BULLETIN OF COMPUTER SCIENCE RESEARCH
ISSN 2774-3659 (Media Online)
Vol 1, No 3, Agustus 2021
Hal 83-92
https://hostjournals.com/bulletincsr

Perbandingan Algoritma RC4+ dan RC4a dalam Pengamanan File Teks

Lisa Oktasari

Program Studi Teknik Informatika, Universitas Budi Darma, Medan, Indonesia
Email: lisa2233okta@gmail.com

Abstrak−Seiring perkembangan teknologi, dokumen yang seharusnya bersifat rahasia bisa diakses oleh orang yang tidak berhak. Maka diperlukan pengamanan untuk melakukan pencegahan atas sampainya informasi ketangan yang tidak berhak. Keamanan dari suatu text menjadi berkurang atau tidak aman dalam hal penyimpanan. Tidak menutupi kemungkinan ada pihak ketiga yang merubah atau mengambil text tersebut, maka text tersebut disandikan menjadi kode-kode yang tidak dipahami, sehingga bila ada pihak ketigayang ingin merubah akan kesulitan dalam menterjemahkan isi text yang sebenarnya dan harus membongkar enkripsinya terlebih dahulu. Keamanan dilakukannya penyandian file text, untuk menghindari masalah tersebut perlu dilakukan proses menyembunyikan atau menyamarkan suatu informasi sedemikian rupa sehingga tidak jatuh kepada pihak lain yang tidak berwenang. Dalam penelitian ini membandingkan metode Algoritma RC4+ dan Algoritma RC4A dalam pengamanan file teks. Aplikasi pengamanan file teks untuk membantu mengatasi masalah keamanan data yang dibuat atau disimpan menggunakan file yang berformat seperti doc, txt.

Kata Kunci: Algoritma RC4+; Algoritma RC4A; Kriptografi

Abstract−Along with the development of technology, documents that are supposed to be confidential can be accessed by unauthorized people. So safeguards are needed to prevent information from reaching unauthorized hands. The security of a text becomes reduced or insecure in terms of storage. It does not cover the possibility of a third party changing or taking the text, then the text is encoded into codes that are not understood, so that if there is a third party who wants to change it will have difficulty translating the actual content of the text and must decrypt it first. The security of encrypting text files, to avoid this problem, it is necessary to carry out a process of hiding or disguising information in such a way that it does not fall to other unauthorized parties. In this study, comparing the RC4 + Algorithm and the RC4A Algorithm in securing text files. Text file security application to help solve security problems of data created or stored using files that are formatted such as doc, txt.

Keywords: RC4+ Algorithm; RC4A Algorithm; Cryptography

1. PENDAHULUAN

Seiring perkembangan teknologi, dokumen yang seharusnya bersifat rahasia bisa diakses oleh orang yang tidak berhak. Maka diperlukan pengamanan untuk melakukan pencegahan atas sampainya informasi ke tangan yang tidak berhak. Keamanan dari suatu teks menjadi berkurang atau tidak aman dalam hal penyimpanan. Tidak menutup kemungkinan ada pihak ketiga yang ingin merubah atau mengambil teks tersebut. Salah satu cara untuk mempertahankan kerahasiaan dari teks tersebut, maka teks tersebut disandikan menjadi kode-kode yang tidak dipahami, sehingga bila ada pihak ketiga yang ingin merubah akan kesulitan dalam menterjemahkan isi teks yang sebenarnya dan harus membongkar enkripsinya terlebih dahulu.

File teks merupakan salah satu bentuk informasi yang berupa data berbentuk teks banyak format teks yang ada seperti doc, txt, docx, pdf. File yang berisi informasi-informasi dalam bentuk teks. Data yang berasal dari dokumen pengolah kata, angka yang digunakan dalam perhitungan, nama dan alamat dalam basis data merupakan contoh masukan data teks yang terdiri dari karakter, angka dan tanda baca. Hanya dirinya dan pihak tertentu yang boleh mengetahuinya. Bahkan para petugas yang bekerja sebagai agen-agen rahasia pun mempunyai data-data rahasia. Perlu diketahui oleh pihak yang mempunyai data rahasia yang tidak boleh dibajak dan diketahui oleh orang lain.

Keamanan dilakukannya penyandian file teks, untuk menghindari masalah tersebut perlu dilakukan proses menyembunyikan atau menyamarkan suatu informasi sedemikian rupa sehingga tidak jatuh kepada pihak lain yang tidak berwenang misalnya seperti data yang bersifat rahasia, data keuangan, seperti data password, pin, maka dilakukan lah proses penyandian file teks tersebut ke dalam bentuk yang tidak dapat dimengerti yaitu berupa kriptografi, Salah satunya dengan menggunakan metode Algoritma RC4+ dan Algoritma RC4A.

