Jurnal CyberTech

Vol.4. No.9, September 2021, pp. xx~xx
P-ISSN : 9800-3456
E-ISSN : 2675-9802

Implementasi Kriptografi Pengamanan Data Gaji Menggunakan Metode Rivest Code 4 (RC4) Pada PT. Deli Food

Debby Cynthia Balqis [*], Puji Sari Ramadhan [**], Erika Fahmi Ginting [***]
[*] Sistem Informasi, STMIK Triguna Dharma
[**] Sistem Informasi, STMIK Triguna Dharma
[***] Sistem Informasi, STMIK Triguna Dharma

Article Info

Article history:
Received 12th, 2021
Revised 20th, 2021
Accepted 26th, 2021

Keyword:
Kriptografi
Pengamanan RC4
Data Gaji Karyawan

ABSTRACT

Pada PT. Deli Food yang belum memiliki sistem pengamanan data. Dengan adanya Penelitian ini Membahas Tentang bagaimana cara mengamankan data menggunakan kriptografi. Pada PT. Deli Food ada berbagai jenis data penting yang perlu dijaga kerahasiaannya. Yaitu data gaji,data absensi, data pendapatan perusahaan dan surat-surat lainnya, jadi perlunya pemahaman dan pengetahuan mengenai kriptografi agar terjaga keamanan pada surat-surat yang ada pada perusahaan tersebut
Keamanan merupakan salah satu yang diinginkan setiap orang pada saat melakukan komunikasi dan saat melalukan proses pengiriman data yang berisikan pesan penting.
Oleh sebab itu, maka akan dibangun sebuah sistem Algoritma Rives code 4 (RC4) pada kriptografi berbasis desktop untuk mengamankan data notaris. Kriptografi merupakan proses penyandian/penyamaran data yang bertujuan untuk mengamankan data atau informasi yang bersifat rahasia.

Corresponding Author: *First Author
Nama : Debby Cynthia Balqis
Program Studi : Sistem Informasi
STMIK Triguna Dharma
Email: debbycynthiabalqis26@gmail.com

1. PENDAHULUAN

Pada PT. Deli Food di kota Medan masalah pengamanan data gaji karyawan merupakan salah satu data yang paling dijaga kerahasiaanya agar tidak terjadi kebocoran data yang memungkinkan diketahui oleh pihak yang tidak berwenang. Hal ini kurang mendapat perhatian dari perancang atau pengelola sistem informasi. Salah satu hal terpenting dalam menjaga kerahasiaan dan keamanan data adalah dengan proses enkripsi.

Ada berbagai cara dilakukan untuk permasalahan tersebut, untuk menjamin keamananan informasi rahasia tersebut, salah satunya caranya yaitu dengan menyandikan isi informasi asli (plain text) menjadi suatu kode-kode yang tidak dimengerti (chipper text)

2. METODE PENELITIAN

2.1 Pengertian Kriptografi

Keamanan telah menjadi aspek yang sangat penting dari suatu sistem informasi. Sebuah informasi umumnya hanya ditujukan bagi segolongan tertentu. Oleh karena itu sangat penting untuk mencegahnya jatuh kepada pihak-pihak lain yang tidak berkepentingan. Untuk melaksanakan tujuan tersebutlah dirancang suatu sistem keamanan yang berfungsi melindungi sistem informasi *(Rinaldi, 2011).*

a. Pengertian Algoritma Kriptografi

"Dalam kriptografi terdapat dua macam algoritma kriptografi, yaitu: algoritma simetris dan algoritma asimetris".(Rinaldi, 2011)

b. Pengertian Algoritma Simetris

"Algoritma Simetri adalah algoritma yang mempergunakan kunci yang sama pada enkripsi dan dekripsinya" .(Rivest Shamir Adleman,2016)

