KETIK : Jurnal Informatika
ISSN: 3026-183X (Media Online)
Vol. 01, No. 03, Januari 2024 Hal. 30-46
Faatuatua Media Karya
https://jurnal.faatuatua.com/index.php/KETIK

Analisis dan Implementasi Modifikasi Algoritma Kriptografi GOST Menggunakan Blum Blum Shub Generator Pada Sistem Pengamanan Login Pada Website

Sri Hartati Waruwu[1*], Rivalri Kristianto Hondro[2], Sarwandi[3]

[1]Universitas Budi Darma, Indonesia, email: srihartatiwaruwu@gmail.com*
[2]Universitas Budi Darma, Indonesia, email: rivalryhondro@gmail.com
[3]Universitas Budi Darma, Indonesia, email: sarwandi@gmail.com
**coressponding author)*

Info Artikel

Diajukan: *-*
Diterima: *-*
Diterbitkan: *-*

Kata Kunci:
Analisis, Implementasi, Modifikasi, GOST, Blum Blum Shub, Generator, Login, Website

Keywords:
Analysis, Implementation, Modification, GOST, Blum Blum Shub, Generator, Login, Website

Lisensi: *cc-by-sa*

Copyright © 2023 by Author. Published by Faatuatua Media Karya

Abstrak

Salah satu cara yang dapat di gunakan dalam mengamankan pengiriman informasi, yaitu dengan menyandingkan informasi menjadi kode-kode yang tidak dengan mudah dimengerti oleh orang-orang yang tidak berkepentingan, Teknik ini disebut Kriptografi. Analisa terhadap penggabungan dan modifikasi gost metode tersebut dilakukan dengan menggunakan Analisa matematis dan Analisa program. Algoritma GOST adalah salah satu algoritma kriptografi simetris yang beroperasi pada ukuran blok pesan yang panjangnya 64 bit, sedangkan panjang kuncinya 256 bit. Algoritma Blum Blum Shub merupakan pembangkit bilangan acak semu yang cukup mudah untuk dibangkitkan melalui persamaannya namun bilangan acak yang dihasilkan tidak mudah untuk diprediksi Luaran atau hasil dari penelitian ini adalah sebuah aplikasi login di sebuah website yang sistem enkripsi data user dan passwordnya menerapkan algoritma GOST yang pembangkitan kuncinya di modifikasi dengan Blum Blum Shub Generator.

Abstract

One method that can be used to secure the transmission of information is by combining information into codes that are not easily understood by unauthorized people. This technique is called cryptography. Analysis of the combination and modification of the GOST method was carried out using mathematical analysis and program analysis. The GOST algorithm is a symmetric cryptographic algorithm that operates on a message block size of 64 bits, while the key length is 256 bits. The Blum Blum Shub algorithm is a pseudo random number generator which is quite easy to generate through the equation, but the resulting random number is not easy to predict. The output or result of this research is a login application on a website whose user data and password encryption system applies the GOST algorithm which generates the key is modified with the Blum Blum Shub Generator.

1. PENDAHULUAN

Mengamankan data yang mengandung informasi penting salah satu teknik yang dapat digunakan yaitu dengan menyandikan informasi menjadi kode-kode yang tidak dengan mudah dimengerti oleh orang-orang yang tidak berkepentingan, teknik ini disebut kriptografi. Kriptografi bertujuan untuk memberikan layanan keamanan informasi (yang dinamakan juga sebagai aspek-aspek keamanan informasi), yaitu: Kerahasiaan (*confidentiality*), Integritas Data (*integrity*), Otentikasi (*authentication*), Nir penyangkalan (*non repudiation*), Merupakan layanan yang ditujukan untuk menjaga agar pesan tidak dapat dibaca oleh pihak-pihak yang tidak berhak.

Fenomena ketidakamannya data pada database terletak pada record database login yang kerap kali menjadi sasaran serangan saber saat ini, fenomena ini dituliskan oleh dblab dilamannya pertanggal 24 maret 2023 [1]. Jenis serangan yang kerap menyerang record database adalah jenis serangan SQL Injection. SQL Injection adalah salah satu jenis serangan yang mefokuskan pada penyerangan dengan cara mempengaruhi SQL command yang telah ditentukan.

Algoritma GOST adalah salah satu algoritma kriptografi simetris yang beroperasi pada ukuran blok pesan yang panjangnya 64 bit, sedangkan panjang kuncinya 256 bit. Jumlah putaran pada

GOST adalah 32 putaran, setiap putaran menggunakan kunci internal. Kunci GOSTmenggunakan 8 buah S-Box yang berbeda-beda, GOST juga menggunakan operasi XOR dan left circular shift 11bit atau rotate left shift dan menggunakan modulo 232. Dalam artikel Irfan anas menjelaskan Pemanfaatan algoritma GOST dalam pengamanan data berjenis teks sangat baik karena data asli akan disandikan dan akan menghasilkan cipher yang cukup acak serta rumit diketahui makna aslinya [2].

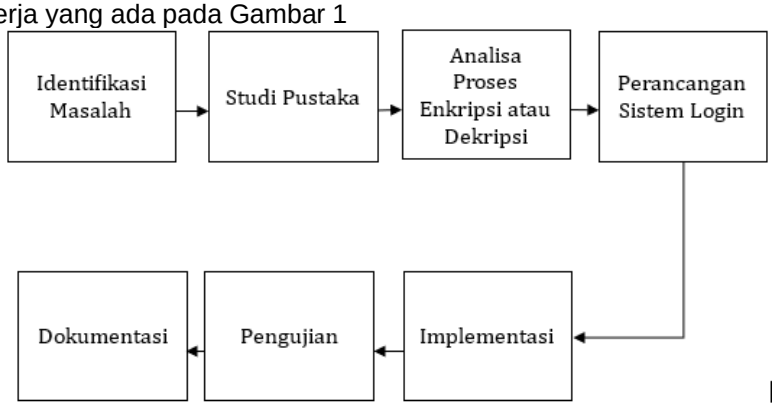
Algoritma Blum Blum Shub merupakan pembangkit bilangan acak semu yang cukup mudah untuk dibangkitkan melalui persamaannya namun bilangan acak yang dihasilkan tidak mudah untuk diprediksi [3]. Adapun analisis dalam penelitian ini dilakukan terhadap modifikasi algoritma GOST dengan menggunakan analisis matematis dan analisa program. Cryptographically Secure Pseudorandom Number Generator (CSPRNG) yang digunakan adalah algoritma blum blum shub[3]. CSPRNG ini dirancang berdasarkan teori bilangan. Algoritma ini terkenal karena kesederhanaannya dalam proses perhitungan namun tetap menghasilkan bilangan acak yang aman [4].

Pemilihan algoritma GOST dan Blum Blum Shub pada sistem pengamanan login website, penulis berlandaskan pada penelitian terdahulu yang telah menggunakan kedua algoritma ini dalam penelitiannya. Salah satu penelitian yang dilakukan oleh Titih fatimah dengan judul implementasi kriptografi menggunakan algoritma GOST untuk pengiriman e-mail pada aplikasi cirus-mail, dalam penelitiannya menyimpulkan bahwa proses implementasi GOST berhasil dilakukan terbukti pesan pada email dapat terenkrip dengan baik sehingga teks dapat dikirim tanpa mengalami kebocoran informasi kepada pihak yang tidak berkepentingan. Waktu untuk mengirim pesan berbanding lurus dengan ukuran teks dan ukuran file yang akan dienkripsi, semakin kecil ukuran teks dan filenya maka semakin cepat waktu pengirimannya [5].