Penelitian sebelumnya yang telah dilakukan oleh Dani Iqbal, Asep Roy Panggabean, Indra Williamsyah Sinaga, Taronisokhi Zebua dengan judul "Implementasi Algoritma RC4+ Untuk Mengamankan Pesan Teks Pada Aplikasi Chatting" Penelitian ini menguraikan tentang pengamanan pesan teks pada aplikasi chatting berdasarkan algoritma RC4+. Algoritma ini akan melakukan proses *key schedulling algorithm* dan proses *pseudo random generation algorithm* serta melakukan proses pemutaran *state* dan proses pemindahan sejumlah bit pada posisi tertentu untuk memperoleh kunci yang lebih acak yang dibutuhkan pada proses enkripsi dan dekripsi sehingga diperoleh *cipher* pesan asli yang sulit untuk dipecahkan oleh penyerang [1]

Sedangkan penelitian sebelumnya yang telah dilakukan oleh Nur hayati, Mohammad Andri Budiman, Amer Sharif dengan judul "Implementasi Algoritma RC4A dan MD5 Untuk Menjamin *Confidentiality* dan *Integrity* Pada File Teks. (Sinkron)" Pada FSE 2004, sebuah modifikasi baru dari *RC4* telah diusulkan oleh *souradyuti paul* dan *bart preneel* yang diberi nama RC4A, RC4A adalah *stream cipher* yang berorientasi *byte*. Tahap pembentukan dari RC4A lebih efisien dibanding RC4, tetapi tahap inisialisasinya memerlukan setidaknya dua kali proses inisialisasi dari RC4 [2].

Dalam penelitian ini membandingkan metode Algoritma RC4+ dan Algoritma RC4A dalam pengamanan file teks. Adapun dilakukan perbandingan proses enkripsi dengan tujuan untuk membandingkan dan mengetahui algoritma mana yang lebih baik dari segi kecepatan (Running Time) berdasarkan dari nilai MSE dan PSNR dan kompleksitas algoritma. Pada penelitian ini akan dilakukan enkripsi file teks dengan menggunakan panjang kunci yang berbeda–beda dan file teks yang sama serta enkripsi file teks dengan menggunakan panjang kunci yang sama dan ukuran file teks yang berbeda–beda. Aplikasi pengamanan file teks untuk membantu mengatasi masalah keamanan data yang dibuat atau disimpan menggunakan file yang berformat seperti doc, txt.

2. METODOLOGI PENELITIAN

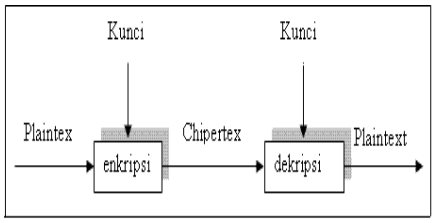
2.1 Kriptografi

Kata *Cryptography* berasal dari bahasa Yunani yang terdiri dari dua kata yaitu *kryptos* yang berarti rahasia dan *graphein* yang berarti tulisan (Mollin, 2007). Menurut terminologinya, kriptografi adalah ilmu dan seni untuk menjaga keamanan pesan ketika pesan dikirim dari suatu tempat ke tempat lain.[3].

2.2 Algoritma

Algoritma ditinjau dari asal usul kata, kata algoritma mempunyai sejarah yang menarik, kata ini muncul dalam kamus Webster sampai akhir tahun 1957 hanya menemukan kata *algorism* yang mempunyai arti proses perhitungan dengan bahasa Arab. Algoritma berasal dari nama penulis buku Arab yang terkenal yaitu Abu Ja'far Muhammad Ibnu Musa al-Khuwarizmi (al-Khuwarizmi dibaca oleh orang barat menjadi *algorism*). Kata *algorism* lambat laun berubah menjadi *algorithm*. Definisi terminologinya algoritma adalah urutan langkah-langkah logis untuk penyelesaian masalah yang disusun secara sistematis. Algoritma kriptografi merupakan langkah-langkah logis bagaimana menyembunyikan pesan dari orang-orang yang tidak berhak atas pesan tersebut.[3] Algoritma kriptografi terdiri dari tiga fungsi dasar yaitu:

1. *Enkripsi* : Enkripsi merupakan hal yang sangat penting dalam kriptografi yang merupakan pengamanan data yang dikirimkan dan terjaga kerahasiannya. Pesan asli disebut *plaintext* yang dirubah menjadi kode-kode yang tidak dimengerti atau disebut dengan *chipertext*.
2. Dekripsi : Dekripsi merupakan kebalikan dari enkripsi, pesan yang telah di enkripsi dikembalikan kebentuk asalnya (plainteks) disebut dengan dekripsi pesan.
3. *Kunci* : Kunci yang dimaksud disini adalah kunci yang dipakai untuk melakukan enkripsi dan dekripsi, kunci terbagi menjadi dua bagian yaitu kunci pribadi (*private key*) dan kunci umum (*public key*). Secara umum fungsi tersebut dapat dilihat pada Gambar 1. [3]



