Journal of Informatics, Electrical and Electronics Engineering
ISSN 2807-9507 (Media Online)
Vol 1, No 3, Maret 2022
Hal 110-118
https://djournals.com/jieee

Modifikasi Algoritma Lucifer Dengan Menerapkan Pembangkitan Kunci Berdasarkan Naive Shuffle

Berkat Kasih Laia

Falkultas Ilmu Komputer Dan Teknologi Informasi, Program Studi Teknik Informatika, Univesitas Budi Darma, Medan, Indonesia
Email : berkatkh1405@gmail.com

Abstrak - Keamanan data merupakan suatu hal dalam menjaga kerahasiaan infromasi terutama yang berisi informasi sensitif yang hanya boleh di ketahui isinya oleh pihak yang berhak saja, apa lagi jika pengirimannya dilakukan melalui jaringan public, apa bila data tersebut tidak diamankan terlebih dahulu, akan sangat mudah disadap dan di ketahui isi informasinya oleh pihak pihak yang tidak memiliki wewenang. Salah satu cara untuk mengamankannya adalah menggunkan teknik kriptografi yaitu dengan menggunakan teknik algoritma Lucifer dan kemudian melakukan pengacakan kunci berdasarkan Naïve Shuffle, ini dilakukan agar kunci dalam sebuah data tersebut jauh lebih acak dan tidak mudah untuk di ketahui isinya. Tenik ini dapat meminimalkan akan terjadinya sebuah kebocoran dari pesan yang dirahasiakan, dan pengirim tidak perlu khawatir terhadap pesan tersebut. Penelitian ini diharapkan dapat menghasilkan dan menunjukkan hasil dalam mengamankan pesan serta dapat mempermudah masyarkat untuk dalam mengamankan pesan rahasia.

Kata Kunci : Keamanan Data; Kriptografi; Pengacakan; Algoritma Lucifer; Algoritma Naïve Shuffle.

Abstract - Data security is a matter of maintaining the confidentiality of information, especially those containing sensitive information that only authorized parties may know, especially if the transmission is done via a public network, what if the data is not secured first, it will be very easy to be tapped and retrieved. know the contents of the information by parties who do not have the authority. One way to secure it is to use cryptographic techniques, namely by using the Lucifer algorithm technique and then randomizing the keys based on the Naïve Shuffle, this is done so that the keys in the data are much more random and not easy to find out the contents. This technique can minimize the occurrence of a leak of confidential messages, and the sender does not have to worry about the message. This research is expected to produce and show results in securing messages and to make it easier for the community to secure secret messages.

Keywords : Data Security; Cryptography; Randomization; Lucifer Algorithm; Naïve Shuffle Algorithm.

1.PENDAHULUAN

Keamanan data pada era teknologi ini sangatlah diperlukan, mengingat semakin meningkatnya pengguna aplikasi sosial media saat ini yang mengirim atau membagikan pesan atau data yang bersifat rahasia lewat aplikasi sosial media tersebut. Aplikasi-aplikasi sosial media seperti facebook, instagram, twitter dan lain-lain, sangat rentan akan pembobolan dan pencurian data. Tidak sedikit orang yang tidak bertanggung jawab melakukan pembobolan untuk melihat atau mencuri isi dari data yang distribusian lewat sosial media tersebut. Oleh karena itu pengamanan data yang didistribusikan melalui sosial media sangat penting dilakukan, agar tidak terjadi hal yang tidak diinginkan[1]. Salah satu teknik yang dapat digunakan adalah teknik kriptografi.

Kriptografi merupakan salah satu ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi, seperti kerahasian data, keaslian data, integritas data, serta autentikasi data. Kriptografi dapat juga diartikan sebagai ilmu atau seni untuk menjaga keamanan pesan, pesan yang dikirim akan berupa kata kunci yang membuat penyadapnya menjadi lebih sulit untuk memecahkan isi pesan tesebut atau dengan kata lain dengan melakukan Enkripsi.

Enkripsi adalah sebuah proses penyandian yang melakukan perubahan kode (pesan) dari yang bisa di mengerti (plaintext) menjadi sebuah kode (pesan) yang tidak bisa di mengerti (ciphertext). Berdasarkan penelitian terdahulu bahwa teknik kriptografi dapat meningkatkan atau dapat membantu dalam mengamankan data yang didistribusikan di sosial media dari orang yang tidak berhak untuk mengetahui isi data tersebut[2]. Salah satu algoritma yang dimiliki oleh teknik kriptografi adalah algoritma Lucifer. Algoritma ini merupakan salah satu jenis algoritma block cipher pertama yang menggunakan feistel cipher. Proses enkripsi dapat dilakukan terhadap 128 bits dari plaintext setiap bloknya dan dengan jumlah kunci 128 bits. Proses enkripsi dilakukan dengan menggunakan fungsi-f di dalam jaringan Feistel yang digunakan.

Salah satu algoritma pengacakan yang umum digunakan adalah algoritma *Naive Shuffle* yang merupakan proses pengacakan yang dilakukan terhadap urutan angka, teks, atau data akan menghasilkan permutasi acak. Apabila proses pengacakan dilakukan secara berulang-ulang sampai n kali, maka permutasi acak yang dihasilkan adalah sejumlah panjang data factorial[4]. Berdasarkan penelitian sebelumnya bahwa algoritma *Naïve Shuffle* sedikit lebih baik dari pada *Fisher Yates Shuffle* dalam pengacakan[4].