Luaran atau hasil dari penelitian ini adalah sebuah aplikasi login di sebuah website yang sistem enkripsi data user dan passwordnya menerapkan algoritma GOST yang pembangkitan kuncinya di modifikasi dengan Blum Blum Shub Generator. Berdasarkan uraian latar belakang masalah di atas, maka berikut judul penelitian yang telah dibuat “Analisis dan Implementasi Modifikasi Algoritma Kriptografi GOST Menggunakan Blum Blum Shub Generator Pada Sistem Pengamanan Login Pada Website”.

2. METODE PENELITIAN

Metodologi penelitian menggambarkan kerangka penelitian atau yang biasa disebut dengan tahapan penelitian. Kerangka kerja terdiri dari beberapa langkah yang secara sistematis. Kerangka kerja ini diperlukan untuk memudahkan pelaksanaan penelitian yang dilakukan penulis dengan judul Analisis dan Implementasi Modifikasi Algoritma Kriptografi GOST Menggunakan Blum Blum Shub Generator Pada Sistem Pengamanan Login Pada Website. Tahapan yang akan dilakukan dapat dilihat dalam kerangka kerja yang ada pada Gambar 1



Gambar 1. Metode Penelitian

2.1 Pengertian Analisis

Kata analisis diadaptasi dari bahasa inggris "analysis" yang secara etimologi berasal dari bahasa Yunani kuno yang dibaca Analusis. Kata Analusis terdiri dari dua suku kata, yaitu "ana" yang artinya kembali, dan "luein" yang artinya melepas atau mengurai. Bila digabungkan maka kata tersebut memiliki arti menguraikan kembali. Berdasarkan pengertian tersebut maka dapat disimpulkan pengertian analisis adalah proses memecah topik atau substansi yang kompleks menjadi bagian-bagian yang lebih kecil untuk mendapatkan pemahaman yang lebih baik [6].

2.2 Pengertian Implementasi

Implementasi secara sederhana dapat diartikan sebagai pelaksanaan atau penerapan. Seperti yang dituliskan dalam kamus besar bahasa indonesia, kata implementasi berarti penerapan. Browne dan Wildavsky mengemukakan bahwa kata implementasi adalah perluasan aktivitas yang saling menyesuaikan yang bermuara pada aktivitas, adanya aksi, tindakan atau mekanisme suatu sistem tertentu [7].

2.3 Kriptografi

Kriptografi adalah ilmu untuk mempelajari penulisan secara rahasia dengan tujuan bahwa komunikasi dan data dapat dikodekan (*encode/encrypt*) dan dikodekan (*decode/decrypt*) kembali untuk mencegah pihak-pihak lain yang ingin mengetahui isinya. Kriptografi (*Cryptography*) berasal dari bahasa Yunani yaitu dari kata kryptos yang artinya tersembunyi. Kriptografi dapat diartikan sebagai tulisan yang dirahasiakan atau dapat diartikan juga sebagai suatu ilmu ataupun seni yang mempelajari bagaimana sebuah data, informasi dan dokumen dikonversi kebentuk tertentu yang sulit untuk dimengerti.

Proses yang dilakukan untuk mengubah plaintext menjadi *ciphertext* disebut enkripsi (*encryption*) atau encipherment sedangkan proses untuk mengubah *ciphertext* kembali ke plaintext disebut dekripsi (*decryption*) atau decipherment. Kriptografi memerlukan parameter untuk proses konversi yang dikendalikan oleh sebuah kunci atau beberapa kunci. Kriptografi saat ini telah menjadi salah satu syarat penting dalam keamanan teknologi informasi terutama dalam pengiriman pesan rahasia.

Algoritma kriptografi dibagi menjadi tiga bagian berdasarkan kunci yang dipakainya:

1. Algoritma Simetri (menggunakan satu kunci untuk enkripsi dan dekripsinya). Ini adalah jenis kriptografi yang paling umum dipergunakan. Kunci untuk membuatpesan yang disandikan sama dengan kunci untuk membuka pesan yang disandikan itu. Jadi pembuat pesan dan penerimanya harus memiliki kunci yang sama persis. Siapapun yang memiliki kunci tersebut, termasuk pihak-pihak yang tidak diinginkan, dapat membuat dan membongkar rahasia ciphertext. *Problem* yang paling jelas disini terkadang bukanlah masalah pengiriman ciphertext-nya, melainkan masalah bagaimana menyampaikan kunci simetris tersebut kepada pihak yang diinginkan.
2. Algoritma Asimetri (menggunakan kunci yang berbeda untuk enkripsi dan dekripsi). Algoritma asimetri sering juga disebut dengan algoritma kunci publik, dengan arti kata kunci yang digunakan untuk melakukan enkripsi dan dekripsi berbeda. Pada algoritma asimetri kunci terbagi menjadi dua bagian, yaitu:
   a Kunci umum (public key): kunci yang boleh diketahui semua orang(dipublikasikan).
   b Kunci rahasia (private key): kunci yang dirahasiakan (hanya boleh diketahui olehsatu orang saja). Kunci –kunci tersebut berhubungan satu sama lain. Dengan kunci public orang dapat mengenkripsi pesan tetapi tidak bisa mendekripsinya. Hanya orang yang memiliki kunci rahasia yang dapat mendekripsi pesan tersebut.
   c Hash Function: Fungsi HASH adalah fungsi yang menerima masukan string yang panjangnya sembarang dan mengonversinya menjadi string keluaran yang panjangnya tetap (fixed), yang umumnya berukuran jauh lebih kecil dari pada ukuran semula.

Beberapa macam penyerangan terhadap pesan yang sudah dienkripsi, berdasarkan ketersediaan data yang ada, dan tingkat kesulitannya bagi penyerang, dimulai dari yang paling sulit adalah :

1. *Ciphertext only attack*, penyerang hanya mendapatkan ciphertext dari sejumlah pesan yang seluruhnya telah dienkripsi menggunakan algoritma yang sama. Sehingga, metode yang digunakan untuk memecahkannya adalah exhaustive key search, yaitu mencoba semua kemungkinan yang ada untuk menemukan kunci.
2. *Known plaintext attack*, dimana penyerang selain mendapatkan sandi, juga mendapatkan pesan asli. Terkadang disebut pula clear-text attack.
3. *Choosen plaintext attack*, sama dengan known plaintext attack, namun penyerang bahkan dapat memilih penggalan mana dari pesan asli yang akan disandikan. Serangan jenis ini lebih hebat daripada known-plaintext attack, karena kriptoanalis dapat memilih plainteks tertentu untuk dienkripsikan, yaitu plainteks-plainteks yang lebih mengarahkan penemuan kunci.
4. *Chosen-ciphertext attack*, Pada tipe ini, kriptoanalis dapat memilih cipherteks yang berbeda untuk didekripsi dan memiliki akses atas plaintext yang didekripsi.
5. *Chosen-key attack*. Kriptoanalis pada tipe penyerangan ini memiliki pengetahuan tentang hubungan antara kunci-kunci yang berbeda dan memilih kunci yang tepat untuk mendekripsi pesan.