Gambar 1. Data Kunci

2.3 Algoritma Simetri

Algoritma simetri adalah algoritma dimana penyandian kunci enkripsi dan kunci dekripsi bernilai sama. Kunci pada penyandian simetri diasumsikan bersifat rahasia dan hanya pihak yang melakukan enkripsi dan dekripsi yang mengetahui nilainya. Algoritma ini sering disebut dengan algoritma klasik karena memakai kunci yang sama untuk kegiatan enkripsi dan dekripsi. Algoritma ini sudah ada sejak lebih dari 4000 tahun yang lalu. Bila mengirim pesan dengan menggunakan algoritma ini, si penerima pesan harus diberitahu kunci dari pesan tersebut agar bisa mendekripsikan pesan yang dikirim. Keamanan dari pesan yang menggunakan algoritma ini tergantung pada kunci. Jika kunci tersebut diketahui oleh orang lain maka orang tersebut akan dapat melakukan enkripsi dan dekripsi terhadap pesan. Algoritma yang memakai kunci simetri di antaranya adalah DES, RC2, RC4, RC5, RC6, IDEA, AES, A5, OTP, dan sebagainya [4].

2.4 Algoritma RC4+

RC4+ adalah salah satu jenis dari algoritma RC4. Di mana algoritma RC4+ merupakan salah satu lagoritma kunci simetris yang berbentuk *stream cipher* yang melakukan proses enkripsi/dekripsi dalam 1 byte dan menggunakan kunci yang sama.[1] *Stream Cipher* digunakan untuk mengenkripsi *plaintext* menjadi *ciphertext* bit per bit (1 bit setiap kali transformasi) atau byte per byte (1 karakter = 1 byte). Struktur dasar dari RC4+ sama seperti RC4 (Subhadeep, Santanu & Raghu, 2013). RC4 menggunakan variabel yang panjang kuncinya dari 1 sampai 256 bit yang digunakan untuk menginisialisasikan tabel sepanjang 256 bit.

RC4+ menggunakan struktur seperti RC4 dan menambahkan beberapa operasi untuk memperkuat *cipher*. Struktur ini mencoba untuk memanfaatkan poin yang baik pada algoritma RC4 kemudian memberikan beberapa fitur tambahan

untuk batas keamanan yang lebih baik (Paul & Subhamoy, 2012) untuk mengatasi kelemahan RC4 itu sendiri seperti tingginya peluang untuk menghasilkan. Algoritma RC4+ memiliki dua bagian utama yaitu *Key Scheduling Algorithm* (KSA) dan *Pseudo Random Generation Algorithm* (PRGA) [1].

3. HASIL DAN PEMBAHASAN

Penelitian ini menguraikan proses-proses dalam membandingakan kedua algoritma tersebut agar didapatkan *cipherteks* yang sulit dipecahkan. Proses yang dilakukan adalah file teks asli (*plainteks*) terlebih dahulu akan dienkripsi berdasarkan algoritma RC4+ dengan menggunakan kunci penyandian yang sama pada algoritma RC4A sehingga menghasilkan *cipherteks* pertama, kemudian *cipherteks* pertama dienkripsi berdasarkan algoritma RC4+ sehingga akan menghasilkan *plainteks* dan selanjutnya *file* teks akan di enkripsi kembali berdasarkan algoritma RC4A dengan menggunakan kunci penyandian yang sama pada algoritma RC4+ sehingga menghasilkan *cipherteks* kedua dan kemudian didekripsi berdasarkan algoritma RC4A sehingga menghasilkan *plainteks*. Setelah hasil *cipherteks* dari perbandingan algoritma masing-masing telah dihasilkan dengan menggunakan kunci yang penyandian yang sama maka hasil *file* teks menjadi *chiperteks* yang berbeda-beda dan serta ukuran *file* teks.

3.1 Penerapan Algoritma RC4+

Proses Enkripsi file teks dengan Algotima RC4+ sebelum melakukan Enkripsi dimulai *key scheduling* dengan menginisialisasikan *state* awal berupa larik yang terdiri dari 256 elemen, jadi hasil dari *KSA+* adalah permutasi awal setelah itu kunci dimasukan kedalam larik berukuran 256 dengan mengulang kunci hingga larik terisi penuh kemudian Kunci file diubah ke dalam kode *ASCII*. selanjutnya *Pseudo-Random Generation Algorithm(PRGA+)* untuk Enkripsi Setelah melakukan tahap *key scheduling* maka akan dilakukan proses enkripsi file teks. Tahap ini menghasilkan nilai *pseudo-random* yang kemudian di- *XOR*-kan dengan *plainteks* untuk proses enkripsi atau dengan *ciphertext* pada proses dekripsi.