Penelitian ini menguraikan bagaimana mengoptimalkan ketahanan kunci algoritma *lucifer* dengan melakukan pengacakan kunci sebelum digunakan dalam proses enkripsi maupun dekripsi. Proses pengacakan kunci dilakukan berdasarkan algoritma *Naive Shuffle*, sehingga kunci algoritma *Lucifer* yang digunakan dalam proses enkripsi dan dekripsi lebih acak.

ISSN 2807-9507 (Media Online)
Vol 1, No 3, Maret 2022
Hal 110-118
https://djournals.com/jieee

Berdasarkan penelitian sebelumnya, mengatakan bahwa pengacakan kunci dalam algoritma ini kurang baik, sehingga masih mudah dipecahkan oleh penyerang[3]. Agar kunci yang digunakan dalam proses enkripsi maupun dekripsi lebih tahan terhadap serangan, maka perlu dilakukan modifikasi pembentukan kuncinya. Salah satu algoritma pengacakan yang umum digunakan adalah algoritma Naive Shuffle yang merupakan proses pengacakan yang dilakukan terhadap urutan angka, teks, atau data akan menghasilkan permutasi acak. Apabila proses pengacakan dilakukan secara berulang-ulang sampai n kali, maka permutasi acak yang dihasilkan adalah sejumlah panjang data factorial[4].

2. METODOLOGI PENELITIAN

2.1 Keamanan Data

Keamanan data atau data security adalah prosedur yang dapat dilakukan dengan dukungan dari regulasi dan teknologi untuk mencegah data dari perusakan data, modifikasi data, serta penyebaran luaskan data penting baik yang disengaja maupun tidak disengaja[5].

2.2 Kriptografi

Secara umum kriptografi adalah suatu ilmu yang mempelajari bagaimana cara membuat suatu pesan yang dikirim oleh pengirim, dapat tersampaikan dengan aman pada penerima dengan menyembunnyikan dalam bentuk sandi atau simbol yang sulit dimengerti. Kriptografi (Criyptography) berasal dari bahasa Yunani yang terdiri dari dua suku kata yaitu "Cryptos" artinya "Secret" (rahasia), dan "graphein" artinya "writing" (tulisan)[6].

2.3 Algoritma Lucifer

Lucifer adalah block cipher pertama yang dibuat. Lucifer pertama kali dikembangkan oleh Horst Feistel dan koleganya di IBM. Lucifer sendiri adalah pendahulu langsung dari block cipher yang sekarang banyak digunakan yaitu Data Encryption Standard (DES)[7]. Langkah-langkah dalam proses enkripsi algoritma lucifer yaitu :

1. Plaintext dibagi menjadi dua bagian yaitu Left bit (L0) dan Right bit (R0).
2. Lakukan perhitungan pertama dengan rumus :
   $Ri+1 = Li \oplus (R, i)$ ..... (4)
   Rumus di atas adalah untuk mecari nilai R dan setelah mendapatkan hasilnya akan digunakan untuk mencari nilai di putaran kedua.
3. Nilai L dicari dengan rumus :
   $Li+1 = Ri$ ..... (5)
   4. Setelah setiap putaran dilakukan kecuali yang terakhir, bagian kanan dan kiri block ditukar.

Proses di atas masing-masing diulang sebanyak 16 ronde. Berdasarkan uraian di atas, terlihat bahwa dalam pengimplementasiannya, algoritma Lucifer telah menggunakan fungsi permutasi dan iterasi.

2.4 Naïve Shuffle

Naive Shuffle merupakan metode pengacakan umum yang dilakukan yang bertujuan untuk menghasilkan permutasi yang seimbang. Adapun kelebihan algortima ini yaitu memiliki nilai acak yang akurat dibandingkan dengan algoritma lainnya. Secara umum kelemahanya algoritma ini yaitu, untuk mendapatkan nilai acak yang akurat harus dilakukan penghitungan nilai berulangkali dengan jumlah data yang lebih besar[4]. Langkah-langkah pengacakan menggunakan algoritma Naive Shuffle[4], yaitu :

1. Siapkan teks atau pesan (dalam bentuk angka ataupun karakter).
2. Pilih salah satu nomor karakter secara acak di antara keseluruhan karakter (panjang karakter -1) dari jumlah angka atau karakter yang belum dianggap teracak (dihapus).
3. Hapus nomor yang dipilih diatas dari daftar karakter yang belum teracak, lalu karakter yang dipilih tersebut dijadikan sebagai karakter result.
4. Ulangi langkah 2 dan 3 hingga semua karakter teracak.
5. Urutan karakter yang dijadikan karakter result sehingga membentuk permutasi acak dari karakter
   Proses ini dapat dilakukan dengan Pseudo Code dari Naïve shuffle sebagai berikut :

```
function algoNaiveShuffle (A)
    for i     A.length-1 down to 1 do
    s = random number from 0 to A.length-1
    swap(A[i],A[s]).
```