6. *Rubber-hose cryptanalysis*. Pada tipe penyerangan ini, kriptoanalis mengancam, menyiksa, memeras, memaksa, atau bahkan menyogok seseorang hingga mereka memberikan kuncinya. Ini adalah cara yang paling ampuh untuk mendapatkan kunci.
7. *Adaptive – chosen – plaintext attack*. Penyerangan tipe ini merupakan suatu kasus khusus chosen-plaintext attack. Kriptoanalis tidak hanya dapat memilih plainteks yang dienkripsi, ia pun memiliki kemampuan untuk memodifikasi pilihan berdasarkan hasil enkripsi sebelumnya. Dalam chosen-plaintext attack, kriptoanalis mungkin hanya dapat memiliki plainteks dalam suatu blok besar untuk dienkripsi; dalam adaptive-chosen-plaintext attack ini ia dapat memilih blok plainteks yang lebih kecil dan kemudian memilih yang lain berdasarkan hasil yang pertama, proses ini dapat dilakukannya terus menerus hingga ia dapat memperoleh seluruh informasi.

2.4 Algoritma GOST

Berikut beberapa langkah penerapan proses enkripsi dan dekripsi algoritma GOST:

1. Proses Pembangkitan Kunci
   Kunci internal pada algoritma GOST dibangkitkan dari kunci eksternal yang diberikan oleh pengguna[3]. Pembangkitan kunci internal dilakukan dengan membagi kunci eksternal 256 bit (k1, k2, k3, k4,…, k256) ke dalam delapan bagian yang masingmasing panjangnya 32 bit. Pembagiannya adalah sebagai berikut :
   K0 = (k32, ……. ,k1)
   K1 = (k64, ……. ,k33)
   K2 = (k96, ……. ,k65)
   K3 = (k128, ……. ,k97)
   K4 = (k160, ……. ,k129)
   K5 = (k192, ……. ,k161)
   K6 = (k224, ……. ,k193)
   K7 = (k256, ……. ,k225)
2. Proses Enkripsi
   Proses Enkripsi pada algoritma GOST untuk satu putaran (iterasi), adalah sebagai berikut :
   1) 64 bit plainteks dibagi menjadi 2 buah bagian 32 bit, yaitu Li dan Ri.
   Caranya :
   Input a1(0), a2(0), ….. ,a32(0) ; b1(0), b2(0),….. ,b32(0)
   R0 = a32(0), a31(0), ….. ,a1(0)
   L0 = b32(0), b31(0), ….. ,b1(0)
   2) $(R_i + K_i) \bmod 2^{32}$. Hasil dari penjumlahan modulo 2^{32} berupa 32 bit.
   3) Hasil dari penjumlahan modulo 2^{32} dibagi menjadi 8 bagian, dimana masing-masing bagian terdiri dari 4 bit. Setiap bagian dimasukkan ke dalam table S-box yang berbeda, 4 bit pertama menjadi input dari *S-box* 0, 4 bit kedua menjadi S-Box 1 dan seterusnya.
3. Proses Dekripsi
   Di dalam proses dekripsi terdapat aturan sama dengan proses enkripsi yaitu untuk langkah ke-5 dan ke-6 pada putaran ke-31 sebagai berikut :
   R32 = R31 sebelum dilakukan proses
   L32 = L31 XOR R31
   Sehingga, plainteks yang dihasilkan pada proses dekripsi adalah [4],
   L32 : b(32), b(31), ….. ,b(1)
   R32 : a(32), a(31), ….. ,a(1)
   Plainteks = a(1), ….. ,a(32); b(1), ….. ,b(32).

2.5 Blum Blum Generator

Blum-Blum Shub (BBS) merupakan suatu Pseudo Random Number Generator yang diajukan pada tahun 1986 oleh Lenore Blum, Manuel Blum dan Michael Shub. BBS memiliki bentuk persamaan: Xn+1 = Xn 2 mod m (1) dengan m merupakan hasil dari perkalian dua buah bilangan prima besar p dan q, serta output-nya dalam Least Significant Bit dari Xn dimana hal yang sama sebagai parity dari Xn. Dua buah bilangan prima p dan q harus kongruen terhadap 3 mod 4 dan Greatest Common Divisor (GCD) harus kecil. Generator ini sering digunakan untuk aplikasi kriptografi, karena generator ini tidak begitu cepat. Bagaiman pun juga, generator ini mempunyai bukti keamanan yang kuat, dimana berhubungan dengan kualitas generator karena sulitnya faktorisasi integer. Berikut langkahlangkah algoritma dari BBS [11]:

1. Pilih dua bilangan prima p dan q, di mana p dan q keduanya kongruen terhadap 3 modulo 4. $p \equiv 3 \bmod 4$ dan $q \equiv 3 \bmod 4$.
2. Hasilkan bilangan bulat Blum n dengan menghitung $n = p \times q$.

3. Pilih lagi sebuah bilangan acak s sebagai umpan, bilangan yang dipilih harus memenuhi kriteria: a. 2 ≤ s < n. b. s dan n adalah relatif prima.
4. Hitung nilai xo = s 2 mod n.
5. Hasilkan bilangan bit acak dengan cara : a. Hitung xi = x(i – 1) 2 mod n. b. Hasilkan zi = bit - bit yang diambil dari xi . Bit yang diambil bisa merupakan LSB (Least Significant Bit) atau hanya satu bit atau sebanyak j bit (j tidak melebihi log2 (log2 n)). Bilangan bit acak dapat digunakan langsung atau di-format dengan aturan tertentu, sedemikian hingga menjadi bilangan bulat.

3. HASIL DAN ANALISIS

Penerapan metode mencakup pada penerapan Algoritma Kriptografi GOST dengan proses pembentukan kunci Blum Blum Shub Generator pada sistem pengamanan data login pada website. Berikut adalah sampel data yang digunakan. Dari banyak *record* yang ada, maka diambil sampel untuk dilakukan proses enkripsi dan dekripsi dengan algoritma GOST yang telah dimodifikasi. Data yang diambil dari record database yaitu *field username* “SRI_HARTATI” dan *field password* “123#Ab”, maka nilai plaintext sama dengan SRI_HARTATI123#Ab.

Modifikasi GOST yang dilakukan penulis pada penelitian ini adalah mengganti proses pembentukan kunci GOST dengan menggunakan Blum Blum Shub Generator, sementara untuk Proses Enkripsi dan Dekripsi menggunakan algoritma GOST.

Algoritma penggunaan blum blum shub generator:

1. Pilih dua bilangan prima, dalam penerapan ini penulis menggunakan p = 11 dan q = 19.
2. Mencari nilai n, dengan melakukan perkalian n = p*q
   n = 11 * 19 = 209
3. Pilih bilangan bulat acak lain untuk nilai *s* antara nilai 2 sampai 209
   2 ≤ s ≤ n
   s = 5
4. Melakukan iterasi sesuai panjang yang diinginkan.
   Xi = 0 sampai dengan 209 (sesuai keinginan)

Berikut peroses pembangkitan kunci sesuai algoritma diatas dan dilakukan sepanjang kunci GOST sebanyak 256 bit dengan jumlah bilangan 32, berikut peroses perhitungan Blum Blum Shub Generator:

Tabel 1. Pembangkitan Angka Kunci dengan Blum Blum Shub

X	nilai s	s pangkat	mod	hasil	binner
1	5	2	209	25	00011001
2	25	2	209	207	11001111
3	207	2	209	4	00000100
4	4	2	209	16	00010000
5	16	2	209	47	00101111
6	47	2	209	119	01110111
7	119	2	209	158	10011110
8	158	2	209	93	01011101
9	93	2	209	80	01010000
10	80	2	209	130	10000010
11	130	2	209	180	10110100
12	180	2	209	5	00000101
13	5	2	209	25	00011001
14	25	2	209	207	11001111
15	207	2	209	4	00000100
16	4	2	209	16	00010000
17	16	2	209	47	00101111
18	47	2	209	119	01110111
19	119	2	209	158	10011110
20	158	2	209	93	01011101