Proses Enkripsi :

Dimisalkan file teks dalam *plainteks* dengan nama “LISA” dan kunci “saya”, Dalam algoritma RC4+ ubah terlebih dahulu *kunci* ke ASCII dari setiap huruf.

Nilai ASCII untuk plainteks “LISA” yaitu :

L = 76
I = 73
S = 83
A = 65

Nilai ASCII untuk kunci “saya” yaitu :

s = 115
a = 97
y = 121
a = 97

Jumlah karakter plainteks = 4 *(Length of Plaintext)*
Jumlah karakter kunci = 4 *(Length of key)*

Tabel 1. Karakter kunci dibuat dalam bentuk array :

Index	0	1	2	3
Kunci	s	a	y	a
Decimal	115	97	121	97

Tahap *Key Scheduling Algorithm (KSA+)* Langkah *key scheduling* dimulai dengan pembentukan Tabel S-Box, dilakukan berdasaran *pseudocode* dan menghasilkan array dengan nilai 0 sampai dengan 255, sehingga tabel S-Box :

Tabel 2. *S-Box*

115	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	**0**	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

ISSN 2774-3659 (Media Online)
Vol 1, No 3, Agustus 2021
Hal 83-92
https://hostjournals.com/bulletincsr

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										

Nilai mulai dari 0 hingga 255
Pada saat nilai *i* = 0, maka
j= (*j* + S[i] + *Key* [*i* mod *keylength*]) Mod 256
j=(0+ S [0] + *Key* [0 mod 4]) mod 256
j=(0+0+*Key* [0]) Mod 256
j=(0 + 0 + 115) Mod 256
j=115 Mod 256
j=115
Nilai S[i] Swap dengan Nilai S[j]
Nilai S [0] ditukar (swap) dengan Nilai S [115]
Maka :
S[0]= 115 dan S [115]=0

Tabel 3. *S-Box*

115	**213**	**55**	**178**	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	**2**
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	**0**	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	**3**	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	**1**	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										

Nilai mulai dari 0 hingga 255
Pada saat nilai *i* = 1, maka
j= (*j* + S[i] + *Key* [*i* mod *keylength*]) Mod 256
j=(115+ S [1] + *Key* [1 mod 4]) mod 256
j=(115+1+*Key* [1]) Mod 256
j=(115 + 1 + 97) Mod 256
j= 213 Mod 256
j= 213
Nilai S[*i*] Swap dengan Nilai S[*j*]
Nilai S [1] ditukar (swap) dengan Nilai S [213]
Maka :
S[1]= 213 dan S [213]=1

Tabel 4. State S [ ] acak proses KSA

Index	1	2	2	3	4	5	6	7	8	9	10	11	12	13
Nilai	16	5	47	54	162	143	127	44	59	235	193	255	143	57
	14	15	16	17	18	19	20	21	22	23	24	25	26	27
	49	131	168	80	215	39	253	114	76	216	36	151	60	97
	28	29	30	31	32	33	34	35	36	37	38	39	40	41

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

Pseudo-Random Generation Algorithm(PRGA+) untuk Enkripsi

PRGA dilakukan sebanyak elemen plainteks. Proses PRGA menggunakan nilai-nilai table S [ ] yang telah melalui proses KSA. Nilai biner dari hasil proses PRGA, akan di XOR-kan dengan nilai biner masing-masing plainteks, hingga dihasilkan cipherteks.

Karakter "**s**"
Cari Nilai *i*, *a* dan *j* :
Nilai *i* dan *j* dimulai dari 0
i = (*i* + 1) Mod 256 → (0+1) Mod 256 = 1
a = S*[i]* → S [*i*]= 4
j = (*j*+*a*) Mod 256 → (0+4) Mod 256 = 4
j = *j* → Nilai *j* selanjutnya adalah nilai *j* terakhir
Tukarkan nilai *i*, *a* dan *j*
b = S[*i*] →S [4] =213
S [*i*] = b →S [1] = 213
S [*j*] = a →S [4] = 4
Selanjutnya adalah menghitung nilai c dan z
Berdasarkan proses sebelumnya telah di tetapkan nilai *i* dan *j* adalah :
i = 1 dan j = 4
S [*i*] = b → *S* [1] = 213
S [*j*] = a → *S* [4] = 4
Cari nilai c dan z
c = (*S*[((i<<4) ⊕ (j>>3)) mod 256 + *S*[((*j*<<4) ⊕ (*i*>>3)) mod 256]) mod 256
= (*S*[((i<<4) ⊕ (j>>3)) mod 256 + *S*[((*j*<<4) ⊕ (*i*>>3)) mod 256]) mod 256
= (*S*[((1<<4) ⊕ (4>>3)) MOD 256) + *S*[((4<<4) ⊕ (1>>3)) mod 256]) mod 256
= (*S*[(16 ⊕ 0) mod 256] + *S* [(64 ⊕ 0) mod 256]) mod 256
= (*S*[16 mod 256] + *S* [64 mod 256]) mod 256