3. HASIL DAN PEMBAHASAN

3.1 Pembahasan

Kuat lemahnya metode kriptografi tidak terletak dari hasil enkripsi atau dekripsi, melainkan terletak pada kunci yang digunakan. Setiap metode yang ada dalam algoritma kriptografi memiliki kelemahan masing-masing yang salah satu kelemahan salah satu kelemahan tersebut tersebut ada yang terletak pada kunci yang dibentuk. Sebelum dilakukan

proses pembangkitan kunci berdasarkan algoritma Lucifer. Langkah pertama disediakan kunci awal dan diberikan kepada penerima pesan. Selanjutnya dilakukan pengacakan terhadap posisi karakter kunci awal. Proses pengacakan kunci pada penelitian ini dilakukan Berdasarkan sebuah algoritma pengacak yaitu naïve shuffle. Adapun langkah-langkah yang dilakukan untuk mengenkripsi dan dekripsi pesan berdasarkan algoritma lucifer adalah :

a. Proses Pembentukan kunci
   Proses pembentukan kunci diawali dengan proses pengonversian kunci ke dalam bentuk bilangan biner berdasarkan tabel ASCII, Setelah kunci dikonversi kedalam bentuk bilangan biner.
b. Proses enkripsi
   Proses enkripsi pada tahap ini diawali dengan pengonversian terhadap plaintext kedalam bentuk bilangan biner, setelah dikonversi kedalam bentuk bilangan biner, langkah berikutnya adalah melakukan proses enkripsi. Berdasarkan ketentuan Lucifer tiap block harus 128 bit, kemudian dibagi dua, block awal menjadi left block (L) dan right block (R). Lakukan perhitungan pertama dengan rumus Li=R0, untuk mencari nilai L. Kemudian gunakan rumus $Ri+1 = Li \oplus (R, i)$.
c. Proses dekripsi
   Proses dekripsi merupakan proses kebalikan dari proses enkripsi, hanya saja perbedaannya adalah putaran yang dilakukan dalam proses ini dimulai dari putaran ke 16, sehingga yang dicari adalah R15 dan L15, dengan rumus Rn-1=Ln. Setelah biner-biner dari cipher sudah dikelompokkan menjadi 128-bit kemudian dilakukan XOR antara L16 dengan K16 dengan menggunakan rumus $Li-1 = Li \oplus F(Ri, Ki)$ sehingga didapatkan nilai L15 dan R15. Proses ini diulang sebanyak 16 kali. Sebagai contoh kasus yang digunakan pada analisa ini adalah "INIPINNYA_303044" sebagai pesan yang akan didistribusikan kepada penerima (plaintext) dan "BERKATKASIHLAIA0" sebagai kuncinya.
d. Proses Pembentukan Kunci
   Sebelum dilakukan proses pembangkitan kunci berdasarkan Lucifer, langkah pertama yang dilakukan adalah melakukan proses pengacakan kunci berdasarkan algoritma Naïve Shuffle. Dimisalkan kata kunci awal adalah BERKATKASIHLAIA0. Agar lebih mudah dalam proses pengacakan, maka kata kunci dipetakan dengan index karakternya masing-masing, sebagai berikut :

Karakter Kunci	B	E	R	K	A	T	K	A	S	I	H	L	A	I	A	0
Index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

   Proses pengacakan kunci awal berdasarkan algoritma naive shuffle dilakukan berdasarkan langkah-langkah yang telah dituangkan.
e. Misalnya nilai acak bilangan yang diambil (K) adalah 3 maka, posisi karakter ke-3 dan karakter ke-4 (posisi setelah nilai k) ditukar atau posisi karakter ke-4 bergeser ke posisi karakter ke-3 sehingga posisi karakternya menjadi 1 2 4 5 6 7 8 9 10 11 12 13 14 15 16. Posisi karkater ke-3 atau karakter R yang dipilih (n) akan hapus dari susunan karakter yang awal (susunan karakter yang belum teracak) dan diletakkan pada variabel result. Sehingga susunan karakter yang belum teracak adalah B E K A T K A S I H L A I A 0 .
f. Proses pengacakan karakter yang lainnya dilakukan seperti langkah tersebut di atas hingga seluruh karakter dari kata kunci awal teracak. Hasil akhir pengacakan, disajikan pada tabel 3.1.

Tabel 1. Hasil Proses Pengacakan Kunci Berdasarkan *Naïve Shuffle*

No	*Range*	*Roll (K)*	*Scratch*																*Result (Karakter Acak)*	
Karakter kunci			*B*	*E*	*R*	*K*	*A*	*T*	*K*	*A*	*S*	*I*	*H*	*L*	*A*	*I*	*A*	**0**	***Posisi***	**Karakter**
Posisi/index			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16		
1	1-16	3	B	E	K	A	T	K	A	S	I	H	L	A	I	A	0		3	R
2	1-15	6	B	E	K	A	K	A	S	I	H	L	A	I	A	0			5 3	TR
3	1-14	12	B	E	K	A	K	A	S	I	H	L	A	A	0				12 5 3	ITR
4	1-13	12	B	E	K	A	K	A	S	I	H	L	A	0					12 12 5 3	AITR
5	1-12	12	B	E	K	A	K	A	S	I	H	L	A						12 12 12 5 3	0AITR
6	1-11	5	B	E	K	A	A	S	I	H	L	A							5 12 12 12 5 3	K0AITR
7	1-10	4	B	E	K	A	S	I	H	L	A								4 5 12 12 12 5 3	AK0AITR
8	1-9	8	B	E	K	A	S	I	H	A									8 4 5 12 12 12 5 3	LAK0AITR
9	1-8	8	B	E	K	A	S	I	H										8 8 4 5 12 12 12 5 3	ALAK0AITR
10	1-7	4	B	E	K	S	I	H											4 8 8 4 5 12 12 12 5 3	AALAK0AITR
11	1-6	6	B	E	K	S	I												6 4 8 8 4 5 12 12 12 5 3	HAALAK0AITR
12	1-5	3	B	E	S	I													3 6 4 8 8 4 5 12 12 12 5 3	KHAALAK0AITR
13	1-4	3	B	E	I														3 3 6 4 8 8 4 5 12 12 12 5 3	SKHAALAK0AITR
14	1-3	1	E	I															1 3 3 6 4 8 8 4 5 12 12 12 5 3	BSKHAALAK0AITR
15	1-2	2	E																2 1 3 3 6 4 8 8 4 5 12 12 12 5 3	IBSKHAALAK0AITR
																		Hasil Pengacakan	1 2 1 3 3 6 4 8 8 4 5 12 12 12 5 3	EIBSKHAALAK0AITR