21	93	2	209	80	01010000
22	80	2	209	130	10000010
23	130	2	209	180	10110100
24	180	2	209	5	00000101
25	5	2	209	25	00011001
26	25	2	209	207	11001111
27	207	2	209	4	00000100
28	4	2	209	16	00010000
29	16	2	209	47	00101111
30	47	2	209	119	01110111
31	119	2	209	158	10011110
32	158	2	209	93	01011101

Maka berdasarkan perhitungan diatas, berikut biner kunci untuk proses enkripsi dengan algoritma GOST, sebagai berikut:
00011001 11001111 00000100 00010000 00101111 01110111 10011110 01011101 01010000 10000010 10110100 00000101 00011001 11001111 00000100 00010000 00101111 01110111 10011110 01011101 01010000 10000010 10110100 00000101 00011001 11001111 00000100 00010000 00101111 01110111 10011110 01011101

2.1 Penerapan Algoritma GOST

Penerapan algoritma GOST dilakukan setelah proses generator kunci dengan menggunakan Blum Blum Shub dilakukan. Proses penerapan algoritma GOST, sebagai berikut:

1. Pengelompokan Kunci GOST, yang nilai binernya diambil dari hasil penerapan Blum Bum Shub

Tabel 2. Pengelompokan Biner Kunci

Kunci	Posisi Bit	Pengambilan Biner	Bilangan Desimal
K[0]=	32...1	00001000 00100000 11110011 10011000	136377240
K[1]=	64...33	10111010 01111001 11101110 11110100	3128553204
K[2]=	96...65	10100000 00101101 01000001 00001010	2687320330
K[3]=	128...97	00001000 00100000 11110011 10011000	136377240
K[4]=	160...129	10111010 01111001 11101110 11110100	3128553204
K[5]=	192...161	10100000 00101101 01000001 00001010	2687320330
K[6]=	224...193	00001000 00100000 11110011 10011000	136377240
K[7]=	256...225	10111010 01111001 11101110 11110100	3128553204

2. Proses Enkripsi GOST, terhadap *plaintext* [SRI_HARTATI123#Ab]

Tabel 3. Konversi *Plaintext* ke Biner

Karakter	Decimal	Biner
S	83	01010011
R	82	01010010
I	73	01001001
_	95	01011111
H	72	01001000
A	65	01000001
R	82	01010010
T	84	01010100
A	65	01000001
T	84	01010100
I	73	01001001
1	49	00110001
2	50	00110010
3	51	00110011
#	35	00100011
A	65	01000001
b	98	01100010

Berdasarkan tabel diatas proses enkripsi pertama dilakukan terhadap plainteks berikut:

Tabel 4. Plainteks Pertama

Karakter	Decimal	Biner
S	83	01010011
R	82	01010010
I	73	01001001
_	95	01011111
H	72	01001000
A	65	01000001
R	82	01010010
T	84	01010100

maka selanjutnya menggabungkan seluruh biner sesuai dengan ketentuan L[0] dan R[0] pada algoritma GOST
R = 01010011 01010010 01001001 01011111
L = 01001000 01000001 01010010 01010100

Enkripsi Putaran i = 0 :

Tahap 1 R[0] = 11111010 10010010 01001010 11001010 = 4203891402
L[0] = 00101010 01001010 10000010 00010010 = 709526034

Tahap 2 (R[0] + K[0]) Mod 2^{32}
4203891402 + 136377240 = 4340268642 Mod 2^{32} = 45301346
Nilai Biner:
00000010 10110011 00111110 01100010

Tahap 3 Melakukan permutasi terhadap S-BOX GOST

Biner Kelompok	Dec Nilai Bit	S-BOX	Hasil Permutasi S-BOX	Biner
0000	0	S-BOX(0)	4	0100
0010	2	S-BOX(1)	4	0100
1011	11	S-BOX(2)	7	0111
0011	3	S-BOX(3)	1	0001
0011	3	S-BOX(4)	1	0001
1110	14	S-BOX(5)	15	1111
0110	6	S-BOX(6)	5	0101
0010	2	S-BOX(7)	2	0010

Tahap 4 Gabung seluruh nilai biner hasi permutasi terhadap S-BOX, selanjutnya lakukan *Rotasi Left Shift* (RLF) 11 bit
Nilai biner hasil gabungan:
01000100 01110001 00011111 01010010
Hasil RLS [0] 11 bit:
10001000 11111010 10010010 00100011

Tahap 5 R[1] = RLS [0] XOR L[0]
RLS[0] = 10001000111110101001001000100011
L[0] = 00101010010010101000001000010010 ⊕
R[1] = 10100010 10110000 00010000 00110001

Enkripsi Putaran i = 1 : nilai L[1] diambill dari nilai R[0]

Tahap 1 R[1] = 10100010 10110000 00010000 00110001 = 2729447473
L[1] = 11111010 10010010 01001010 11001010 = 4203891402

Tahap 2 (R[1] + K[1]) Mod 2^{32}
2729447473 + 3128553204 = 5858000677 Mod 2^{32} = 1563033381
Nilai Biner:
01011101 00101001 11111111 00100101

Tahap 3 Melakukan permutasi terhadap S-BOX GOST

Biner Kelompok	Dec Nilai Bit	S-BOX	Hasil Permutasi S-BOX	Biner

0101	5	S-BOX(0)	8	1000
1101	13	S-BOX(1)	7	0111
0010	2	S-BOX(2)	1	0001
1001	9	S-BOX(3)	4	0100
1111	15	S-BOX(4)	2	0010
1111	15	S-BOX(5)	14	1110
0010	2	S-BOX(6)	4	0100
0101	5	S-BOX(7)	7	0111

Tahap 4 Gabung seluruh nilai biner hasi permutasi terhadap S-BOX, selanjutnya lakukan *Rotasi Left Shift* (RLF) 11 bit
Nilai biner hasil gabungan:
10000111000101000010111001000111
Hasil RLS [1] 11 bit:
10100001011100100011110000111000

Tahap 5 R[2] = RLS [1] XOR L[1]
RLS[1] = 10100001011100100011110000111000
L[1] = 11111010100100100100101011001010 ⊕
R[2] = 01011011111000000111011011110010 = 1541437170

Enkripsi Putaran i = 2 : nilai L[2] diambill dari nilai R[1]

Tahap 1 R[2] = 01011011111000000111011011110010 = 1541437170
L[2] = 10100010 10110000 00010000 00110001 = 2729447473

Tahap 2 $(R[2] + K[2]) \text{ Mod } 2^{32}$
$1541437170 + 2687320330 = 4228757500 \text{ Mod } 2^{32} = 4228757500$
Nilai Biner:
11111100 00001101 10110111 11111100

Tahap 3 Melakukan permutasi terhadap S-BOX GOST

Biner Kelompok	Dec Nilai Bit	S-BOX	Hasil Permutasi S-BOX	Biner
1111	15	S-BOX(0)	3	0011
1100	12	S-BOX(1)	0	0000
0000	0	S-BOX(2)	5	0101
1101	13	S-BOX(3)	11	1011
1011	11	S-BOX(4)	14	1110
0111	7	S-BOX(5)	13	1101
1111	15	S-BOX(6)	12	1100
1100	12	S-BOX(7)	6	0110

Tahap 4 Gabung seluruh nilai biner hasi permutasi terhadap S-BOX, selanjutnya lakukan *Rotasi Left Shift* (RLF) 11 bit
Nilai biner hasil gabungan:
00110000010 11011111010110111000110
Hasil RLS [2] 11 bit:
11011111010110111000110 00110000010

