Jurnal CyberTech
Vol., No., September 2020, pp.
P-ISSN:
E-ISSN:

PENGAMANAN DATA SURAT-SURAT BERHARGA di KANTOR NOTARIS SUSANTO, S.H., M.Kn., MENGGUNAKAN KRIPTOGRAFI dengan METODE *RIVES CODE 4* (RC4)

Muhammad Muid Maulana*, Azanuddin, Suharsil****
*Program Studi Sistem Informasi, STMIK Triguna Dharma
**Program Studi Sistem Informasi, STMIK Triguna Dharma

Article Info
Article history:

Received
Revised
Accepted

Keyword:
Kriptografi
Algoritma RC4
Pengamanan Data
Pengamanan Data Notaris

ABSTRACT
Pada kantor camat kec. Kedai durian yang belum memiliki sistem pengamanan data.
Dengan adanya Penelitian ini Membahas Tentang bagaimana cara mengamankan data menggunakan kriptografi. Ada berbagai jenis data pada kantor camat kec. Kedai durian yang perlu dijaga kerahasiaannya. Yaitu data penerima bantuan,data bantuan,data agaran belanja, dan surat-surat lainnya, jadi perlunya pemahaman dan pengetahuan mengenai kriptografi agar terjaga keamanan pada data-data di kantor camat 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 penerima bantuan. Kriptografi merupakan proses penyandian/penyamaran data yang bertujuan untuk mengamankan data atau informasi yang bersifat rahasia.

First Author
Nama: Muhammad Muid Maulana
Kampus:STMIK Triguna Dharma
Program Studi : Sistem Informasi
E-Mail : lana.lanuk16@gmail.com

1. PENDAHULUAN

Pada kantor camat Kec. Kedai durian ada berbagai jenis data atau dokumen yang perlu dijaga kerahasiaan datanya diantaranya: data atau dokumen calon penerima bantuan, data bantuan, data anggaran tahunan dan lain sebagainya. Kerusakan atau hilangnya data yang diperoleh dengan biaya yang tinggi dan waktu yang lama tentu saja sangat tidak diinginkan, apalagi data tersebut merupakan data atau informasi sangat rahasia.

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

3. 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).*

3.1 Pengertian Algoritma Kriptografi

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

3.2 Pengertian Algoritma Simetris

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

2.3 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

2.4 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).

2.5 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).

4. ANALISIS 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 “customer” maka kalimat yang akan diubah kedalam bentuk desimal “112 101 110 101 114 105 109 97”. 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	p	112	0
2	e	101	1
3	n	110	2
4	e	101	3
5	r	114	4
6	i	105	5
7	m	109	6
8	a	97	7
….	…..	…..	…..
255	m	109	254
256	a	97	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 :

Iterasike- 1 :

i= 0, maka
j= (j +S[i] + K[i]) mod 256
=(j +S[0] + K[0]) mod 256
= (0+0+112) mod 256= 112

Swap S[0] dengan S[112]

Iterasi ke- 2 :

i= 0, maka
j= (j +S[i] + K[i]) mod 256
=(j +S[1] + K[1]) mod 256
= (112+1+101) mod 256
= 214

Swap S[1] dengan S[214]

Iterasi ke- 3 :

i= 0, maka
j= (j +S[i] + K[i]) mod 256
=(j +S[2] + K[2]) mod 256
= (214+2+110) mod 256
= 70

Swap S[2] dengan S[70]

Iterasi ke- 4 :

i= 0, maka
j= (j +S[i] + K[i]) mod 256
=(j +S[3] + K[3]) mod 256
= (70+3+101) mod 256
= 174

Swap S[3] dengan S[174]

Iterasi ke- 5 :

i= 0, maka
j= (j +S[i] + K[i]) mod 256
=(j +S[4] + K[4]) mod 256
= (174+4+114) mod 256
= 36

…..
…..
…..