Hasil dari proses pengacakan kunci awal berdasarkan algoritma *naïve shuffle* yaitu 1 2 1 3 3 6 4 8 8 4 5 12 12 12 5 3 dan bila dirubah menjadi karakter, makaakan menghasilkan E I B S K H A A L A K 0 A I T R. Langkah selanjutnya adalah proses pembangkitan kunci berdasarkan *Lucifer*. kata kunci awal yang telah diacak berdasarkan algoritma *naïve shuffle*, yaitu E I B S K H A A L A K 0 A I T R.

ISSN 2807-9507 (Media Online)
Vol 1, No 3, Maret 2022
Hal 110-118
https://djournals.com/jieee

Tabel 2. Hasil Konversi Kunci Ke Biner

Char	Biner	Nomor Key Byete
E	01000101	0
I	01001001	1
B	01000010	2
S	01010011	3
K	01001011	4
H	01001000	5
A	01000001	6
A	01000001	7
L	01001100	8
A	01000001	9
K	01001011	10
0	00110000	11
A	01000001	12
I	01001001	13
T	01010100	14
R	01010010	15

Tabel di atas merupakan hasil konversi kunci ke dalam bilangan biner yang disusun berdasarkan bilangan biner ASCII. Setelah kunci dikonversi kedalam bentuk bilangan biner, tahap berikutnya adalah mengatur pengaksesan kunci bedasarkan tabel *Key byte access schedule.* Melalui *tabel message byte*, maka nomor *byte* karakter kunci akan diacak susunannya.

Tabel 3. *message byte*

		MESSAGE BYTE							
		0	1	2	3	4	5	6	7
C-I-D ROUND	1	0	1	2	3	4	5	6	7
	2	7	8	9	10	11	12	13	14
	3	14	15	0	1	2	3	4	5
	4	5	6	7	8	9	10	11	12
	5	12	13	14	15	0	1	2	3
	6	3	4	5	5	7	8	9	10
	7	10	11	12	13	14	15	0	1
	8	1	2	3	4	5	6	7	8
	9	8	9	10	11	12	13	14	15
	10	15	0	1	2	3	4	5	6
	11	6	7	8	9	1	11	12	13
	12	13	14	15	0	1	2	3	4
	13	4	5	6	7	8	9	10	11
	14	11	12	13	14	15	0	1	2
	15	2	3	4	5	6	7	8	9
	16	9	10	11	12	13	14	15	0

Hasil penjadwalan kunci berdasarkan tabel *message byte* ditunjukkan pada tabel 3.4.

Tabel 4. Tabel hasil proses *message byte key*

index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1	01000101	01001001	01000010	01010011	01001011	01001000	01000001	01000001
2	01000001	01001100	01000001	01001011	00110000	01000001	01001001	01010100
3	01010100	01010010	01000101	01001001	01000010	01010011	01001011	01001000
4	01001000	01000001	01000001	01001100	01000001	01001011	00110000	01000001
5	01000001	01001001	01010100	01010010	01000101	01001001	01000010	01010011
6	01010011	01001011	01001000	01000001	01000001	01001100	01000001	01001011
7	01001011	00110000	01000001	01001001	01010100	01010010	01000101	01001001
8	01001001	01000010	01010011	01001011	01001000	01000001	01000001	01001100
9	01001100	01000001	01001011	00110000	01000001	01001001	01010100	01010010
10	01010010	01000101	01001001	01000010	01010011	01001011	01001000	01000001
11	01000001	01000001	01001100	01000001	01001011	00110000	01000001	01001001
12	01001001	01010100	01010010	01000101	01001001	01000010	01010011	01001011
13	01001011	01001000	01000001	01000001	01001100	01000001	01001011	00110000
14	00110000	01000001	01001001	01010100	01010010	01000101	01001001	01000010
15	01000010	01010011	01001011	01001000	01000001	01000001	01001100	01000001
16	01000001	01001011	00110000	01000001	01001001	01010100	01010010	01000101

Berdasaran tabel di atas (tabel 3.4), maka diketahui *sub key* yang akan di digunakan untuk melakukan proses enkripsi dan dekripsi pada setiap *round* adalah sebagai berikut :