Tahap 5 R[3] = RLS [2] XOR L[2]
RLS[2] = 11011111011011100011000110000010
L[2] = 10100010101100000001000000110001 ⊕
R[3] = 01111101110111100010000110110011

Enkripsi Putaran i = 3 : nilai L[3] diambill dari nilai R[2]

Tahap 1 R[3] = 01111101110111100010000110110011 = 2111709619
L[3] = 01011011111000000111011011110010 = 1541437170

Tahap 2 $(R[3] + K[3]) \text{ Mod } 2^{32}$
$2111709619 + 136377240 = 2248086859 \text{ Mod } 2^{32} = 2248086859$
Nilai Biner:
10000101111111110001010101001011

Tahap 3 Melakukan permutasi terhadap S-BOX GOST

Biner Kelompok	Dec Nilai Bit	S-BOX	Hasil Permutasi S-BOX	Biner
1000	8	S-BOX(0)	6	0110
0101	5	S-BOX(1)	13	1101
1111	15	S-BOX(2)	11	1011
1111	15	S-BOX(3)	3	0011
0001	1	S-BOX(4)	12	1100
0101	5	S-BOX(5)	2	0010
0100	4	S-BOX(6)	3	0011
1011	11	S-BOX(7)	14	1110

Tahap 4 Gabung seluruh nilai biner hasi permutasi terhadap S-BOX, selanjutnya lakukan *Rotasi Left Shift* (RLF) 11 bit
Nilai biner hasil gabungan:
01101101101100111100001000111110
Hasil RLS [3] 11 bit:
10011110000100011111001101101101

Tahap 5 R[4] = RLS [3] XOR L[3]
RLS[3] = 10011110000100011111001101101101
L[3] = 01011011111000000111011011110010 $\oplus$
R[4] = 11000101111100011000010110011111 = 3320939935

Enkripsi Putaran i = 4 : nilai L[4] diambill dari nilai R[3]

Tahap 1 R[4] = 11000101111100011000010110011111 = 3320939935
L[4] = 01111101110111100010000110110011 = 2111709619

Tahap 2 (R[4] + K[4]) Mod 2^{32}
3320939935 + 3128553204 = 6449493139 Mod 2^{32} = 2154525843
Nilai Biner:
10000000011010110111010010010011

Tahap 3 Melakukan permutasi terhadap S-BOX GOST

Biner Kelompok	Dec Nilai Bit	S-BOX	Hasil Permutasi S-BOX	Biner
1000	8	S-BOX(0)	6	0110
0000	0	S-BOX(1)	14	1110
0110	6	S-BOX(2)	4	0100
1011	11	S-BOX(3)	12	1100
0111	7	S-BOX(4)	8	1000
0100	4	S-BOX(5)	7	0111
1001	9	S-BOX(6)	10	1010
0011	3	S-BOX(7)	0	0000

Tahap 4 Gabung seluruh nilai biner hasi permutasi terhadap S-BOX, selanjutnya lakukan *Rotasi Left Shift* (RLF) 11 bit
Nilai biner hasil gabungan:
01101110010011001000011110100000
Hasil RLS [4] 11 bit:
01100100001111010000001101110010

Tahap 5 R[5] = RLS [4] XOR L[4]
RLS[4] = 01100100001111010000001101110010
L[4] = 01111101110111100010000110110011 $\oplus$
R[5] = 00011001111000110010001011000001 = 434315969

Enkripsi Putaran i = 5 : nilai L[5] diambill dari nilai R[4]

Tahap 1 R[5] = 00011001111000110010001011000001 = 434315969
L[5] = 11000101111100011000010110011111 = 3320939935

Tahap 2 (R[5] + K[5]) Mod 2^{32}
434315969 + 2687320330 = 3121636299 Mod 2^{32} = 3121636299
Nilai Biner:

10111010000100000110001111001011

Tahap 3 Melakukan permutasi terhadap S-BOX GOST

Biner Kelompok	Dec Nilai Bit	S-BOX	Hasil Permutasi S-BOX	Biner
1011	11	S-BOX(0)	12	1100
1010	10	S-BOX(1)	8	1000
0001	1	S-BOX(2)	8	1000
0000	0	S-BOX(3)	7	0111
0110	6	S-BOX(4)	13	1101
0011	3	S-BOX(5)	0	0000
1100	12	S-BOX(6)	6	0110
1011	11	S-BOX(7)	14	1110

Tahap 4 Gabung seluruh nilai biner hasi permutasi terhadap S-BOX, selanjutnya lakukan *Rotasi Left Shift* (RLF) 11 bit
Nilai biner hasil gabungan:
11001000100001111101000001101110
Hasil RLS [5] 11 bit:
00111110100000110111011001000100

Tahap 5 R[6] = RLS [5] XOR L[5]
RLS[5] = 00111110100000110111011001000100
L[5] = 11000101111100011000010110011111 ⊕
R[6] = 11111011011100101111001111011011 = 4218614747

Enkripsi Putaran i = 6 : nilai L[6] diambill dari nilai R[5]

Tahap 1 R[6] = 11111011011100101111001111011011 = 4218614747
L[6] = 00011001111000110010001011000001 = 434315969

Tahap 2 (R[6] + K[6]) Mod 2^{32}
4218614747 + 136377240 = 4354991987 Mod 2^{32} = 60024691
Nilai Biner:
00000011100100111110011101110011

Tahap 3 Melakukan permutasi terhadap S-BOX GOST

Biner Kelompok	Dec Nilai Bit	S-BOX	Hasil Permutasi S-BOX	Biner
0000	0	S-BOX(0)	4	0100
0011	3	S-BOX(1)	12	1100
1001	9	S-BOX(2)	15	1111
0011	3	S-BOX(3)	1	0001
1110	14	S-BOX(4)	11	1011
0111	7	S-BOX(5)	13	1101
0111	7	S-BOX(6)	9	1001
0011	3	S-BOX(7)	0	0000

Tahap 4 Gabung seluruh nilai biner hasi permutasi terhadap S-BOX, selanjutnya lakukan *Rotasi Left Shift* (RLF) 11 bit
Nilai biner hasil gabungan:
01001100111100011011110110010000
Hasil RLS [6] 11 bit:
10001101111011001000001001100111

Tahap 5 R[7] = RLS [6] XOR L[6]
RLS[6] = 10001101111011001000001001100111
L[6] = 00011001111000110010001011000001 ⊕
R[7] = 10010100000011111010000010100110 = 2484052134

Enkripsi Putaran i = 7 : nilai L[7] diambill dari nilai R[6]

Tahap 1 R[7] = 10010100000011111010000010100110 = 2484052134
L[7] = 11111011011100101111001111011011 = 4218614747

Tahap 2 $(R[7] + K[7]) \text{ Mod } 2^{32}$
2484052134 + 3128553204 = 5612605338 Mod 2^{32} = 1317638042
Nilai Biner:
01001110100010011000111110011010

Tahap 3 Melakukan permutasi terhadap S-BOX GOST

Biner Kelompok	Dec Nilai Bit	S-BOX	Hasil Permutasi S-BOX	Biner
0100	4	S-BOX(0)	13	1101
1110	14	S-BOX(1)	5	0101
1000	8	S-BOX(2)	14	1110
1001	9	S-BOX(3)	4	0100
1000	8	S-BOX(4)	4	0100
1111	15	S-BOX(5)	14	1110
1001	9	S-BOX(6)	10	1010
1010	10	S-BOX(7)	3	0011