2.2 Algoritma *RC4*

"RC4 adalah cipher aliran yang digunakan secara luas pada sistem keamanan seperti protokol *Secure Socket Layer* (SSL). Algoritma kriptografi ini sederhana dan mudah diimplementasikan. RC4 dibuat oleh Ron Rivest dari laboratorium RSA (RC adalah singkatan dari *Ron's Code*). RC4 membangkitkan keystream yang kemudian di-XOR-kan dengan *plaintext* pada waktu enkripsi (atau di-XOR-kan dengan bit-bit *ciphertext* pada waktu dekripsi). RC4 tidak seperti *cipher* aliran yang memproses data dalam bit. RC4 memproses data daalam ukuran *byte* (1 *byte* = 8 bit). RC4 menggunakan dua buah kotak substitusi (S-box) array 256 *byte*". (*Slamet Maryono, 2012*)

Rumusan Algoritma *RC4*:

Cara kerja algoritma RC4 yaitu inisialisasi S-Box pertama, S[0],S[1],...,S[255], dengan bilangan 0 sampai 255. Pertama isi secara berurutan S[0] = 0, S[1] = 1,..., S[255]. Kemudian inisialisasi array lain (S-Box lain), misal array K dengan panjang 256. Isi array K dengan kunci ang diulang sampai seluruh array K[0], K[1],..., K[255] terisi seluruhnya.

1. Proses inisialisasi S-Box (Array S)
   For i = 0 to 255, S[i] = i
2. Proses inisialisasi S-Box (Array K)
   For i = 0 to 255, K[i] = i
3. Kemudian lakukan langkah pengacakan S-Box sebagai berikut:
   i = 0 ; j = 0
   for i = 0 to 255 {
   j = (j+S[i] + K[i]) mod 256
   swap S[i] dan S[j] }
4. Setelah itu, buat *pseudo random byte* sebagai berikut:
   i = (i + 1) mod 256
   j = (j + S[i]) mod 256
   swap S[i] dan S[j]
   t = (S[i] + S[j]) mod 256
   K = S[t]
5. Byte K di-XOR-kan dengan plainteks untuk menghasilkan cipherteks atau di-XOR-kan dengan cipherteks untuk menghasilkan plainteks.

Sebagai contoh perhitungan adalah sebagai berikut : Berikut adalah implementasi algoritma RC4 dengan mode 4 byte (untuk lebih menyederhanakan dalam perhitungan manual) serta untuk kebutuhan sistem yang sangat terbatas. S-Box dengan panjang 4 byte, dengan S[0]=0, S[1]=1, S[2]=2 dan S[3]=3 sehingga array S menjadi:

0 1 2 3

Inisialisasi 4 byte kunci array, K. Misalkan kunci Ulang kunci sampai memenuhi seluruh adalah 2 5 7 3, sehingga array K berisi 2 5 7 3 dan mencoba untuk mengenkripsikan kata HALO.

Inisialisasi i dan j dengan 0 kemudian dilakukan KSA agar tercipta state-array yang acak. Penjelasan iterasi lebih lanjut dapat dijelaskan sebagai berikut:

Iterasi 1

i = 0

j = (0 + S[0] + K [0 mod 4]) mod 4

= (0 + 0 + 2) mod 4 = 2

Swap (S[0],S[2])

Hasil Array S

2 1 0 3

Iterasi 2

i = 1

j = (2 + S[1] + K [1 mod 4]) mod 4

= (2 + 1 + 5) mod 4 =0

Swap (S[1],S[0])

Hasil Array S

1 2 0 3

Iterasi 3

i = 2

j = (0 + S[2] + K [2 mod 4]) mod 4

= (0 + 0 + 7) mod 4 = 3

Swap (S[2],S[3])

Hasil

1 2 3 0

Iterasi 4

i = 3

j = (3 + S[3] + K [3 mod 4]) mod 4

= (3 + 0 + 3) mod 4 = 2

Swap (S[3],S[2])

Hasil Array S

1 2 0 3

Setelah melakukan KSA, akan dilakukan PRGA. PRGA akan dilakukan sebanyak 4 kali dikarenakan plainteks yang akan dienkripsi berjumlah 4 karakter. Hal ini disebakan karena dibutuhkan 1 kunci dan 1 kali pengoperasian XOR untuk tiap tiap karakter pada plainteks. Berikut adalah tahapan penghasilan kunci enkripsi dengan PRGA.

