tabel S-Box	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S-Box 0	4	10	9	2	13	8	0	14	6	11	1	12	7	15	5	3
S-Box 1	14	11	4	12	6	13	15	10	2	3	8	1	0	7	5	9
S-Box 2	5	8	1	13	10	3	4	2	14	15	12	7	6	0	9	11
S-Box 3	7	13	10	1	0	8	9	15	14	4	6	12	11	2	5	3
S-Box 4	6	12	7	1	5	15	13	8	4	10	9	14	0	3	11	2
S-Box 5	4	11	10	0	7	2	1	13	3	6	8	5	9	12	15	14
S-Box 6	13	11	4	1	3	15	5	9	0	10	14	7	6	8	2	12
S-Box 7	1	15	13	0	5	7	10	4	9	2	3	14	6	11	8	12

   e. Hasil yang didapat dari substitusi ke S-Box kemudian digabungkan kembali menjadi 32 bit dan kemudian dilakukan RLS (*Rotate Left Shift* / pergeseran ke kiri) sebanyak 11 bit.
   f. R_{i+1} = RLS XOR L_i.

Tabel 2. Kebenaran Gerbang XOR

A	B	Y = A ⊕ B
0	0	0
0	1	1
1	0	1
1	1	0

   g. L_{i+1} = R_i sebelum dilakukan proses.
      Langkah nomor 2 sampai 6 dilakukan sebanyak 32 kali (putaran). Pada langkah nomor 2 penggunaan kunci dijadualkan penggunaanya sesuai dengan putarannya.

Tabel 3. Penjadwalan Kunci Internal Enkripsi GOST

Putaran	0	1	2	3	4	5	6	7
Kunci Internal	K0	K1	K2	K3	K4	K5	K6	K7
Putaran	8	9	10	11	12	13	14	15
Kunci Internal	K0	K1	K2	K3	K4	K5	K6	K7
Putaran	16	17	18	19	20	21	22	23
Kunci Internal	K0	K1	K2	K3	K4	K5	K6	K7
Putaran	24	25	26	27	28	29	30	31
Kunci Internal	K7	K6	K5	K4	K3	K2	K1	K0

Proses enkripsi dilakukan sebanyak 32 round (mulai dari Round 0 s/d 31) Pseudocode untuk setiap round :

j = 0 //array bit kunci; nilai I adalah putaran
For i = 0 to 31
//kelompok bit plaintext
L[i] = 32 bit L[i] R[i] = 32 bit R[i]
//jumlahkan R[i] dan L[i], kemudian mod2^{32}
if i < 24 then
if j = 7 then j = 0
(R[i] + K[j]) Mod 2^{32}
j = j +1
else
j = 7
(R[i] + K[j]) Mod 2^{32}
j = j – 1 End If

Hasil (R[i] + K[j]) Mod 232 Pecah menjadi 8 kelompok (4 bit per kelompok), kemudian permutasikan dengan SBox Gabungkan biner-biner hasil permutasi SBox. RLS = Lakukan proses Rotate Left Shift dari bit gabungan sebanyak 11 bit (dari kiri ke kanan)

If i < 31 then
R[i +1] = RLS[i] XOR L[i]
L[i +1] = R[i] sebelum diproses
Else
R[i +1] = R[i] sebelum proses
L[i +1] = RLS[i] XOR L[i]
R[i +1] =Reverse R[i]
L[i+1] = Reverse L[i]
Cipher = Reverse R[i], Reverse L[i]
Next i

Aturan Penggunaan Kunci :
Round 0 s/d Round 23 □ K[0] ... K[7]
Round 24 s/d 31 □ K[7] ... K[0]

3. Proses Dekripsi

Tabel 4. Penjadwalan Kunci Internal Dekripsi GOST

Putaran	0	1	2	3	4	5	6	7
Kunci Internal	K0	K1	K2	K3	K4	K5	K6	K7
Putaran	8	9	10	11	12	13	14	15
Kunci Internal	K7	K6	K5	K4	K3	K2	K1	K0
Putaran	16	17	18	19	20	21	22	23
Kunci Internal	K7	K6	K5	K4	K3	K2	K1	K0
Putaran	24	25	26	27	28	29	30	31
Kunci Internal	K7	K6	K5	K4	K3	K2	K1	K0