= (*S*[16] + S[64] mod 256
= (168 + 26) mod 256
= 194 mod 256
c = 194

Berdasarkan proses sebelumnya telah di tetapkan nilai *i* dan *j* adalah :
i = 1 dan *j* = 4
a = S [i] →S [1] = 4
j = (*j*+*a*) mod 256 → (0 + 4)mod 256 = 4
S [*i*] = *b* →*S* [1] = 213
S [*j*] = *a* →*S* [4] = 4
Nilai c = 194
Selanjutnya mencari nilai z :
z = ((*S*[*a*+*b*] mod 256+*S* [((c ⊕ 00AA) mod 256)]⊕*S*[((*j*+*b*) mod 256]) mod 256
=((*S*[4+213]mod 256+*S*[((194⊕170)mod 256)]⊕*S*[((4+213)mod 256])mod 256
= ((*S*[217 mod 256 + S [((194 ⊕ 170) mod 256)]) ⊕ S[217 mod 256]) mod 256
= ((*S*[217] + *S* [((104 mod 256)]) ⊕ *S* [217 mod256]) mod 256
= ((*S*[217] + *S*[104]) ⊕ *S* [217]) mod 256
= ((58 + 202) ⊕ 58) mod 256
= (260 ⊕ 58) mod 256
= 318 mod 256
z = 62
P[0] **L** = 76 → 01001100
K [0] = 62 → 00111110 ⊕
C [0] = → 01110010 = 114 dalam tabel ASCII merupakan karakter " **r** ".

Pada saat nilai *i* = 1, maka
j= (*j* + S[i] + *Key* [*i* mod *keylength*]) Mod 256
j=(213+ S [1] + *Key* [1 mod 4]) mod 256
j=(213+1+*Key* [1]) Mod 256
j=(213 + 1 + 97) Mod 256
j= 311 Mod 256
j= 55
Nilai S[*i*] Swap dengan Nilai S[*j*]
Nilai S [1] ditukar (swap) dengan Nilai S [55]
Maka :
S[1]= 55 dan S [55]=1

Karakter "**a**"
Cari Nilai *i*, *a* dan *j* :
Nilai *i* dan *j* dimulai dari 1
i = (*i* + 1) Mod 256 → (1+1) Mod 256 = 1
a = S*[i]* → S [*i*]= 4
j = (*j*+*a*) Mod 256 → (1+4) Mod 256 = 4
j = *j* → Nilai *j* selanjutnya adalah nilai *j* terakhir
Tukarkan nilai *i*, *a* dan *j*
b = S[*i*] →S [4] =55
S [*i*] = b →S [1] = 55
S [*j*] = a →S [4] = 4
Selanjutnya adalah menghitung nilai c dan z
Berdasarkan proses sebelumnya telah di tetapkan nilai *i* dan *j* adalah :
i = 1 dan j = 4
S [*i*] = b → *S* [1] = 55
S [*j*] = a → *S* [4] = 4
Cari nilai c dan z
c = (*S*[((i<<4) ⊕ (j>>3)) mod 256 + *S*[((*j*<<4) ⊕ (*i*>>3)) mod 256]) mod 256
= (*S*[((i<<4) ⊕ (j>>3)) mod 256 + *S*[((*j*<<4) ⊕ (*i*>>3)) mod 256]) mod 256
= (*S*[((1<<4) ⊕ (4>>3)) MOD 256) + *S*[((4<<4) ⊕ (1>>3)) mod 256]) mod 256
= (*S*[(16 ⊕ 0) mod 256] + *S* [(64 ⊕ 0) mod 256]) mod 256
= (*S*[16 mod 256] + *S* [64 mod 256]) mod 256

= (*S*[16] + S[64] mod 256
= (168 + 26) mod 256
= 194 mod 256
c = 194

Berdasarkan proses sebelumnya telah di tetapkan nilai *i* dan *j* adalah :
i = 1 dan *j* = 4
a = S [i] →S [1] = 4
j = (*j*+*a*) mod 256 → (0 + 4)mod 256 = 4
S [*i*] = *b* →*S* [1] = 55
S [*j*] = *a* →*S* [4] = 4
Nilai c = 194
Selanjutnya mencari nilai z :
z = ((*S*[*a*+*b*] mod 256+*S* [((c ⊕ 00AA) mod 256)]⊕*S*[((*j*+*b*) mod 256]) mod 256
=((*S*[4+55]mod 256+*S*[((194⊕170)mod 256)]⊕*S*[((4+55)mod 256])mod 256
= ((*S*[59 mod 256 + S [((194 ⊕ 170) mod 256)]) ⊕ S[59 mod 256]) mod 256
= ((*S*[59] + *S* [((104 mod 256)]) ⊕ *S* [59 mod256]) mod 256
= ((*S*[59] + *S*[104]) ⊕ *S* [59]) mod 256
= ((130 + 202) ⊕ 130) mod 256
= (332 ⊕ 130) mod 256
= 462 mod 256
z = 206
P[1] **I** = 73 → 01001001
K [1] = 206 → 11001110 ⊕
C [1] = 10000111 = 135 dalam tabel ASCII merupakan karakter " **ê** ".