Tahap 4 Gabung seluruh nilai biner hasi permutasi terhadap S-BOX, selanjutnya lakukan *Rotasi Left Shift* (RLF) 11 bit
Nilai biner hasil gabungan:
11010101111001000100111010100011
Hasil RLS [7] 11 bit:
00100010011101010001111010101111

Tahap 5 R[8] = RLS [7] XOR L[7]
RLS[7] = 00100010011101010001111010101111
L[7] = 11111011011100101111001111011011 $\oplus$
R[8] = 11011001000001111111011010111010 0 = 3641175412

Enkripsi Putaran i = 8 : nilai L[8] diambill dari nilai R[7]

Tahap 1 R[8] = 11011001000001111111011010111010 0 = 3641175412
L[8] = 10010100000011111010000010100110 = 2484052134

Tahap 2 $(R[8] + K[0]) \text{ Mod } 2^{32}$
3641175412 + 136377240 = 3777552652 Mod 2^{32} = 3777552652
Nilai Biner:
11100001001010001110000100001100

Tahap 3 Melakukan permutasi terhadap S-BOX GOST

Biner Kelompok	Dec Nilai Bit	S-BOX	Hasil Permutasi S-BOX	Biner
1110	14	S-BOX(0)	5	0101
0001	1	S-BOX(1)	11	1011
0010	2	S-BOX(2)	1	0001
1000	8	S-BOX(3)	14	1110
1110	14	S-BOX(4)	11	1011
0001	1	S-BOX(5)	11	1011
0000	0	S-BOX(6)	13	1101
1100	12	S-BOX(7)	6	0110

Tahap 4 Gabung seluruh nilai biner hasi permutasi terhadap S-BOX, selanjutnya lakukan *Rotasi Left Shift* (RLF) 11 bit
Nilai biner hasil gabungan:
01011011000111101011101111010110
Hasil RLS [8] 11 bit:
11110101110111101011001011011000

Tahap 5 R[9] = RLS [8] XOR L[8]
RLS[8] = 11110101110111101011001011011000
L[8] = 10010100000011111010000010100110 $\oplus$
R[9] = 01100001110100010001001001111110 = 1641091710

Enkripsi Putaran i = 9 : nilai L[9] diambill dari nilai R[8]

Tahap 1 R[9] = 01100001110100010001001001111110 = 1641091710
L[9] = 11011001000001111111010101110100 = 3641175412

Tahap 2 (R[9] + K[1]) Mod 2^{32}
1641091710 + 3128553204 = 4769644914 Mod 2^{32} = 474677618
Nilai Biner:
00011100010010110000000101110010

Tahap 3 Melakukan permutasi terhadap S-BOX GOST

Biner Kelompok	Dec Nilai Bit	S-BOX	Hasil Permutasi S-BOX	Biner
0001	1	S-BOX(0)	10	1010
1100	12	S-BOX(1)	0	0000
0100	4	S-BOX(2)	10	1010
1011	11	S-BOX(3)	12	1100
0000	0	S-BOX(4)	6	0110
0001	1	S-BOX(5)	11	1011
0111	7	S-BOX(6)	9	1001
0010	2	S-BOX(7)	13	1101

Tahap 4 Gabung seluruh nilai biner hasi permutasi terhadap S-BOX, selanjutnya lakukan *Rotasi Left Shift* (RLF) 11 bit
Nilai biner hasil gabungan:
10100000101011000110101110011101
Hasil RLS [9] 11 bit:
01100011010111001110110100000101

Tahap 5 R[10] = RLS [9] XOR L[9]
RLS[9] = 01100011010111001110110100000101
L[9] = 11011001000001111111010101110100 ⊕
R[10] = 10111010010110110000000001110001 = 3126526065

Enkripsi Putaran i = 10 : nilai L[10] diambill dari nilai R[9]

Tahap 1 R[10] = 10111010010110110000000001110001 = 3126526065
L[10] = 01100001110100010001001001111110 = 1641091710

Tahap 2 (R[10] + K[2]) Mod 2^{32}
3126526065 + 2687320330 = 5813846395 Mod 2^{32} = 1518879099
Nilai Biner:
01011010100010000100000101111011

Tahap 3 Melakukan permutasi terhadap S-BOX GOST

Biner Kelompok	Dec Nilai Bit	S-BOX	Hasil Permutasi S-BOX	Biner
0101	5	S-BOX(0)	8	1000
1010	10	S-BOX(1)	8	1000
1000	8	S-BOX(2)	14	1110
1000	8	S-BOX(3)	14	1110
0100	4	S-BOX(4)	5	0101
0001	1	S-BOX(5)	11	1011
0111	7	S-BOX(6)	9	1001
1011	11	S-BOX(7)	14	1110

Tahap 4 Gabung seluruh nilai biner hasi permutasi terhadap S-BOX, selanjutnya lakukan *Rotasi Left Shift* (RLF) 11 bit
Nilai biner hasil gabungan:
10001000111011100101101110011110
Hasil RLS [10] 11 bit:
01110010110111001111010001000111

Tahap 5 R[11] = RLS [10] XOR L[10]
RLS[10] = 01110010110111001111010001000111

L[10] = 01100001110100010001001001111110 ⊕
R[11] = 00010011000011011110011000111001 = 319678009

Enkripsi Putaran i = 11 : nilai L[11] diambill dari nilai R[10]

Tahap 1 R[11] = 00010011000011011110011000111001 = 319678009
L[11] = 10111010010110110000000001110001 = 3126526065

Tahap 2 (R[11] + K[3]) Mod 2^{32}
319678009 + 136377240 = 456055249 Mod 2^{32} = 456055249
Nilai Biner:
00011011001011101101100111010001

Tahap 3 Melakukan permutasi terhadap S-BOX GOST

Biner Kelompok	Dec Nilai Bit	S-BOX	Hasil Permutasi S-BOX	Biner
0001	1	S-BOX(0)	10	1010
1011	11	S-BOX(1)	1	0001
0010	2	S-BOX(2)	1	0001
1110	14	S-BOX(3)	5	0101
1101	13	S-BOX(4)	3	0011
1001	9	S-BOX(5)	6	0110
1101	13	S-BOX(6)	8	1000
0001	1	S-BOX(7)	15	1111

Tahap 4 Gabung seluruh nilai biner hasi permutasi terhadap S-BOX, selanjutnya lakukan *Rotasi Left Shift* (RLF) 11 bit
Nilai biner hasil gabungan:
10100001000101010011011010001111
Hasil RLS [11] 11 bit:
10101001101101000111110100001000

Tahap 5 R[12] = RLS [11] XOR L[11]
RLS[11] = 10101001101101000111110100001000
L[11] = 10111010010110110000000001110001 ⊕
R[12] = 00010011111011110111110101111001 = 334462329

Enkripsi Putaran i = 12 : nilai L[12] diambill dari nilai R[11]

Tahap 1 R[12] = 00010011111011110111110101111001 = 334462329
L[12] = 00010011000011011110011000111001 = 319678009

Tahap 2 (R[12] + K[4]) Mod 2^{32}
334462329 + 3128553204 = 3463015533 Mod 2^{32} = 3463015533
Nilai Biner:
11001110011010010110110001101101

Tahap 3 Melakukan permutasi terhadap S-BOX GOST

Biner Kelompok	Dec Nilai Bit	S-BOX	Hasil Permutasi S-BOX	Biner
1100	12	S-BOX(0)	7	0111
1110	14	S-BOX(1)	5	0101
0110	6	S-BOX(2)	5	0101
1001	9	S-BOX(3)	4	0100
0110	6	S-BOX(4)	13	1101
1100	12	S-BOX(5)	9	1001
0110	6	S-BOX(6)	5	0101
1101	13	S-BOX(7)	11	1011