Proses dekripsi dilakukan sebanyak 32 round (mulai dari Round 0 s/d 31) sama seperti proses enkripsi
Pseudocode untuk setiap round :

```
    j= 0  //array bit kunci; nilai i adalah putran
    For i = 0 to 31
//kelompok bit ciphertext
L[ i ] = 32 bit L[ i ] R[ i ] = 32 bit R[ i ]
//jumlahkan R[ i ] dan L[ i ], kemudian mod 2^32
        if i  < 8 then
            (R[ i ] + K[ j ] ) Mod 2^32
            j = j +1
        else
            j = 7
            if  j = 0 then
                j = j + 7
            else
                j = j - 1
            end if
            (R[ i ] + K[ j ] ) Mod 2^32
        End If
```

Hasil (R[i] + K[j]) Mod 2^{32} Pecah menjadi 8 kelompok (4 bit per kelompok), kemudian permutasikan dengan SBox. Gabungkan biner-biner hasil permutasi SBox. RLS = Lakukan proses Rotate Left Shift dari bit gabungan sebanyak 11 bit (dari kiri ke kanan)

```
If i < 31 then
    R[ i +1] = RLS[ i ]  XOR  L[ i ]
    L[ i +1 ] = R[ i ]  sebelum diproses
Else
        R[ i +1] = R[ i ] sebelum proses
        L[ i +1 ] = RLS[ i ]  XOR L[ i ]

        R[ i +1] =Reverse R[ i ]
        L[i+1] = Reverse L[ i ]
        Cipher = Reverse R[ i ], Reverse L[ i ]
End if
Next i
```

Aturan Penggunaan Kunci :
Round 0 s/d Round 7 □ K[0] ... K[7]
Round 8 s/d 31 □ K[7] ... K[0]

2.2 ALGORITMA *DYNAMIC MARKOV COMPRESION* (DMC)

Algoritma *Dynamic Markov Compression* (DMC) merupakan kompresi yang tergolong dengan teknik *lossess* yaitu tidak menghilangkan informasi sedikitpun. Algoritma *Dynamic Markov Compression* pertama kali diperkenalkan oleh *Gordon Cormack* dan *Nigel Horspool*, algoritma ini menggunakan pengkodean aritmatika mirip dengan prediksi oleh pencocokan sebagai (PPM), kecuali bahwa input diperkirakan satu bit pada satu waktu (bukan dari satu *byte* pada suatu waktu) [18].

Berikut adalah Langkah algoritma *Dynamic Markov Compression* (DMC) sebagai berikut :

1. Urutkan karakter berdasarkan frekuensi kemunculan karakter.
2. Setiap karakter dinyatakan sebagai daun atau pohon bersimpul tunggal.
3. Untuk setiap bit pada langkah 1, lakukan traversal pada cabang yang bersesuaian.
4. Ulangi langkah 1, 2 dan 3 sampai bertemu daun. Kode kerangkaian bit yang telah di baca dengan karakter di daun.

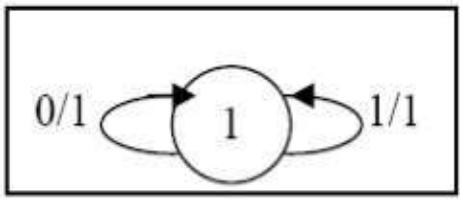
Dinamic Markov Compression (DMC) merupakan teknik pemodelan yang didasarkan pada model *finite-state* dan memiliki rasio kompresi yang baik dan kecepatan moderat, mirip dengan PPM, tapi memerlukan sedikit lebih banyak memori dan tidak diterapkan secara luas. Algoritma DMC merupakan teknik kompresi yang adaptif, karena struktur mesin *finite-state* berubah seiring dengan pemrosesan *file*. Pada DMC, simbol alfabet input proses per bit, bukan per *byte*. Setiap output transisi menandakan berapa banyak simbol yang muncul. Perhitungan tersebut dipakai untuk memperkirakan probabilitas dari transisi.