Pada saat nilai *i* = 2, maka
j= (*j* + S[i] + *Key* [*i* mod *keylength*]) Mod 256
j=(55+ S [2] + *Key* [2 mod 4]) mod 256
j=(55+2+*Key* [2]) Mod 256
j=(55 + 2 + 121) Mod 256
j= 178 Mod 256
j= 178
Nilai S[*i*] Swap dengan Nilai S[*j*]
Nilai S [2] ditukar (swap) dengan Nilai S [178]
Maka :
S[2]= 178 dan S [178]=2

Karakter "**y**"
Cari Nilai *i*, *a* dan *j* :
i = (*i* + 2) Mod 256 → (1+1) Mod 256 = 2
a = S*[i]* → S [*i*]= 6
j = (*j*+*a*) Mod 256 → (2+4) Mod 256 = 6
j = *j* → Nilai *j* selanjutnya adalah nilai *j* terakhir
Tukarkan nilai *i*, *a* dan *j*
b = S[*i*] →S [6] =178
S [*i*] = b →S [2] = 178
S [*j*] = a →S [6] = 6
Selanjutnya adalah menghitung nilai c dan z
S [*i*] = b → *S* [2] = 178
S [*j*] = a → *S* [6] = 6
Cari nilai c dan z
c = (*S*[((i<<6) ⊕ (j>>3)) mod 256 + *S*[((*j*<<6) ⊕ (*i*>>3)) mod 256]) mod 256
= (*S*[((i<<6) ⊕ (j>>3)) mod 256 + *S*[((*j*<<6) ⊕ (*i*>>3)) mod 256]) mod 256
= (*S*[((2<<6) ⊕ (6>>3)) MOD 256) + *S*[((6<<6) ⊕ (2>>3)) mod 256]) mod 256
= (*S*[(128 ⊕ 0) mod 256] + *S* [(128 ⊕ 0) mod 256]) mod 256
= (*S*[128 mod 256] + *S* [128 mod 256]) mod 256
= (*S*[128] + S[128] mod 256
= (170 + 170) mod 256
= 340 mod 256
c = 84

ISSN 2774-3659 (Media Online)
Vol 1, No 3, Agustus 2021
Hal 83-92
https://hostjournals.com/bulletincsr

Berdasarkan proses sebelumnya telah di tetapkan nilai *i* dan *j* adalah :
i = 2 dan *j* = 6
a = S [i] →S [2] = 6
j = (*j*+*a*) mod 256 → (0 + 6)mod 256 = 6
S [*i*] = *b* →*S* [2] = 178
S [*j*] = *a* →*S* [6] = 6
Nilai c = 84
Selanjutnya mencari nilai z :
z = ((*S*[*a*+*b*] mod 256+*S* [((c ⊕ 00AA) mod 256)]⊕*S*[((*j*+*b*) mod 256]) mod 256
=((*S*[6+178]mod 256+*S*[((84⊕170)mod 256)]⊕*S*[((6+178)mod 256])mod 256
= ((*S*[184 mod 256 + S [((84 ⊕ 170) mod 256)]) ⊕ S[184 mod 256]) mod 256
= ((*S*[184] + *S* [((254 mod 256)]) ⊕ *S* [184 mod256]) mod 256
= ((*S*[184] + *S*[254]) ⊕ *S* [184]) mod 256
= ((184 + 254) ⊕ 184) mod 256
= (438 ⊕ 184) mod 256
= 270 mod 256
z = 14
P[2] **S** = 83 → 01010011
K [2] = 14 → 00001110 ⊕
C [2] = 01011101 = 93 dalam tabel ASCII merupakan karakter " **]** ".