Tahap 4 Gabung seluruh nilai biner hasi permutasi terhadap S-BOX, selanjutnya lakukan *Rotasi Left Shift* (RLF) 11 bit
Nilai biner hasil gabungan:
01110101010101001101100101011011
Hasil RLS [12] 11 bit:
10100110110010101101101110101010

Tahap 5 R[13] = RLS [12] XOR L[12]
RLS[12] = 10100110110010101101101110101010
L[12] = 00010011000011011110011000111001 ⊕
R[13] = 10110101110001110011110110010011 = 3049733523

Enkripsi Putaran i = 13 : nilai L[13] diambill dari nilai R[12]

Tahap 1 R[13] = 10110101110001110011110110010011 = 3049733523
L[13] = 00010011111011110111110101111001 = 334462329

Tahap 2 (R[13] + K[5]) Mod 2^{32}
3049733523 + 2687320330 = 5737053853 Mod 2^{32} = 1442086557
Nilai Biner:
01010101111101000111111010011101

Tahap 3 Melakukan permutasi terhadap S-BOX GOST

Biner Kelompok	Dec Nilai Bit	S-BOX	Hasil Permutasi S-BOX	Biner
0101	5	S-BOX(0)	8	1000
0101	5	S-BOX(1)	13	1101
1111	15	S-BOX(2)	11	1011
0100	4	S-BOX(3)	0	0000
0111	7	S-BOX(4)	8	1000
1110	14	S-BOX(5)	15	1111
1001	9	S-BOX(6)	10	1010
1101	13	S-BOX(7)	11	1011

Tahap 4 Gabung seluruh nilai biner hasi permutasi terhadap S-BOX, selanjutnya lakukan *Rotasi Left Shift* (RLF) 11 bit
Nilai biner hasil gabungan:
10001101101100001000111110101011
Hasil RLS [13] 11 bit:
10000100011111010101110001101101

Tahap 5 R[14] = RLS [13] XOR L[13]
RLS[13] = 10000100011111010101110001101101
L[13] = 00010011111011110111110101111001 ⊕
R[14] = 10010111100100100010000100010100 = 2542936340

Enkripsi Putaran i = 14 : nilai L[14] diambill dari nilai R[13]

Tahap 1 R[14] = 10010111100100100010000100010100 = 2542936340
L[14] = 10110101110001110011110110010011 = 3049733523

Tahap 2 (R[14] + K[6]) Mod 2^{32}
2542936340 + 136377240 = 2679313580 Mod 2^{32} = 2679313580
Nilai Biner:
10011111101100110001010010101100

Tahap 3 Melakukan permutasi terhadap S-BOX GOST

Biner Kelompok	Dec Nilai Bit	S-BOX	Hasil Permutasi S-BOX	Biner
1001	9	S-BOX(0)	11	1011
1111	15	S-BOX(1)	9	1001
1011	11	S-BOX(2)	7	0111
0011	3	S-BOX(3)	1	0001
0001	1	S-BOX(4)	12	1100
0100	4	S-BOX(5)	7	0111
1010	10	S-BOX(6)	14	1110
1100	12	S-BOX(7)	6	0110

Tahap 4 Gabung seluruh nilai biner hasi permutasi terhadap S-BOX, selanjutnya lakukan *Rotasi Left Shift* (RLF) 11 bit
Nilai biner hasil gabungan:
10111001011100011100011111100110
Hasil RLS [14] 11 bit:

10001110001111110011010111001011

Tahap 5 R[15] = RLS [14] XOR L[14]
RLS[14] = 10001110001111110011010111001011
L[14] = 10110101110001110011110110010011 ⊕
R[15] = 00111011111110000000100001011000 = 1006110808

Enkripsi Putaran i = 15 : nilai L[15] diambill dari nilai R[14]

Tahap 1 R[15] = 00111011111110000000100001011000 = 1006110808
L[15] = 10010111100100100010000100010100 = 2542936340

Tahap 2 (R[15] + K[7]) Mod 2^{32}
1006110808 + 3128553204 = 4134664012 Mod 2^{32} = 4134664012
Nilai Biner:
11110110011100011111011101001100

Tahap 3 Melakukan permutasi terhadap S-BOX GOST

Biner Kelompok	Dec Nilai Bit	S-BOX	Hasil Permutasi S-BOX	Biner
1111	15	S-BOX(0)	3	0011
0110	6	S-BOX(1)	15	1111
0111	7	S-BOX(2)	2	0010
0001	1	S-BOX(3)	13	1101
1111	15	S-BOX(4)	2	0010
0111	7	S-BOX(5)	13	1101
0100	4	S-BOX(6)	3	0011
1100	12	S-BOX(7)	6	0110

Tahap 4 Gabung seluruh nilai biner hasi permutasi terhadap S-BOX, selanjutnya lakukan *Rotasi Left Shift* (RLF) 11 bit
Nilai biner hasil gabungan:
00111111001011010010110100110110
Hasil RLS [15] 11 bit:
01101001011010011011000111111001

Tahap 5 R[16] = RLS [15] XOR L[15]
RLS[15] = 01101001011010011011000111111001
L[15] = 10010111100100100010000100010100 ⊕
R[16] = 11111110111110111001000011101101 = 4277899501

Dari proses hasil enkrispi algoritma GOST maka dihasilkan nilai *ciphertext* dengan,

R[32] :
10100110101100001010010000100111000010001100011011100011100101000100111111110000010011111100000011100001010111000010111001110110000111001010001001111111100000100111111000001110000101011100001011100111010110001001000111100001101000

L[32] :
00010010001111000011010000100110011011011010101001110001101110001111000111010011010010011101110000111000011010101101111000011111

Nilai gabungan biner:
10100110101100001010010000100111000010001100011011100011100101000100111111110000010011111100000011100001010111000010111001110101100010010001111000011010000100110011011011010101001110001101110001111000111001010010100111011100000111000011010101101111000011111

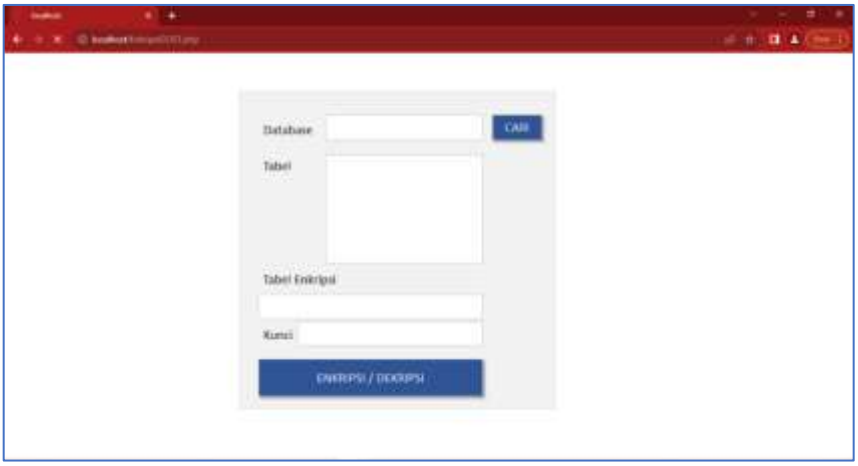
Nilai *hexadecimal*:
53585213846371ca27f827c1c2b85ceb123c34266daa71b8f1ca53b8386ade1f

Setelah nilai *hexadecial* dikonversi ke karakter menghasilkan bentuk karakter sebagai berikut: **SXR_ cqÊ'ø'ÁÂ¸\ë_<4&mªq¸ñÊS¸8jÞ**