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*)

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

1. Tahap selanjutnya peroses enkripsi yaitu meng-XOR-kan *pseudo random byte* dengan *plainteks*, misalkan *plainteks* “jamal” , 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**
j	106	6a	1101010
a	97	61	1100001
m	109	6d	1101101
a	97	61	1100001
l	108	6c	1101100

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+214) mod 256
= 214
Swap S[1] dan S[214]
Swap S[214] dan S[244]
t= (S[i] + S[j] mod mod 256
= (S[1] + S[214] mod mod 256
= (244 + 214) mod 256= 202
K= S[t] = S[202] = 104 = 01101000

Iterasi 2

i= (i+1) mod 256
= (1+1) mod 256
=2
j= (j+S[i]) mod 256
= (j+S[2]) mod 256
= (214 + 52) mod 256
= 10
Swap S[2] dan S[10]
Swap S[52] dan S[132]
t= (S[i] + S[j] mod 256
= (S[2] + S[10] mod 256
= (132+ 52) mod 256
= 184

K= S[t] = S[184] = 94 = 10111110

Iterasi 3

i= (i+1) mod 256
= (2+1) mod 256
=3
j= (j+S[i]) mod 256
= (j+S[3]) mod 256
= (10+71) mod 256
j= 81
Swap S[3] dan S[81]
Swap S[71] dan S[95]
t= (S[i] + S[j] mod mod 256
= (S[10] + S[81] mod mod 256
= (95+ 71) mod 256
= 166
K= S[t] = S[166] = 29 = 00011101

Tabel 3.5 Tabel Enkripsi *plainteks* "jamal"

Iterasi	Plainteks				Key (K)			Chiperteks		
	Teks	Des	Hex	Biner	DES	XOR		Teks	Des	Hex
						Biner	Biner			
1	j	106	6a	1101010	104	1101000	100110	&	38	26
2	a	97	61	1100001	94	1011110	101101	-	45	2D
3	m	109	6d	1101101	29	11101	11110100	¶	244	F4
4	a	97	61	1100001	73	1001001	110000	0	48	30
5	l	108	6c	1101100	98	1100010	111110	>	62	3E

2. berikut adalah proses pendeskripsian yaitu meng-XOR-kan presudo random byte dengan cipherteks, dan cipherteks nya adalah **"41, 23, 77, 13, 35" .** 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
&	38	26	100110
-	45	2D	101101
¶	244	F4	11110100
0	48	30	110000
>	62	3E	111110

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* **"&, -,¶ , 0,>"**

Iterasi	Chiperteks				Key (K)	Plainteks				
	Teks	Des	Hex	Biner	Des	XOR		Hex	Des	Teks
						Biner	Biner			
1	&	38	26	100110	104	1101000	1101010	6a	106	j
2	-	45	2D	101101	94	1011110	1100001	61	97	a
3	¶	244	F4	11110100	29	11101	1101101	6d	109	m
4	0	48	30	110000	73	1001001	1100001	61	97	a
5	>	62	3E	111110	98	1100010	1101100	6c	108	l

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 Azanuddin, S.Kom., M.Kom dan Bapak Suharsil, S.E. M.M. beserta pihak–pihak lainnya yang mendukung penyelesaian jurnal skripsi ini.

DAFTAR PUSTAKA

[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.

[9] Rosa A. S and M. Shalahuddin, "REKAYASA PERANGKAT LUNAK TERSTRUKTUR DAN BERORIENTASI OBJEK," 2014.

[10] J. ENTERPRISE, " OTODIDAK VISUAL BASIC, " 2017.

[11] Andi, "MEMBANGUN APLIKASI DATABASE DENGAN VISUAL BASIC.NET, TUTORIAL 10 HARI," 2007.

[12] A. Kristanto, "PERANCANGAN SISTEM INFORMASI DAN APLIKASINYA," 2018.

BIOGRAFI PENULIS

	Nama : Muhammad Muid Maulana Seorang pria kelahiran Desa Suka Makmur, 16 April 1996. Anak ketiga dari empat bersaudara ini merupakan mahasiswa STMIK Triguna Dharma yg dalam proses menyelesaikan scripsi
	Nama : Azanuddin, S.Kom., M.Kom NIDN : 0126068901 Beliau merupakan dosen tetap STMIK Triguna Dharma dan aktif sebagai pengajar sekaligus sebagai wakil program study Sistem Informasi
	Nama : Suharsil, S.E,. M.M. NIDN : 9901004019 Beliau merupakan dosen tetap STMIK Triguna Dharma dan aktif sebagai pengajar pada bidang Sistem Informasi