Pada saat nilai *i* = 3, maka
j= (*j* + S[i] + *Key* [*i* mod *keylength*]) Mod 256
j=(178 + S [3] + *Key* [3 mod 4]) mod 256
j=(178+3+*Key* [3]) Mod 256
j=(178 + 3 + 97) Mod 256
j= 278 Mod 256
j= 22
Nilai S[*i*] Swap dengan Nilai S[*j*]
Nilai S [3] ditukar (swap) dengan Nilai S [22]
Maka :
S[3]= 22 dan S [22]=3

Karakter "**a**"
Cari Nilai *i*, *a* dan *j* :
i = (*i* + 3) Mod 256 → (1+3) Mod 256 = 3
a = S*[i]* → S [*i*]= 7
j = (*j*+*a*) Mod 256 → (3+4) Mod 256 = 7
j = *j* → Nilai *j* selanjutnya adalah nilai *j* terakhir
Tukarkan nilai *i*, *a* dan *j*
b = S[*i*] →S [7] =22
S [*i*] = b →S [7] = 22
S [*j*] = a →S [7] = 7
Selanjutnya adalah menghitung nilai c dan z
S [*i*] = b → *S* [3] = 22
S [*j*] = a → *S* [7] = 7
Cari nilai c dan z
c = (*S*[((i<<7) ⊕ (j>>3)) mod 256 + *S*[((*j*<<7) ⊕ (*i*>>3)) mod 256]) mod 256
= (*S*[((i<<7) ⊕ (j>>3)) mod 256 + *S*[((*j*<<7) ⊕ (*i*>>3)) mod 256]) mod 256
= (*S*[((3<<7) ⊕ (7>>3)) MOD 256) + *S*[((7<<7) ⊕ (3>>3)) mod 256]) mod 256
= (*S*[(128 ⊕ 0) mod 256] + *S* [(128 ⊕ 0) mod 256]) mod 256
= (*S*[128 mod 256] + *S* [128 mod 256]) mod 256
= (*S*[128] + S[128] mod 256
= (170 + 170) mod 256
= 340 mod 256
c = 84
Berdasarkan proses sebelumnya telah di tetapkan nilai *i* dan *j* adalah :
i = 3 dan *j* = 7
a = S [i] →S [3] = 7
j = (*j*+*a*) mod 256 → (0 + 7)mod 256 = 6

$S[i] = b \rightarrow S[3] = 22$
$S[j] = a \rightarrow S[7] = 7$
Nilai c = 84
Selanjutnya mencari nilai z :
$z = ((S[a+b] \bmod 256 + S[((c \oplus 00AA) \bmod 256)] \oplus S[((j+b) \bmod 256]) \bmod 256$
$= ((S[7+22] \bmod 256 + S[((84 \oplus 170) \bmod 256)] \oplus S[((7+22) \bmod 256]) \bmod 256$
$= ((S[29 \bmod 256 + S[((84 \oplus 170) \bmod 256)]) \oplus S[29 \bmod 256]) \bmod 256$
$= ((S[29] + S[((254 \bmod 256)]) \oplus S[29 \bmod 256]) \bmod 256$
$= ((S[29] + S[254]) \oplus S[29]) \bmod 256$
$= ((29 + 254) \oplus 29) \bmod 256$
$= (283 \oplus 29) \bmod 256$
$= 262 \bmod 256$
z = 6
P[3] **A** = 65 → 01000001
K [3] = 6 → 00000110 ⊕
C [3] = 01000111 = 71 dalam tabel ASCII merupakan karakter " **G** ".

Hasil *chiperteks* dari proses enkripsi dengan Plainteks "LISA" adalah 76, 73, 83, 65 dalam tabel ASCII yaitu : r ê] G = 62, 206, 14, 6

3.1 Hasil Pengujian Program

Sebagai contoh, hasil pengujian program sebagai berikut. Berikut pengujian program pada perbandingan enkripsi file teks (plainteks) menjadi chiperteks pada algoritma RC4+ dengan RC4A :