Array S

1 2 0 3

Inisialisasi

i = 0

j = 0

Iterasi 1

i = (0 + 1) mod 4 = 1

j = (0 + S[1]) mod 4 = (0+ 2) mod

4 = 2

swap (S[1],S[2])

1 0 2 3

K1 = S[(S[1]+S[2]) mod 4] = S[2

mod 4] = 2

K1 = 00000010

Iterasi 2

i = (1 + 1) mod 4 = 2

j = (2 + S[2]) mod 4 = (2+ 2) mod

4 = 0

swap (S[2],S[0])

2 0 1 3

K2 = S[(S[2]+S[0]) mod 4] = S[3

mod 4] = 3

K2 = 00000011

Iterasi 3

i = (2 + 1) mod 4 = 3

j = (0 + S[3]) mod 4 = (0+ 3) mod

4 = 3

swap (S[3],S[3])

1 0 2 3

K3 = S[(S[3]+S[3]) mod 4] = S[6

mod 4] = 2

K3 = 00000010

Iterasi 4

i = (3 + 1) mod 4 = 0

j = (3 + S[0]) mod 4 = (3+ 1) mod

4 = 0

swap (S[0],S[0])

1 0 2 3

K1 = S[(S[0]+S[0]) mod 4] = S[2

mod 4] = 2

K4 = 00000010

Setelah menemukan kunci untuk tiap karakter, makadilakukan operasi XOR antara karakter pada plaintext dengan kunci yang dihasilkan. Berikut adalah tabel ASCII untuk tiap-tiap karakter pada plaintks yang digunakan.

Huruf	Kode ASCII (Binary 8 bit)
H	01001000
A	01000001
L	01001100
O	01001111

Berikut adalah proses pengXORan dari plainteks dengan key yang telah didapat:

H A L O : 01001000 01000001 01001100 01001111

Key : 00000010 00000011 00000010 00000010

Cipherteks : 01001010 01000010 01001110 01001101

2.3 Flowchart

Flowchart adalah pengambaran secara grafik dari langkah-langkah dan urutan-urutan prosedur dari suatu program. *Flowchart* menolong analisis dan programer untuk memecahkan masalah kedalam segmen-segmen yang lebih kecil dan menolong dalam menganalisis alternatif lain dalam pengoperasian. *Flowchart* merupakan logika atau urutan-urutan intruksi program. Diagram alur dapat menunjukan secara jelas alur pengendalian algoritma, yakni bagai mana rangkaian pelaksanaan kegiatan. Suatu diagram alur memberi gambaran dua dimensi berupa simbol-simbol grafis masing-masing simbol punya arti tersebut.Menurut Kendall, K.E, dan J.E. Kendall Sistem *Flowchart* merupakan alat yang banyak digunakan untuk menggambarkan sistem secara fisikal (Rosa A.S. dan Salahuddin, 2014).

2.4 UML (*Unified Modeling Language*)

Unified Modelling Language (UML) adalah suatu alat untuk memvisualisasikan dan mendokumentasikan hasil analisa dan desain yang berisi sintak dalam memodelkan sistem secara visual (Braun, et. al. 2001). Juga merupakan satu kumpulan konvensi pemodelan yang digunakan untuk menentukan atau menggambarkan sebuah sistem software yang terkait dengan objek (Whitten, et. al. 2004).

3. ANALISA DAN HASIL

3.1 Algoritma Sistem

3.1.1 Proses Perhitungan Algoritma *RC4*

Algoritma kriptografi *Rivest Code 4 (RC4)* merupakan salah satu algoritma kunci simetris dibuat oleh *RSA Data Security Inc* (*RSA*DSI). Algoritma ini ditemukan pada tahun 1987 oleh Ronald Rivest dan menjadi simbol keamanan *RSA* (merupakan singkatan dari tiga nama penemu: Rivest Shamir Adleman). Algoritma *RC4* menggunakan dua buah *S-Box* yaitu *array* sepanjang 256 yang berisi permutasi dari bilangan 0 sampai 255, dan *Sbox* kedua, yang berisi permutasi merupakan fungsi dari kunci dengan panjang yang variabel. Cara kerja algoritma *RC4* yaitu inisialisasi *S-Box* pertama, S[0],S[1],...,S[255], dengan bilangan 0 sampai 255. Pertama isi secara berurutan S[0] = 0, S[1] = 1,...,S[255] = 255. Kemudian inisialisasi *array* lain (*S-Box* lain), misal *array* K dengan panjang 256. Isi *array* K dengan kunci yang diulangi sampai seluruh *array* K[0], K[1],...,K[255] terisi seluruhnya.