3.1 Pengujian dan Hasil

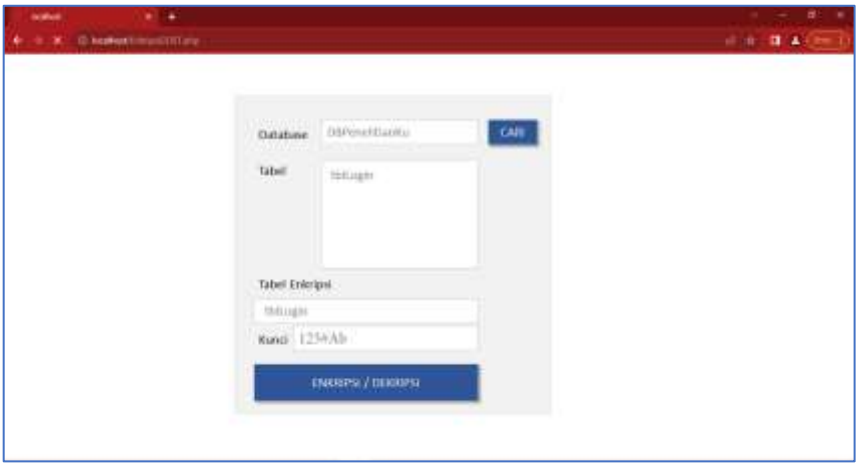
Pengujian dilakukan pada proses penyandian record data pada tabel tblLogin yang ada di database dengan nama database DBPenelitinanKu. Lebih jelasnya dapat dilihat dari hasil tangkap layar pada gambar 2, dan gambar 3.

Gambar 2. Pengujian Halaman Login



Gambar 3. Pengujian Halaman Enkripsi

Selanjutnya adalah proses enkrispi dengan memasukan nama database dan memilih tabel yang akan di enkripsi seperti pada gambar 4



Gambar 4. Pengujian Halaman Enkripsi untuk Proses Enkripsi

4. KESIMPULAN

Kesimpulan dari penelitian dengan analisis dan implementasi modifikasi algoritma kriptografi GOST menggunakan Blum Blum Shub Generator pada sistem pengamanan login pada website, sebagai berikut, Memodifikasi algoritma GOST dalam pengamanan data login website berhasil dilakukan dimana proses pembangkitan kunci menggunakan metode blum blum generator dan menghasilkan panjang biner kunci 32 bit perbloknya sebanyak 0 sampai dengan 256 posisi bit. Penerapan Blum Blum Shub generator sebagai metode pembangkitan kunci terhadap algoritma GOST

memiliki panjang bit perbloknya 32 bit dan dapat dilakukan pengelompokan kunci hingga panjang 256 bit mengikuti panjang blok algoritma enkrispi dan dekripsi GOST dan berhasil dilakukan. Pengujian terhadap Algoritma Kriptografi GOST yang telah dimodifikasi dalam pengamanan login suatu website berhasil dilakukan dalam sebuah sistem yang telah dibangun menggunakan pemograman web, data login pada database langsung terenkripsi sehingga tidak dapat dipahami apa isi data username dan password pengguna yang tersimpan di tabel login yang ada didatabase.

REFERENSI

[1] R. Silaban, "Penerapan Metode Fibonacci Code Dalam Mengkompresi File Video," *KOMIK (Konferensi Nasional Teknologi Informasi dan Komputer)*, vol. 6, no. 1, pp. 208–215, 2023.

[2] M. Syahrial, "Penerapan Algoritma Inverted Elias Delta Pada Aplikasi Penyimpanan File Terkompresi," *Pelita Informatika: Informasi dan Informatika*, vol. 9, no. 4, pp. 281–288, 2021.

[3] N. N. Diarse and R. Bendi, "Penerapan Algoritma RSA pada Sistem Kriptografi File Audio MP3," *Jurnal Hoaq Teknologi Informasi*, vol. 7, no. 2, pp. 567–575, 2016.

[4] D. Anggraini, "Penerapan Algoritma Fibonacci Code Pada File Transfer Berbasis Android," *Journal of Informatics Management and Information Technology*, vol. 1, no. 3, pp. 91–94, 2021.

[5] R. Handayani, "Perancangan Aplikasi Kompresi File Audio Menerapkan Algoritma Universal Codes," *KOMIK (Konferensi Nasional Teknologi Informasi dan Komputer)*, vol. 5, no. 1, 2021.

[6] J. M. B. Panjaitan, "Penerapan Algoritma Fibonacci Codes Pada Kompresi Aplikasi Audio Mp3 Berbasis Dekstop," *Bulletin of Multi-Disciplinary Science and Applied Technology*, vol. 1, no. 1, pp. 27–33, 2021.

[7] K. M. R. Hutahaean, "Penerapan Algoritma Inverted Elias Delta Untuk Kompresi Konten Pada Aplikasi Psikologi Berbasis Android," *Jurnal Riset Teknik Informatika dan Data Sains*, vol. 1, no. 1, pp. 26–37, 2022.

[8] L. Y. Telaumbanua, E. Bu'ulolo, and K. Ulfa, "Implementasi Algoritma Code-Excited Linear Prediction (Celp) Pada Kompresi File Audio," *KOMIK (Konferensi Nasional Teknologi Informasi dan Komputer)*, vol. 6, no. 1, pp. 317–321, 2023.

[9] S. D. Nasution, "Perancangan Aplikasi Kompresi File Teks Dengan Menerapkan Algoritma Goldbach Codes," *J. Ilm. INFOTEK*, vol. 1, no. 1, pp. 104–109, 2013.

[10] A. Tanjung, "Perbandingan Kinerja Algoritma Elias Delta Code Dan Algoritma Punctured Elias Code Dalam Kompresi File Teks," *KOMIK (Konferensi Nasional Teknologi Informasi dan Komputer)*, vol. 6, no. 1, pp. 182–190, 2023.

[11] M. A. Abdullah and R. T. Aldisa, "Perbandingan Metode Preference Selection Index dan Kombinasi Preference Selection Index dan TOPSIS dalam Penilaian Kinerja Karyawan Hotel," *KLIK: Kajian Ilmiah Informatika dan Komputer*, vol. 3, no. 6, pp. 1080–1087, 2023.

[12] M. A. Abdullah and R. T. Aldisa, "Perbandingan Metode Preference Selection Index dan Kombinasi Preference Selection Index dan TOPSIS dalam Penilaian Kinerja Karyawan Hotel," *KLIK: Kajian Ilmiah Informatika dan Komputer*, vol. 3, no. 6, pp. 1080–1087, 2023.

[13] A. Purnamawati, M. N. Winarto, and D. U. E. Saputri, "Sistem Pendukung Keputusan Penentuan Produk Terbaik Menggunakan Metode Preference Selection Index," *Chain J. Comput. Technol. Comput. Eng. Informatics*, vol. 1, no. 2, pp. 56–67, 2023.

[14] A. K. Estetikha, K. Kusrini, and A. H. Muhammad, "Metode Preference Selection Index Dalam Menentukan Distribusi Alat Pelindung Diri di Yogyakarta," *Journal of Information Systems and Informatics*, vol. 3, no. 4, pp. 740–749, 2021.

[15] R. K. Hondro, "Sistem Pendukung Keputusan Pemilihan Klinik Hewan Terbaik Menggunakan Metode PSI (Preference Selection Index)," *Jurnal Ilmiah Core IT: Community Research Information Technology*, vol. 9, no. 3, 2021.