K_0 = 010001010100100101000010010100110100101101001000010000010100000 1
K_1 = 010000010100110001000001010010110011000001000001010010010101010 0
K_2 = 010101000101001001000101010010010100001001010011010010110100100 0
K_3 = 010010000100000101000001010011000100000101001011001100000100000 1
K_4 = 010000010100100101010100010100100100010101001001010000100101001 1
K_5 = 010100110100101110100100001000001010000010100110001000001010010 11
K_6 = 010010110011000001000001010010010101010001010010010001010100100 1
K_7 = 010010010100001001010011010010110100100001000001010000010100110 0
K_8 = 010011000100000101001011001100000100000101001001010101000101001 0
K_9 = 010100100100010101001001010000100101001101001011010010000100000 1
K_{10} = 010000010100000101001100010000010100101100110000010000010100100 1
K_{11} = 010010010101010001010010010001010100100101000010010100110100101 1
K_{12} = 010010110100100001000001010000010100110001000001010010110011000 0
K_{13} = 001100000100000101001001010101000101001001000101010010010100000 10
K_{14} = 010000100101001101001011010010000100000101000001010011000100000 1
K_{15} = 010000010100101100110000010000010100100101010100010100100100010 1

g. Proses Enkripsi

Tahap selanjutnya setelah proses penyusunan *sub key* pada kunci adalah proses enkripsi *plaintext*. tahap pertama yang dilakukukan adalah mengkonversi *plaintext* kedalam bentuk bilangan biner. *Plaintextnya* adalah INIPINNYA_303044 diubah menjadi bilangan biner.

Tabel 5. Hasil Konversi *Plaintext* Ke Biner

Char	Biner	Char	Biner
I	01001001	A	01000001
N	01001110	_	01011111
I	01001001	3	00110011
P	01010000	0	00110000
I	01001001	3	00110011
N	01001110	0	00110000
N	01001110	4	00110100
Y	01011001	4	00110100

Biner*plaintext* pada tabel 3.5 di atas bila digabungkan akan menghasilkan biner berikut ini :

0100100101001110010010010101000001001001010011100100111001011001
0100000101011111001100110011000000011001100110000001101000011 0100

Biner *plaintext* yang berjumlah 128 bit tersebut, dibagi menjadi dua *block* awal menjadi *left block* (L) 64 bit dan *right block* (R) 64 bit sebagai berikut :

L_0 = 0100100101001110010010010101000001001001010011100100111001011001
R_0 = 0100000101011111001100110011000000011001100110000001101000011 0100

Proses enkripsi dilakukan sebanyak 16 *round* dengan menggunakan rumus yang telah disediakan *lucifer*. Rumus untuk mencari nilai L adalah $L_1 = R_0$ dan rumus untuk mencari nilai R adalah $R_1 = L_0 \oplus F(R_0,K_0)$, prosesnya sebagai berikut :

Putaran 1 :

L_1 = R_0, maka:
L_1 = 0100000101011111001100110011000000011001100110000001101000011 0100
R_1 = $L_0 \oplus F(R_0,K_0)$
0100000101011111001100110011000000011001100110000001101000011 0100
0100010101001001010000100101001101001011010010000100000101000001 ⊕
= 0000010000010110011100010110001101111000011110000111010101110101
0100100101001110010010010101000001001001010011100100111001011001
R_1 = 0100110101011000001110000011001100110001001101100011101100101100

Putaran 2 :

L_2 = R_1, maka:
L_2 = 0100110101011000001110000011001100110001001101100011101100101100
R_2 = $L_1 \oplus F(R_1,K_1)$
0100110101011000001110000011001100110001001101100011101100101100
0100000101001100010000010100101100110000010000010100100101010100 ⊕
= 0000110000010100011110010111100000000001011101110111001001111000
0100000101011111001100110011000000011001100110000001101000011 0100

R_2 = 010011010100101101001010010010000011001001000111010001100100 1100

Proses ini dilakukan sebanyak 16 putaran. Berdasarkan proses di atas, maka hasil dari seluruh *round* pada proses enkripsi di atas disajikan pada tabe l 3.6.

Tabel 3.6 Hasil *round* Proses Enkripsi

No	Li	Ri
1	01000001010111110011001100110000001100110010000011010000110100	01001101010110000011100000110011001100010011011000111001100
2	01001101010110000011100000110011001100010011011000111001100101	01001101010010110100101001001000001100100100011110100110011001
3	01001101010010110100101001001000001100100100111010001100101100	01010100010000010011011100110010010000010010001000111010001100
4	01010100010000010011011100110010010000010010001000111010001100	01010001010010110011110000110110001100100010111001000000000100101
5	01010001010010110011110000110110001100100010111001000000000100101	01000100010000110101111101010110001101100100010100111010001100
6	01000100010000110101111101010110001101100100101001111010001100	01000110010000110010101100100001010001010010011100111010001100
7	01000110010000110010101100100001010001010010011100111010001100	01001001001100000011010100111110001001110011000001000010000100111
8	01001001001100000011010100111110001001110011000001000010000100111	01000110001100010100110101010100001010100101011000111010000100111
9	01000110001100010100110101010100001010100101101100011000000101011	01000011010000000011001101011010010011000010111100100000000101110
10	01000011010000000011001101011010010011000010111100100000000101110	01010111001101000011011101001100001101010011001001011000000110100
11	01010111001101000011011101001100001101010011001001011000000110100	01010101001101010100100001010111001100100010110100111001100101011
12	01010101001101010100100001010111001100100011010011100111001100101	01001011010101010010110101011111001001110010111010011100111001100
13	01001011010101010010110101011111001001110010111010011000100101100	01010101001010000010010001001000001100000001100010100000000100111
14	01010101001010000010010001001000001100000011000010000000100101100	00101110001111000100000001000010001011000010100100111001100101011
15	00101110001111000100000001000010001011000010100100111010001100101	00111001010001110010111101000010010111010101100100111001100101111
16	00111001010001110010111101000010010111010101100100111000000101011	01010110001100000101111101000001001110000010010001011100000100111

Langkah berikutnya adalah menggabungkan nilai R16 dan L16 sehingga menghasilkan 128 bit, kemudian dikelompokkan menjadi 8 bit perkelompok untuk menghasilkan *ciphertext*.