1. Proses inisialisasi *S-Box* (*Array* S)
2. Selanjutnya proses inisialisasi *S-Box* (*Array* K)
3. Kemudian langkah pengacakan *S-Box*
4. Lalu membuat *pseudo random byte*

Byte K di-XOR-kan dengan plainteks untuk menghasilkan cipherteks atau di-XOR-kan dengan cipherteks menghasilkan plainteks. Berikut adalah implementasi algoritma RC4 dengan mode 256 byte.Berikut adalah implementasi algoritma RC4 dengan mode 256 byte.

1. Inisialisasi S-Box dengan panjang 256 byte, dengan S[0]=0, S[1]=1, S[2]=2, S[3]=3,…., S[225]=255, maka *array* S menjadi :

Table 3.1 Inisialisasi S-Box Dengan Panjang 256 *byte*

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

Inisialisasi dengan 8 byte kunci array Ki, misalkan kunci terdiri dari 8 byte yaitu “keuangan” maka kalimat yang akan diubah kedalam bentuk desimal “107 101 119 97 110 103 97 110”. Ulangi kunci hingga memenuhi seluruh array K sehinga array K menjadi :

Tabel 3.2 Tabel Inisialisai 8 *byte* Kunci Array Ki

Iterasi -i	**Key-char**	**Key[i]**	**Sbox[i]**
1	c	107	0
2	u	101	1
3	s	119	2
4	t	97	3
5	o	110	4
6	m	103	5
7	e	97	6
8	r	110	7
….	.....	…..	…..
255	e	97	254
256	r	110	255

1. Berikutnya mencampurkan operasi dimana akan menggunakan variable I dan j ke index array S[i] dan K[i]. Pertama beri nilai inisial unruk I dan j dengan 0, operasi pencampuran adalah perulangan rumusan (j + S[i] + K[i]) mod 256 yang diikuti dengan pertukaran S[i] dengan S[j]. Sebagai contoh, karena kita menggunakan array dengan panjang 256 byte maka algoritma menjadi :
   For i = 0 to 256
   j= (j +S[i] + K[i]) mod 256
   Swap S[i] dengan S[j]
   Dengan algoritma seperti diatas maka dengan nilai awal i= 0 sampai i= 255 akan menghasilkan array S seperti berikut :
   Iterasi ke- 1 :
   i= 0, maka
   j= (j +S[i] + K[i]) mod 256
   =(j +S[0] + K[0]) mod 256
   = (0+0+99) mod 256
   = 99
   Swap S[0] dengan S[99]
   Iterasi ke- 2 :
   i= 0, maka
   j= (j +S[i] + K[i]) mod 256
   =(j +S[1] + K[1]) mod 256
   = (99+1+117) mod 256
   = 217
   Swap S[1] dengan S[217]

Iterasi ke- 3 :
i= 0, maka
j= (j +S[i] + K[i]) mod 256
=(j +S[2] + K[2]) mod 256
= (217+2+115) mod 256
= 78
Swap S[2] dengan S[78]
Iterasi ke- 4 :
i= 0, maka
j= (j +S[i] + K[i]) mod 256
=(j +S[3] + K[3]) mod 256
= (78+3+116) mod 256
= 197
Swap S[3] dengan S[197]
Iterasi ke- 5 :
i= 0, maka
j= (j +S[i] + K[i]) mod 256
=(j +S[4] + K[4]) mod 256
= (197+4+111) mod 256
= 56
Swap S[4] dengan S[56]
Iterasi ke- 5 :
i= 0, maka
j= (j +S[i] + K[i]) mod 256
=(j +S[5] + K[5]) mod 256
= (56+5+109) mod 256
= 170
Swap S[5] dengan S[170]
Iterasi ke- 6 :
i= 0, maka
j= (j +S[i] + K[i]) mod 256
=(j +S[6] + K[6]) mod 256
= (170+6+101) mod 256
= 21
Swap S[6] dengan S[21]
Iterasi ke- 7 :
i= 0, maka
j= (j +S[i] + K[i]) mod 256
=(j +S[7] + K[7]) mod 256
= (21+7+114) mod 256
= 142
Swap S[7] dengan S[142]
…..
Iterasi ke- 255 :
i= 0, maka
j= (j +S[i] + K[i]) mod 256
=(j +S[254] + K[254]) mod 256
= (140+254+101) mod 256
= 36
Swap S[254] dengan S[36]
Iterasi ke- 256 :
i= 0, maka
j= (j +S[i] + K[i]) mod 256
=(j +S[255] + K[255]) mod 256
= (36+222+114) mod 256
= 116
Swap S[255] dengan S[116]