Berikut adalah algoritma *Dynamic Markov Compression* (DMC) sebagai berikut :
Algoritma kompresi DMC :

Prob {digit = 0 | current state = A} = *n0 / (n0 + n*1)
Prob {digit = 1 | current state = A} = *n*1 / *(n0 + n*1)
Prob {digit = 0 | current state = A}
= *(n0 + c) / (n0 + n*1 + *2c)*
Prob {digit = 1 | current state = A}
= (*n*1 + *c) / (n0 + n*1 + *2c)*

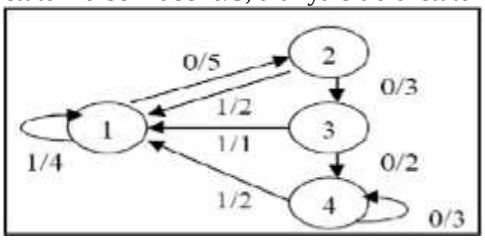
1. s ← 1 (jumlah state sekarang)
2. t ← 1 (state sekarang)
3. T [1][0] = T [1][1] ← 1 (model inisialisasi)
4. C[1][0] = C [1][1] ← 1 (inisialisasi untuk menghindari masalah frekuensi nol)
5. Untuk setiap input bit e :
   a. U ← t
   b. T ← T [u][e] (ikuti transisi)
   c. Kodekan e dengan probabilitas : C[u][e] / (C[u][0] + C[u][1])
   d. C[u][e] ← C[u][e] + 1
   e. Jika ambang batas *cloning* tercapai, maka :
      1. S ← s + 1 (state baru t)
      2. T[u][e] ← s ; T[s][0] ← T[t][0] ; T[s][1] ← T[t][1]
      3. Pindahkan beberapa dari C[t] ke C[s]

Model awal yang digunakan berupa mesin *FSA* dengan transisi 0/1 dan 1/1. Secara umum transisi ditandai dengan 0/p atau 1/q dengan p dan q menunjukkan jumlah transisi dari *state* dengan input 0 atau 1.



Gambar 1. Model Awal DMC

Nilai probabilitas bahwa input selanjutnya bernilai 0 adalah *p/(p+q)* dan input selanjutnya bernilai 1 adalah *q/(p+q)*. Transisi yang keluar dari *state* 1 diberi label 0/5, artinya bit 0 di *state* 1 terjadi sebanyak 5 kali.



Gambar 2. Tampilan Model DMC

Masalah tidak terdapatnya kemunculan suatu bit pada *state* dapat diatasi dengan menginisialisasi model awal *state* dengan satu. Probabilitas dihitung menggunakan frekuensi relatif dari dua transisi yang keluar dari *state* yang baru. Jika frekuensi transisi dari suatu *state t* ke *state* sebelumnya, yaitu *state u* sangat tinggi, maka *state t* dapa di *cloning*. Ambang batas nilai *cloning* harus disetujui oleh *encoder* dan *decoder*. Aturan *cloning* adalah sebagai berikut :

1. Semua transisi dari *state u* dikirim ke *state t*. Semua transisi dari *state* lain ke *state t* tidak berubah.
2. Jumlah transisi yang keluar dari *t*, harus mempunyai rasio yang sama (antara 0 dan 1) dengan jumlah transisi yang keluar dari *t*.

Jumlah transisi yang keluar dari *t* dan diatur agar mempunyai nilai yang sama dengan jumlah transisi yang masuk.

3. ANALISA DAN PEMBAHASAN

3.1 Analisa Masalah

Analisa adalah penguraian dari suatu pembahasan, dalam hal ini membahas mengenai penerapan algoritma GOST dalam mengamankan sebuah data teks dan kompresi menggunakan algoritma DMC. Secara garis besar penulis melakukan analisa masalah tersebut karena dilatar belakangi dari masalah keamanan data yang merupakan

suatu aspek penting dalam pengiriman data maupun informasi melalui jaringan, Hal ini disebabkan karena kemajuan di bidang jaringan computer dengan konsep open system-nya, sehingga memudahkan seseorang untuk masuk ke dalam jaringan tersebut.