Cipher = R_{16},$L_{16,}$ sehinggan biner *ciphertext* adalah :

01010110001100000101111101000001001110000010010001011110010010110011100101000111001011110100 00100101110101011001001101110010111

kemudian bagi menjadi 8 bit perkelompok lalu konversi ke desimal dan ke karakter agar dihasilkan *plaintext*.

01010110 00110000 01011111 01000001 00111000 00100100 01011110 01001011 00111001 01000111 00101111 01000010 01011101 01011001 00110111 00101111

Bedasarkan konversi ke desimal memperoleh hasil *ciphertext*, yaitu **V0_A8$^K9G/B]Y7/**, yang merupakan hasil enkripsi.

1. Proses Dekripsi

Proses enkripsi ini merupakan proses yang dilakukan oleh penerima pesan untuk mengetahui isi dari pesan yang sudah dienkripsi tesebut. *Ciphetext* yang digunakan yang menjadi input pada proses dekripsi ini adalah *cipher* yang dihasilkan pada proses enkripsi yang di atas **V0_A8$^K9G/B]Y7/**. Kunci yang digunakan untuk proses dekripsi ini merupakan kunci yang dibagikan kepada penerima pesan (kunci awal) oleh pengirim pesan. Kemudian kunci tersebut diacak berdasarkan algoritma *naïve shuffle* untuk mendapatkan kunci yang acak, proses ini sama ini sama seperti proses pengacakan diatas, kemudian hasil dari proses pengacakan tersebut akan digunakan pada proses dekripsi, karna kunci yang digunakan pada proses enkripsi diatas yaitu hasil pengacakan kunci awal berdasarkan *naïve shuffle*. Kunci awal = BERKATKASIHLLAIA0 (kunci yang dibagi ke pada penerima). Setelah dilakukan pengacakan berdasarkan algoritma *naïve shuffle* adalah E I B S K H A A L A K 0 A I T R.

Setelah proses pengacakan kunci langkah selanjutnya adalah proses pembangkitan kunci berdasarkan algoritma *lucifer*. Proses pembangkitan kunci pada proses dekripsi sama proses yang dilakukan pada proses enkripsi hanya saja susunan kunci yang digunakan terbalik, hasil dari proses tersebut sebagai berikut :

K_0 = 010001010100100101000010010100110100101101001000010000010100000 1

K_1 = 0100000101001100010000010100101100110000010000010100100101010100
K_2 = 0101010001010010010001010100100101000010010100110100101101001000
K_3 = 0100100001000001010000010100110001000001010010110011000001000001
K_4 = 0100000101001001010101000101001001000101010010010100001001010011
K_5 = 0101001101001011010010000100000101000001010011000100000101001011
K_6 = 0100101100110000010000010100100101010100010100100100010101001001
K_7 = 0100100101000010010100110100101101001000010000010100000101001100
K_8 = 0100110001000001010010110011000001000001010010010101010001010010
K_9 = 0101001001000101010010010100001001010011010010110100100001000001
K_{10} = 0100000101000001010011000100000101001011001100000100000101001001
K_{11} = 0100100101010100010100100100010101001001010000100101001101001011
K_{12} = 0100101101001000010000010100000101001100010000010100101100110000
K_{13} = 0011000001000001010010010101010001010010010001010100100101000010
K_{14} = 0100001001010011010010110100100001000001010000010100110001000001
K_{15} = 0100000101001011001100000100000101001001010101000101001001000101

Ciphertext = **V0_A8$^K9G/B]Y7/**. Proses selanjutnya adalah penerima pesan melakukan konversi *ciphertext* ke dalam bentuk bilangan biner, hasilnya disajikan pada tabel 3.7.

Tabel 7. Hasil Konversi *Ciphertext* Ke Bentuk Bilangan Biner

Char	Biner	Char	Biner
V	01010110	9	00111001
0	00110000	G	01000111
_	01011111	/	00101111
A	01000001	B	01000010
8	00111000	]	01011101
$	00100100	Y	01011001
^	01001110	7	00110111
K	01001011	/	00101111

Langkah selanjutnya adalah menggabungkan biner-biner dari *ciphertext* dan membaginya menjadi dua kelompok (masing-masing 64 bit)

L16 = 0101011000110000010111110100000100111000001001000101111001001011
R16 = 0011100101000111001011110100001001011101010110010011011100101111

Langkah selanjutnya adalah proses enkripsi sebagai berikut Putaran 16 :