Tabel 5. Hasil pengujian

RC4+ File teks	Kunci	Plainteks	Chiperteks	Nilai MSE	Nilai PSNR
Algoritma.txt	lisa	Algoritma simetri adalah algoritma dimana penyandian kunci enkripsi dan kunci dekripsi bernilai sama. Kunci pada penyandian simetri diasumsikan bersifat rahasia dan hanya pihak yang melakukan enkripsi dan dekripsi yang mengetahui nilainya. Algoritma ini sering disebut dengan algoritma klasik karena memakai kunci yang sama untuk kegiatan enkripsi dan dekripsi	+ôü©6à%Sê£1þ"'»7'š—:iûÀ)ÚE¶ s””C• —Hœ □5,Ò¿>R———— ,¦_ºæþÐÈz0ÃA¸É8jjSçá1 àf'”¦KÉ• ¿———— 0$*Ç^0ð'iiÒ€•Ìù_³hV%x ŽWÉeôú-Ù'Ó?·wµ¢£Ì\vs ¡Ø>)‡gW6[{ÊaA@òÞ+¤ Ñ¿ágá• -[<=MË™/Ê˜$C õ*æìL«"¤…r²(1b*òZü:£[• f ï}_p#LáúŽ_À*f*óa4šìiŒ/ Â‰–í¦Ä.— ‡*f*KÞ÷Ãý¢AOA×Tªe@ÿ EfBp¥Ê†šrb¸z.'ýþü|Êü:³*f* ¤çÃý•##	85	16.382 dB
RC4A	Plainteks	Chiperteks	Nilai MSE	Nilai PSNR	
Kriptografi.txt	lisa	Kata *Cryptography* berasal dari bahasa Yunani yang terdiri dari dua kata yaitu kryptos yang berarti rahasia dan graphein yang berarti tulisan (Mollin, 2007). Menurut terminologinya, kriptografi adalah ilmu dan seni untuk menjaga keamanan pesan ketika pesan dikirim dari suatu tempat ke tempat lain	·ÊçÂŒ¬åÚÜÝâÈÞÊãÉå ‰ÕÆÞÊæÂØ‰×ÂÞÒ“ ÃÍÑÔÔÍ‰ÌÖÚÊáÊŒâÔ ÏÓ‰çÆÞÍÜÓÕ‰×ÂÞÒ “ÅáÊ“ÌÍÝÔ□åÊÜÕá‰Þ ÓåÙçÐß‰ìÂÚÐ“ÃÑÛ ÔÓàÒ“ÓÍÑÔÔÕÊ“ÅÍ×“ ÈÞÊãÉÑÒá□åÊáÈŒËØ ÓÍÛçÊŒÝèÍÕÜÔÏŒ‘À ÐØÕÜÏ˜‰¥‘œ œ□Œ¶Ø ÏáÛèÕŒÝØÓÙÒáÐØØ ÚÊÚâÔ□ŒÔåÊÜÝâÈÞ ÊÙÊŒÊ×ÂØÊÛ• ÕÕàÖ	120	13.654 dB

ŒÍÔÏŒÜØÏÕ‰èÏàÐÞ•
ÙÎáËÍÐÔ□×ÎÔÎÍ×ÔÏŒÙ
ØÔÍ×“ÌÑÝÜÌÍ

4. KESIMPULAN

Berdasarkan hasil studi literatur, analisis, perancangan, implementasi, dan pengujian sistem ini, maka kesimpulan yang didapat adalah Teknik yang dilakukan dalam pengamanan *file teks* dengan ekstensi (*.txt) yaitu dengan cara menerapkan Algoritma RC4+ dengan algoritma RC4A kedalamnya yang bisa mengubah data asli ke dalam bentuk data rahasia. Pada perbandingan file teks dienkripsi dengan menggunakan kode Algoritma *RC4+* menggunakan proses *key schedulling algorithm* dan proses *pseudo random generation algorithm* yang didapat nilai MSE 70 dan PSNR 16.274 sedangkan RC4A menggunakan *array* panjang kunci menjadi 256 byte *stream cipher* dengan nilai MSE 103 dan PSNR 13.794db.

REFERENCES

[1] Dani Iqbal, Asep Roy Panggabean, Indra Williamsyah Sinaga, Taronisokhi Zebua, 2019. Implementasi Algoritm RC4+ Untuk Mengamankan Pesan Teks Pada Aplikasi Chatting. (SAINTEKS), AMIK STIEKOM, Sumatera Utara
[2] Nur hayati, Mohammad Andri Budiman, Amer Sharif, 2017. Implementasi Algoritma RC4A dan MD5 Untuk Menjamin *Confidentiality* dan *Integrity* Pada File Teks. (Sinkron), Jurnal & Penelitian Teknik Informatika. POLITEKNIK GANESHA Medan.
[3] Shaw, M. E. dan Costanzo, P. R. 1982. *Theories of Social Psychology, Second Edition*. Tokyo: McGraw-Hill Kogakusha, Ltd.
[4] Munir, R., 2006, Kriptografi, Informatika, Bandung.
[5] Arius, D. 2008, Awal Sejarah Kriptografi Didunia, STMIK AMIKOM, Yogyakarta.
[6] Ariyus, Dony. & Andri Rum, 2008. KODE ASCII 7 BIT. Jurusan Teknik Informatika, STMIK AMIKOM Yogyakarta.
[7] Eko Aribowo, 2008. Pengamanan Document Office Dengan Algoritma Kriptografi Kunci Asimetris Elgamal. Jurnal Informatika, Vol.2 No.2 : Yogyakarta.
[8] Adi Nugroho, 2010, Rekayasa Perangkat Lunak menggunakan UML dan Java, penerbit ANDI : Yogyakarta
[9] Rossa A. S, M. Shalahuddin 2011, Rekayasa Perangkat Lunak Terstruktur dan Berorientasi Objek. Penerbit ANDI, Yogyakarta
[10] Priyanto, Rahmat, 2009, Langsung Bisa Visual Basic.Net 2008 ,Penerbit, ANDI, Yogyakarta.