Hasil yang didapat setelah melakukan seluruh proses iterasi dari 0 sampai dengan 255 dan melakukan pertukaran s-box (swap) adalah sebagai berikut :

Table 3.3 Hasil Pertukaran *S-Box* (*swap*)

232	217	108	106	56	45	10	142	249	13	244	57	238	90	248	124
123	198	60	89	228	24	209	91	30	151	241	128	0	149	66	250
116	122	6	67	254	181	242	227	33	96	64	146	156	171	132	203
166	50	202	125	1	148	93	246	54	28	69	131	218	29	95	207
235	105	214	187	138	46	120	137	7	143	199	41	196	247	94	169
223	98	4	129	231	101	52	180	184	16	224	107	233	168	92	191
192	211	82	112	163	194	35	110	245	183	204	9	28	220	175	127
164	21	88	212	222	200	178	81	229	44	176	159	251	155	160	144
237	134	61	206	15	86	31	174	113	38	75	213	208	243	135	68
153	158	32	214	172	65	193	74	55	23	121	51	34	190	185	63
126	40	73	165	205	141	19	150	70	195	20	225	189	111	157	153
17	152	47	77	197	84	178	62	115	18	102	49	221	252	216	78
130	42	219	100	188	230	85	36	87	253	118	8	71	58	215	37
48	80	210	140	173	12	139	103	162	154	43	192	88	136	27	22
104	234	226	59	53	177	239	97	99	114	3	182	167	255	236	170
147	186	145	109	39	119	72	201	5	240	79	2	161	26	83	11

1. Tahap selanjutnya peroses enkripsi yaitu meng-XOR-kan *pseudo random byte* dengan *plainteks*, misalkan *plainteks* “dedek” , plainteks terdiri dari 6 karakter maka menjadi 6 iterasi. Sebelum melakukan iterasi, terlebihdahulu ubah karakter ke dalam bilangan biner.

Table 3.4 Tabel Biner *Plainteks*

Karakter	**Desimal**	**HexaDesimal**	**Biner**
d	100	64	1100100
e	101	65	1100101
d	100	64	1100100
e	101	65	1100101
k	107	6B	1101011

Berikut iterasi 1 : inisialisasi i dan j dengan i=0, j=0;

i = (i+1) mod 256
= (0+1) mod 256
=1
j= (j+S[i]) mod 256
= (j+S[1]) mod 256
= (0+217) mod 256

= 217
Swap S[1] dan S[217]
Swap S[217] dan S[154]
t= (S[i] + S[j] mod mod 256
= (S[1] + S[154] mod mod 256
= (154 + 217) mod 256
= 115
K= S[t] = S[115] = 212 = 110100

Iterasi 2
i = (i+1) mod 256
= (1+1) mod 256
=2
j= (j+S[i]) mod 256
= (j+S[2]) mod 256
= (217 + 108) mod 256
= 69
Swap S[2] dan S[69]
Swap S[108] dan S[46]
t= (S[i] + S[j] mod 256
= (S[2] + S[46] mod 256
= (46+ 108) mod 256
= 154
K= S[t] = S[154] = 121 = 1111001

Iterasi 3
i = (i+1) mod 256
= (2+1) mod 256
= 3
j= (j+S[i]) mod 256
= (j+S[3]) mod 256
= (69+106) mod 256
j= 175
Swap S[3] dan S[175]
Swap S[106] dan S[153]
t= (S[i] + S[j] mod mod 256
= (S[153] + S[106] mod mod 256
= (153+ 106) mod 256
= 3
K= S[t] = S[3] = 153 = 10011001