$L_{15} = R_{16}$, maka :
L_{15} = 0011100101000111001011110100001001011101010110010011011100101111
$R_{15} = L_{16} \oplus F(R_{16},K_{15})$
0011100101000111001011110100001001011101010110010011011100101111
0100000101001011001100000100000101001001010101000101001001000101 ⊕
= 0111100000001100000111110000001100010100000001101011001010110101 0
0101011000110000010111110100000100111000001001000101111001001011
R_{15}= 0010111000111100010000000100001000101100001010010011101100100001

Putaran 15 :
$L_{14} = R_{15}$, maka :
L_{14} = 0010111000111100010000000100001000101100001010010011101100100001
$R_{14} = L_{15} \oplus F(R_{15},K_{14})$
0010111000111100010000000100001000101100001010010011101100100001
0100001001010011010010110100100001000001010000010100110001000001 ⊕
= 0110110001101111000001011000010100110110101101000001110111011000000
0011100101000111001011110100001001011101010110010011011100101111
R_{14} = 0101010100101000001001000010010000001100000001100010100000000001001111

Berdasarkan proses di atas, maka hasil dari seluruh *round* pada proses enkripsi di atas disajikan pada tabel 8.

Tabel 8. Hasil *Roznd* Proses Enkripsi

No	Li	Ri
1	0100000101011111001100110011000000110011001100000011010000110100	0100110101011000001110000011001100110001001101100011101100101100
2	0100110101011000001110000011001100110001001101100011110011001011 00	0100110101001011010010100100100000110010010001110100011001001100
3	0100110101001011010010100100100000110010010001110100011001001100	0101010001000001001101110011001001000001001000100011011000101000

4	0101010001000001001101110011001001000001001000100 011011000101000	0101000101001011001111000011011000110010001011100100000000100101
5	0101000101001011001111000011011000110010001011100100000000100101	0100010001000011010111101010110001101100100010100110100010111 10
6	0100010001000011010111110101011000110110010001010011101000101111 0	0100011001000011001010110010000101000101001001110011010100110000
7	0100011001000011001010110010000101000101001001110011101010011 0	0100100100110000001101010011111000100111001100000100010000100111
8	0100100100110000001101010011111000100111001100000100010000100111	0100011000110001010011010101010000101010010101100011000001011011
9	0100011000110001010011010101010000101010010101100011100000101011 0	0100001101000000001100110110110010011000010111100100000000101110
10	0100001101000000001100110101101001001100001011110100000000101110	0101011100110100001101110100110000110101001100100101100000110100
11	0101011100110100001101110100110000110101001100100101101100000110 0	0101010100110101010010000101011100110010001011010011100101010011
12	0101010100110101010010000101011100110010001011010011100000101011 0	0100101101010101001011010101111001001110010111010011001000101100
13	0100101101010101001011010101111001001110010111010011100000100101 1	0101010100101000001001000100100000110000001100001010000000100111 1
14	0101010100101000001001000100100000110000001100001010010000000100111 1	0010111000111100010000000100001000101100000101001001110110100001
15	0010111000111100010000000100001000101100001010010011101001000111	0011100101000111001011111010000100101110101011001001101110010111
16	0011100101000111001011110100001001011101010110010011101011001011 1	0101011000110000010111110100000100111000001001000101111001001011

Langkah berikutnya adalah menggabungkan nilai R0 dan L0 yang dihasilkan pada putaran akhir (putaran 1) sehingga menghasilkan 128 bit, kemudian dikelompokkan menjadi 8 bit perkelompok untuk menghasilkan *ciphertext*.
Cipher = R_1, L_1.
Sehinggan biner *ciphertext* adalah :
01001001010011100100100101010000010010010100111001001110010110011
01000001010111110011001100110000001100110011000000110100000110100
kemudian bagi menjadi 8 bit perkelompok lalu konversi ke desimal dan ke karakter agar dihasilkan *plaintext*.
01001001 01001110 01001001 01010000 01001001 01001110 01001110 01011001
01000001 01011111 00110011 00110000 00110011 00110000 00110100 00110100
Konversi biner-biner di atas menjadi karakter, hasilnya disajikan pada tabel 3.9 di bawah ini.

Tabel 9. Konversi Biner Hasil Proses Dekripsi

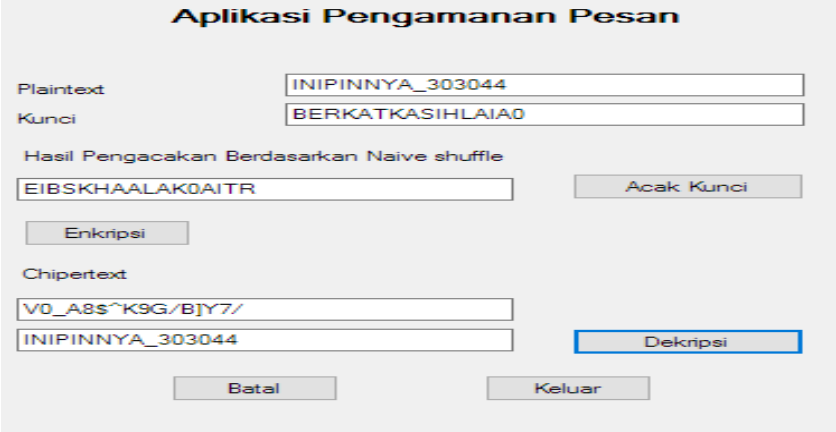
Biner	Karakter	Biner	Karakter
01001001	I	01000001	A
01001110	N	01011111	_
01001001	I	00110011	3
01010000	P	00110000	0
01001001	I	00110011	3
01001110	N	00110000	0
01001110	N	00110100	4
01011001	Y	00110100	0

Bedasarkan konversi ke karakter, maka karakter dari biner-biner *ciphertext* di atas adalah **INIPINNYA_303040**, yang merupakan hasil dekripsi sekaligus isi pesan yang diterima oleh penerima pesan.