3.2 Penerapan Penerapan Kombinasi Enkripsi Gost – Kompresi DMC

Proses Enkripsi :

1. Konversi Plaintext dan Kunci ke Biner
   Plainteks = affsheen
   Kunci= devi_afnita_affsheen_myesha_sary
   Plainteks :

Char	a	f	f	s	h	e	e	n
Dec	97	102	102	115	104	101	101	110
Biner	0110 0001	0110 0110	0110 0110	0111 0011	0110 1000	0110 0101	0110 0101	0110 1110

Gabungan Biner Plainteks :
01100001011001100110011001110011011010000110010101100101011101110
Kunci :

konversi kunci						
char	desimal	biner		char	desimal	biner
d	100	01100100		h	104	01101000
e	101	01100101		e	101	01100101
v	118	01110110		e	101	01100101
i	105	01101001		n	110	01101110
_	95	01011111		_	95	01011111
a	97	01100001		m	109	01101101
f	102	01100110		y	121	01111001
n	110	01101110		e	101	01100101
i	105	01101001		s	115	01110011
t	116	01110100		h	104	01101000
a	97	01100001		a	97	01100001
_	95	01011111		_	95	01011111
a	97	01100001		s	115	01110011
f	102	01100110		a	97	01100001
f	102	01100110		r	144	01110010
s	115	01110011		y	121	01111001

Biner kunci seluruhnya:
01100100011001010111011001101001010111110110000101100110011011100110100101110100011000010101111101100001011001100110011001110011011010000110010101100101011011100101111101101101011100101011100100111100101
Pengelompokan Kunci

Kunci	Posisi Bit	Biner yang diambil
K[0]	32...1	10010110011011101010011000100110
K[1]	64...33	01110110011001101000011011111010
K[2]	96...65	11111010100001100010111010010110
K[3]	128...97	11001110011001100110011010000110
K[4]	160...129	01110110101001101010011000010110
K[5]	192...161	10100110100111101011011011111010
K[6]	224...193	11111010100001100001011011001110
K[7]	256...225	10011110010011101000011011001110

Gabungan Biner Plainteks :
01100001011001100110011001110011011010000110010101100101011011 10
01100001011001100110011001110011→R0
01101000011001010110010101101110 →L0
L[0]= 01110110011001101010011000010110
R[0]= 11001110011001100110011010000110
Putaran 0→ i=0

1. L[0]=01110110011001101010011000010110
   R[0]= 11001110011001100110011010000110
2. (R[0]+K[0]) mod 2^{32}
   R[0] = 3462817414
   K[0] = 2523833894 +
   = 5986651308 mod 2^{32}
   = 1691684012
   = 01100100110101010000110010101100
3. Pengelompokan

Biner Kelompok	Dec Nilai Bit	SBOX	Hasil Permutasi dengan SBOX	Biner
0110	6	→S-Box(0)	0	0000
0100	4	→S-Box(1)	6	0110
1101	13	→S-Box(2)	0	0000
0101	5	→S-Box(3)	8	1000
0000	0	→S-Box(4)	6	0110
1100	12	→S-Box(5)	9	1001
1010	10	→S-Box(6)	14	1110
1100	12	→S-Box(7)	6	0110

4. Gabungkan biner- biner hasil pertukaran dari tabel Sbox dan lakukan Rotate Left Shift 11 bit.
   Gabungan = 00000110000010000110100111100110
   RLS[0] 11 bit = 01000011010011110011000000110000
5. R[1]=RLS[0] XOR L[0]
   RLS[0]= 01000011010011110011000000110000
   L[0] = 01110110011001101010011000010110 ⊕
   R[1] = 00110101001010011001011000100110

L[1]=R[0] sebelum diproses
L[1]= 11001110011001100110011010000110

6. Hasil Putaran -0 atau pada PUTARAN – i = 0 adalah :
   L[1] = 11001110011001100110011010000110
   R[1] = 00110101001010011001011000100110

Putaran 1→ i=1

1. L[1]= 11001110011001100110011010000110
   R[1]= 00110101001010011001011000100110
2. (R[1]+K[1]) mod 2^{32}
   R[1] = 891917862
   K[1] = 1986430714 +
   = 2878348576 mod 2^{32}
   = 2878348576
   = 10101011100100000001110100100000
3. Pengelompokan

Biner Kelompok	Dec Nilai Bit	SBOX	Hasil Permutasi dengan SBOX	Biner
1010	10	→S-Box(0)	1	0001
1011	11	→S-Box(1)	1	0001
1001	9	→S-Box(2)	15	1111
0000	0	→S-Box(3)	7	0111
0001	1	→S-Box(4)	12	1100
1101	13	→S-Box(5)	12	1100
0010	2	→S-Box(6)	4	0100
0000	0	→S-Box(7)	1	0001