Iterasi 4
i = (i+1) mod 256
= (3+1) mod 256
=4
j= (j+S[i]) mod 256
= (j+S[4]) mod 256
= (175+56) mod 256
= 231
Swap S[4] dan S[231]
Swap S[56] dan S[97]
t= (S[i] + S[j] mod mod 256
= (S[97] + S[56] mod mod 256
= (56 + 97) mod 256
= 153
K= S[t] = S[153] = 23 = 10111

Iterasi 5
i = (i+1) mod 256
= 4+1) mod 256 =5
j= (j+S[i]) mod 256

= (j+S[5]) mod 256
= (231+45) mod 256
= 20
Swap S[5] dan S[20]
Swap S[45] dan S[228]
t= (S[i] + S[j] mod mod 256
= (S[288] + S[45] mod mod 256
= (228+ 45) mod 256
= 17
K= S[t] = S[17] = 198 = 11000110

Iterasi 6
i = (i+1) mod 256
= 5+1) mod 256 = 6
j= (j+S[i]) mod 256
= (j+S[6]) mod 256
= (20+10) mod 256
= 30
Swap S[6] dan S[30]
Swap S[10] dan S[66]
t= (S[i] + S[j] mod mod 256
= (S[66] + S[10] mod mod 256
= (66+ 10) mod 256
= 76
K= S[t] = S[76] = 196 = 11000100

Tabel 3.5 Tabel Enkripsi *plainteks* "lelang"

Iterasi	Plainteks				Key (K)			Chiperteks		
	Teks	Des	Hex	Biner	DES	XOR		Teks	Des	Hex
						Biner	Biner			
1	d	100	64	1100100	104	1101000	101001	)	41	29
2	e	101	65	1100101	94	1011110	10111	ETB ()	23	17
3	d	100	64	1100100	29	11101	1001101	M	77	4D
4	e	101	65	1100101	73	1001001	1101	CR ()	13	D
5	k	107	6B	1101011	98	1100010	100011	#	35	23

2. berikut adalah proses pendeskripsian yaitu meng-XOR-kan presudo random byte dengan cipherteks, dan cipherteks nya adalah **"88, 28, 245, 118, 168, 163" .** cipher terdiri dari 5 karakter maka terjadi 5 iterasi. Sebelum melakukan iterasi ubah karakter menjadi bilangan biner.

Tabel 3.6 Tabel biner *cipherteks*

Karakter	Desimal	Hexadesimal	Biner
X	88	58	1011000
	28	1C	11100
Õ	245	F5	11110101
V	118	76	1110110
¨	168	A8	10101000
£	163	A3	10100011

Data dalam bentuk cipherteks sehingga setelah disimpan dapat kembali diubah menjadi plainteks dengan cara melakukan XOR dengan kunci yang sama.

Tabel 3.7 Tabel Deskripsi *cipherteks* **“X, , õ, V, ¨, £”**

Iterasi	Chiperteks				Key (K)	Plainteks				
	Teks	Des	Hex	Biner	Des	XOR		Hex	Des	Teks
						Biner	Biner			
1	X	88	58	1011000	212	110100	1101100	6C	108	d
2		28	1C	11100	121	1111001	1100101	5C	101	e
3	õ	245	F5	11110101	153	10011001	1101100	6C	108	d
4	v	118	76	1110110	23	10111	1100001	61	97	e
5	¨	168	A8	10101000	198	11000110	1101110	6E	110	k

4. KESIMPULAN

Adapun kesimpulan yang dapat diambil dari penelitian yang telah dilakukan adalah sebagai berikut :

1. Proses enkripsi dan dekripsi yang dilakukan pada *database* mahasiswa berhasil dilakukan. *Database* asli (*plaintext*) dapat dienkripsi menjadi *database* yang disandikan (*chipertext)* dan dapat didekripsi menjadi *database* asli kembali.
2. Jumlah karakter pada *database* asli dengan karakter pada *database* enkripsi dan dekripsi sama ukurannya karena proses metode RC4 dilakukan *byte* per *byte*.
3. Proses enkripsi dan dekripsi dengan algoritma RC4 lebih cepat dilakukan karena berbasis *stream cipher* yang melakukan enkripsi *one byte at a time*.
4. Semakin panjang kunci untuk proses enkripsi maka semakin kuat keamanan enkripsi datanya.