3.2 Implementasi

Implementasi merupakan suatu proses penerapan cara kerja algoritma *Lucifer* yang telah dimodifikasi berdasarkan *Naïve Shuffle* dalam sebuah *software* dan dapat dijalankan berdasarkan hasil analisa dan perancangan yang telah dibuat ke dalam suatu bahasa program**.**

a. Pengujian Proses Enkripsi dan Dekripsi
   Pengujian ini dilakukan untuk mengetahui apakah aplikasi dapat melakukan proses enkripsi dan dekripsi pesan berdasarkan algoritma *lucifer* dengan kunci yang diacak berdasarkan *navi shuffle*.



Gambar 4.3 Proses Enkripsi dan Dekripsi

b. Pengujian Waktu Proses Enkripsi dan Dekripsi
Pengujian waktu proses enkripsi dan dekripsi yang dilakukan meliputi pengujian terhadap kalkulasi waktu yang dibutuhkan pada keseluruhan proses mulai dari pengacakan kunci, proses enkripsi maupun waktu yang dibutuhkan pada saat melakukan proses dekripsi. Pengujian ini mengunakan *plaintext* sebanyak 1 blok (128 bit) atau 16 karkter dan kunci yang digunakan sebanyak 1 blok (128 bit) atau 16 karakter. Proses pengujian yang yang dilakukan pada program aplikasi ini dapat dilihat pada tabel 4.1.

Tabel 10. Pengujian Proses Enkripsi dan Dekripsi

No	Plaintext	Kunci Awal	Kunci Hasil Naive	Ciphertext	Waktu Proses
1	INIPINNYA_303044	BERKATKASIHLAIA0	EIBSKHAALAK0AITR	V0_A8$^K9G/B]Y7/	30dtk
2	081355557899_INI	BERKATKASIHLAIA0	EIBSKHAALAK0AITR	0(R/&_7CD*TQ%FG#	30 dtk
3	INIPINNYA_303044	COBABUKASENDIRI_	CBUASII_OEAKRDNB	^BV$@8Q_]<FE0[X3	35 dtk
4	PSWD_SPASI_4KALI	LAPTOP_ASUSPUTIH	APHUPOTSUPTI_LSA	9E[@^T!YC)D,S”5_	38 dtk
5	LAPTOP_ASUSPUTIH	PSWD_SPASI_4KALI	AAPS_LSD_I4ISWPK	70%,X(DRC*V^_QB2	30

4. KESIMPULAN

Berdasarkan hasil penelitian yang telah dilakukan oleh penulis, maka penulis menarik beberapa kesimpulan yang terkait dengan proses penelitian maupun hasil yang didapatkan dari penelitian. Adapun kesimpulannya adalah sebagai berikut.

a. Keacakan kunci yang digunakan pada algoritma *Lucife r*kurang acak, sehingga masih dapat dengan mudah diketahui oleh penyerang untuk mendapatkan *plain* dari *cipher* yang dihasilkan, sehingga sangat perlu dilakukan pengacakan susunan karakter kata kunci sebelum digunakan dalam proses enkripsi pesan/data rahasia berdasarkan algoritma *lucifer*.
b. Penambahan proses pengacakan kata kunci yang digunakan oleh algoritma *lucifer* dapat mempersulit upaya para penyerang dalam memecahkan algoritma ini, karena kunci yang digunakan dalam proses enkripsi dan dekripsi merupakan hasil pengacakan dari kata kunci awal yang didisribusikan kepada penerima pesan, sehinggakeamanan data rahasia lebih optimal.
Pengimplementasipengamanan data rahasia dalam bentuk aplikasi keamanan sangat membantu dan mempermudah pengguna dalam melakukan proses pengamanan data rahasia yang didistribusikan kepada orang lain.

REFERENCES

[1] J. R. Batmetan, M. F. Lumingkewas, C. Tumuyu, and P. P. Ante, “Analisis Perilaku Keamanan Informasi Pengguna Sosmed Dikalangan Generasi Milenial.”
[2] Budi Hartono, “RUANG LINGKUP KRIPTOGRAFI,” vol. IX, no. 2, 2004.
[3] D. Anggara and A. S. Sembiring, “PENINGKATAN KEAMANAN DATA TEKS TERENKRIPSI ALGORITMA LUCIFER,” vol. 3, pp. 439–445, 2019.
[4] A. Farisi, “Analisis Perbandingan Algoritma Fisher Yates Shuffle dan Naive Shuffle,” pp. 1–10.
[5] Forsepoint, “Data Security,” 2017. [Online]. Available: https://www.forcepoint.com/cyber-edu/data-security.
[6] Munir Reinaldi, *Kriptografi*. Bandung: Penerbit Informatika, 2006.
[7] M. Mizan, “STUDI PERBANDINGAN ALGORITMA SIMETRI LUCIFER DAN BLOWFISH,” no. 10.