4. Gabungkan biner- biner hasil pertukaran dari tabel Sbox dan lakukan Rotate Left Shift 11 bit.
   Gabungan = 00010001111101111100110001000001
   RLS[1] 11 bit= 10111110011000100000100010001111
5. R[2]=RLS[1] XOR L[1]
   RLS[1]= 10111110011000100000100010001111
   L[1] = 11001110011001100110011010000110 ⊕
   R[2] = 01110000000001000110111000001001
   L[2]=R[1] sebelum diproses
   L[2]= 00110101001010011001011000100110
6. Hasil Putaran -1 atau pada PUTARAN – i = 1 adalah :
   L[2] = 00110101001010011001011000100110
   R[2] = 01110000000001000110111000001001

Putaran 31 → i = 31

1. L[31]= 11100110011010111001011010011101
   R[31]= 10011110001011011101010110000101
2. (R[31]+K[0]) mod 2^{32}
   R[31] = 2653803909
   K[0] = 2523833894 +
   = 5177637803 mod 2^{32}
   = 882670507
   = 00110100100111000111101110101011
3. Pengelompokan

Biner Kelompok	Dec Nilai Bit	SBOX	Hasil Permutasi dengan SBOX	Biner

0011	3	→S-Box(0)	2	0010
0100	4	→S-Box(1)	6	0110
1001	9	→S-Box(2)	15	1111
1100	12	→S-Box(3)	11	1011
0111	7	→S-Box(4)	8	1000
1011	11	→S-Box(5)	5	0101
1010	10	→S-Box(6)	14	1110
1011	11	→S-Box(7)	14	1110

4. Gabungkan biner- biner hasil pertukaran dari tabel Sbox dan lakukan Rotate Left Shift 11 bit.
   Gabungan = 00100110111110111000010111101110
   RLS[31] 11 bit = 11011100001011110111000100110111
5. R[32]= R[31] sebelum diproses
   R[32]= 10011110001011011101010110000101
6. L[32]=RLS[31] XOR L[31]
   RLS[31] = 11011100001011110111000100110111
   L[31] = 11100110011010111001011010011101 ⊕
   L[32] = 00111010010001001110011110101010
7. Pembalikan penulisan biner biner L[32] dan R[32]
   L[32] = 01010101111001110010001001011100
   R[32] = 10100001101010111011010001111001
8. Gabungan R[32] dan L[32]
   10100001101010111011010001111001010101011110011100100010010111 00
9. Kelompokan bit hasil 8 bit per kelompok, kemudian masing – masing kelompok bit konversikan menjadi karakter menjadi :
   Hasil Chipertext : ¡ « ´ y U ç " \

10100001	10101011	10110100	01111001	01010101	11100111	00100010	01011100
¡	«	´	y	U	Ç	"	\

Proses Kompresi Algoritma DMC

Setelah mendapatkan hasil dari proses enkripsi algoritma gost, maka tahap selanjutnya melakukan proses kompresi mengunakan metode *Dynamic Markov Compression* (DMC). Isi ciphertext yang sudah diamankan datanya dengan proses enkripsi algoritma gost adalah sebagai berikut : "¡ « ´ y U ç " \" . Maka dilakukan pendataan Nilai ASCII dari "¡ « ´ y U ç " \", dimana ¡= 161, « =171, ´=180, y =121, U = 85, ç = 231, " = 34, \= 92.
Akhirnya nilai ASCII ini berubanh menjadi karakter – karakter simbol :
161=¡, 32= , 209= Ñ, 204= Ì, 83=S, 168= ¤, 114=r, 76= L, 40= (, 76=L, 119= w, 254= þ, 220= Ü
Hasil akhir adalah: ¡ ÌS¤rL(LwþÜ ,Seperti pada contoh sebelumnya, representasi biner dari string ¡ ÑÌS¤rL(LwþÜ" adalah sebagai berikut:
"10100001001000001101000111001100010100111010010001110010010011000010100001001100011101111 111111011011100 "

Tabel 5. Tabel Probabilitas

No.	Nilai	Frekuensi	Probabilitas
1.	0	5	5/8 = 0,625
2.	1	3	3/8 = 0,375
3.	00	1	1/8 = 0,125
4.	01	1	1/8 = 0,125
5	10	2	2/8 = 0,25

Probabilitas {digit = 0 | current state = A} = n0/(n0+n1)
= 5/(5+3)
= 5/8
= 0,625
Probabilitas {digit = 1 | current state = A} = n1/(n0+n1)
= 3/(5+3)
= 3/8
= 0,375

Dengan dijumlahkan pada *state* 1 dan 0 pada bilangan biner 01100001 yaitu hasilnya 0,625 + 0,375 = 1, maka jumlah dari *state* dari sebanyak 8 bit.