UCAPAN TERIMA KASIH

Terima kasih kepada dosen pembimbing Bapak Puji Sari Ramadhan, S.Kom.,M.Kom dan Ibu Erika Fahmi Ginting, S.Kom.,M.Kom. beserta pihak–pihak lainnya yang mendukung penyelesaian jurnal skripsi ini.

REFERENSI

[1] C. A. Sari, et al. " PENYEMBUNYIAN DATA UNTUK SELURUH EKSTENSI FILE MENGGUNAKAN KRIPTOGRAFI VERNAM CIPHER DAN BIT SHIFFTING " *Jurnal of Applied Intelligent System*, vol. 1 No. 3, (179-190) 2016.

[2] F. N. Pabokory, I. F. Astuti, and A. H. Kridalaksana, " IMPLEMENTASI KRIPTOGRAFI PENGAMANAN DATA PADA PESAN TEKS, ISI FILE DOKUMEN, DAN FILE DOKUMEN MENGGUNAKAN ALGORITMA *ADVANCE ENCRYPTION STANDARD,* " *Jurnal Informatikan Mulawarman* vol. 10 No.1 2015.

[3] N. D. Nathasia and A. E. Wicaksono, "PENERAPAN TEKNIK KRIPTOGRAFI STREAM CHIPHER UNTUK PENGAMANAN BASIS DATA," *Jurnal Basis Data, ICT Reseach Center UNAS* vol. 6, no. 1, ISSN. 1978-9483, 2011.

[4] D. Herdansyah and Retantyo Wardoyo, "Implementasi Protokol Diffie-Hellman dan Algoritma RC4 Untuk Keamanan Pesan SMS," *IJCCS,* vol. 5, no. 1, 2011.

[5] A. Zelvina, S. Efendi and D. Arisandi, "Perancangan Aplikasi Pembelajaran Kriptografi Kunci Publik ElGamal Untuk Mahasiswa," *JURNAL DUNIA TEKNOLOGI INFORMASI,* vol. 1, no. 1,(56-62) 2012.

[6] J. Simarmata, "Pengamanan Sistem Komputer" 2006.

[7] Rinaldi and Munir, "Analisi Kriptografi Klasik Jepang",2011-2012.

[8] D. Ariyus, "Keamanan Multimedia",2009.

BIBLIOGRAFI PENULIS

	**Debby Cynthia Balqis** Wanita kelahiran Medan, 28 Desember 1997 saat ini sedang menempuh pendidikan Starata Satu (S-1) di STMIK Triguna Dharma menganbil Jurusan Sistem Informasi dan tertarik dalam Photoshop Nirm : 2015020152
	**Puji Sari Ramadhan, S.Kom.,M.Kom** Beliau merupakan Dosen Tetap STMIK Triguna Dharma yang aktif mengajar dan fokus pada bidang keilmuan kecerdasan buatan dan data sains. Telah menulis 1 buku dibidang Ilmu komputer. Memiliki sebanyak 2 Hak Kekayaan Intelektual (HKI). Menjabat sebagai Ketua Program studi SIstem Informasi NIDN : 0126039201 Program Studi : Sistem Informasi Prestasi : Dosen Terbaik Tahun 2018, Pemenang PDP 2018 dan 2019.
	**Erika Fahmi Ginting, S.Kom.,M.Kom** Beliau merupakan Dosen tetap di STMIK Triguna Dharma Serta menjabat sebagai Sekretaris Prodi NIP : 0117119301 Tempat/lahir: Teupin Gajah, 17 november Alamat : jl.Kopi VII no.1 Perumnas Simalingkar Medan Agama : Islam J.kelamin : Perempuan No. Hp : 082272481758 Email : erikafg04@gmail.com Prestasi : Pemenang hibah Dikti 2021 Bidang keahlian :data mining