Tabel 6. Nilai Desimal

Char	Ascii Desimal	Char	Ascii Desimal
¡	161	L	76
□	32	w	119
Ñ	209	Þ	254
Ì	204	Ü	220
S	83		
¤	168		
r	114		
L	76		
(	40		

Dalam *teks* yaitu dengan ukuran 104 bit (13 *byte*), maka dapat di buat nilai probabilitasnya pada tabel dibawah ini, yaitu :

Tabel 7. Nilai Probabilitas Teks

Frekuensi	Probabilitas	Nilai
1	11/13	161,32,209,204,83,168,114,40,119,254,220
2	1/13	76

Dalam nilai probabilitas 1 dan 2, dijumlahkan dengan mengalikan dengan 8 bit, yaitu sebagai berikut :
1 x 8 bit (10100001) = 8 bit (Karakter ¡ = 161)
1 x 8 bit (00100000) = 8 bit (Karakter □ = 32)
1 x 8 bit (11010001) = 8 bit (Karakter Ñ = 209)
1 x 8 bit (11001100) = 8 bit (Karakter Ì = 204)
1 x 8 bit (01010011) = 8 bit (Karakter S = 83)
1 x 8 bit (10100100) = 8 bit (Karakter ¤ = 168)
1 x 8 bit (01110010) = 8 bit (Karakter r = 114)
1 x 8 bit (00101000) = 8 bit (Karakter (= 40)
1 x 8 bit (01110111) = 8 bit (Karakter w = 119)
1 x 8 bit (11111110) = 8 bit (Karakter þ = 254)
1 x 8 bit (11011100) = 8 bit (Karakter Ü = 220)
2 x 8 bit (01001100) = 16 bit (Karakter L = 76)

Pada *teks* tersebut memiliki bilangan bit sebanyak 104 bit (13 *byte*) yang terdiri dari nilai 0 dan 1 yang berjumlah masing-masing yaitu nilai 0 =55 , dan nilai 1 =49. Tabel frekuensi pada dibawah ini, sebagai berikut :

Tabel 8. Nilai Probabilitas 0 dan 1

Frekuensi	Probabilitas	Nilai
1	11/13	161,32,209,204,83,168,114,40,119,254,220

Dalam nilai probabilitas 1, dijumlahkan dengan mengalikan dengan 8 bit, yaitu sebagai berikut :
1 x 8 bit (10100001) = 8 bit (Karakter ¡ = 161)
□

1 x 8 bit (00100000) = 8 bit (Karakter = 32)
1 x 8 bit (11010001) = 8 bit (Karakter Ñ = 209)
1 x 8 bit (11001100) = 8 bit (Karakter Ì = 204)
1 x 8 bit (01010011) = 8 bit (Karakter S = 83)
1 x 8 bit (10100100) = 8 bit (Karakter ¤ = 168)
1 x 8 bit (01110010) = 8 bit (Karakter r = 114)
1 x 8 bit (00101000) = 8 bit (Karakter (= 40)
1 x 8 bit (01110111) = 8 bit (Karakter w = 119)
1 x 8 bit (11111110) = 8 bit (Karakter þ = 254)
1 x 8 bit (11011100) = 8 bit (Karakter Ü = 220)
Jadi jumlah dari bilangan adalah sebagai berikut 8 bit x 11 = 88 bit

Tabel 9. Nilai Probabilitas Digit 0 dan 1

No.	Nilai	Frekuensi	Probabilitas
1.	0	55	55/104 = 0,528
2.	1	49	49/104= 0,471

Probabilitas {digit = 0 | current state = A} = n0/(n0+n1)
= 55/(55+49)
= 55/104
= 0,528
Probabilitas {digit = 1 | current state = A} = n1/(n0+n1)
= 49/(55+49)
= 49/104
= 0,471
Untuk menghitung formula yang tidak terdefinisi dengan nilai e= 104, yaitu :
Probabilitas {digit = 0 | current state = A} =
(n0+e)/(n0+n1+2e)
= (55+104)/(55+49+208)
= 159/312
= 0,5096%
Probabilitas {digit = 1 | current state = A} = (n1+e)/(n0+n1+2e)
= (49+104)/(55+49+208)
= 153/312
= 0,4903%

Jadi, formula pada digit 0 sebesar 0,5096% dan digit 1 sebesar 0,4903% setelah file asli terkompres. Maka *file* sebelum dikompresi sebesar 104 bit (13 *byte)*, dan setelah dikompresi maka akan didapatkan sebesar 88 bit *(*11 *byte).* Maka perbandingan nilai data sebelum dan setelah di kompres adalah sebagai berikut :

1. Ratio Of Compression (RC) = (ukuran sesudah / ukuran sebelum)
=104/88
= 1,1818
2. Compression Of Ratio (CR) = (ukuran sesudah / ukuran sebelum)*100%
= (88/104)*100%
= 0,8461 (84,61%)
3. Space Saving (SS) = 1 - CR
= 1 - 0,8461
= 0,1539 (15,39%)

4. KESIMPULAN

Berdasarkan hasil penelitian simulasi implementasi kriptografi algoritma Gost dan kompresi dengan algoritma DMC pada data text adalah sebagai berikut :

1. Hasil dari pengujian enkripsi-kompresi dapat merubah ukuran data menjadi lebih kecil dengan presentase penghematan ruangan 1.79% memiliki nilai *Space Saving* dengan rata- rata 44.37% dan nilai *Compression Of Ratio* dengan rata-rata 57.70%.
2. Semakin besar kapasitas data yang dienkripsi atau kompresi maka semakin besar pula waktu proses enkripsi dan kompresi yang di butuhkan.

3. Semakin banyak jumlah kunci enkripsi Gost, maka semakin lama waktu yang dibutuhkan untuk memecahkan enkripsinya dan tingkat keamanan semakin tinggi.
4. Nilai rasio kompresi dipengaruhi oleh isi dari data yang dikompresi. Semakin banyak pengulangan karakter pada suatu data yang dikompresi maka semakin tinggi pula rasio kompresinya.

5. REFERENSI

[1] S. Gustaria, & T. Fatimah, "Implementasi Kriptografi Menggunakan Algoritma Gost (*Gosudartevvenyi Standart*) Untuk Pengiriman E-Mail Pada Aplikasi Cirus-Mail Berbasis Web", Fakultas Teknologi Informasi, Universitas Budi Luhur, 2013

[2] S. Bunton, "*The Structure of DMC", Departement Of Computer Science and Engineering, University of Washington*, 1995

[3] E.R. Agustina, & A. Kurniati,"Pemanfaatan Kriptografi Dalam Mewujudkan Keamanan Informasi Pada *e-Voting* Di Indonesia", UPN Veteran Yogyakarta, 2009

[4] D. Wirdasari, "Prinsip Kerja dalam Mengamankan Informasi", STMIK Triguna Darma, vol.II, 2008

[5] A.Hidayatullah, & E.Insanudin," Pengenalan Kriptografi dan Pemakainan Sehari-Hari", Fakultas Sains dan Teknologi, Universitas Islam Negeri Sunan Gunung Djati Bandung, 2016

[6] M.M.Amin, " Implementasi Kriptografi Klasik pada Komunikasi Berbasis Teks", Politeknik Negeri Sriwijaya Palembang, vol. III, 2016

[7] Sumandri, "Studi Model Algoritma Kriptografi Klasik dan Modern", Universitas Negeri Yogyakarta, 2017

[8] S. Gustaria, "Implementasi Kriptografi Menggunakan Algoritma Gost (*Gosudartevvenyi Standart*) Untuk Pengiriman E-Mail Pada Aplikasi Cirus-Mail Berbasis Web", Fakultas Teknologi Informasi, 2013

[9] Karima, & M.N.Diyatan, "Algoritma Kriptografi Gost Dengan Implementasi MD5 Untuk Meningkatkan Nilai *Avalanche Effect*", Fakultas Ilmu Komputer, Universitas Dian Nuswantoro, vol. XV, 2016

[10] Hendri, " Pengamanan Aplikasi Chatting Menggunakan Metode Kriptografi *Goverment Standart* ", Time Ilmiah INTI, Vol.XII, 2017

[11] J. Siregar,"Implemetasi Keamanan Data Teks Dengan Algoritma GOST dan ROT13", STMIK Budidarma Medan, 2017