Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) DOI: 10.25126/jtiik.201744474
Vol. 4, No. 4, Desember 2017, hlm. 275-282 p-ISSN: 2355-7699
Akreditasi KEMENRISTEKDIKTI, No. 51/E/KPT/2017 e-ISSN: 2528-6579

PENGAMANAN CITRA DIGITAL BERDASARKAN MODIFIKASI ALGORITMA RC4

Taronisokhi Zebua[1], Eferoni Ndruru[2]

[1]AMIK STIEKOM Sumatera Utara
[2]STMIK Budi Darma Medan
Email: [1]taronizeb@gmail.com, [2]ronindruru@gmail.com

(Naskah masuk: 9 Oktober 2017, diterima untuk diterbitkan: 24 Desember 2017)

Abstrak

Citra digital yang bersifat pribadi dan rahasia sangat rentang terhadap penyadapan oleh pihak-pihak lain, terutama bila citra tersebut didistribusikan melalui internet. Tindakan penyadapan dan penyalahgunaan terhadap citra yang sifatnya rahasia tentu saja dapat merugikan pihak pemilik citra. Salah satu teknik untuk meminimalkan tindakan tersebut di atas adalah pemanfaatan teknik kriptografi. Teknik kriptografi dapat mengamankan citra digital dengan cara modifikasi nilai-nilai *pixel* citra sehingga citra yang dihasilkan berbeda dengan citra asli. Algoritma RC4 dapat digunakan sebagai salah satu algoritma dalam mewujudkan tujuan teknik kriptografi. Namun algoritma ini memiliki kelemahan di mana pemecahan algoritma ini dengan *know plaintext attack* atau *know ciphertext only* dapat dilakukan dengan mudah. Penelitian ini menguraikan pengamanan citra digital berdasarkan modifikasi algoritma RC4. Modifikasi yang dilakukan adalah menambahkan sebuah blok *intial vector* pada proses enkripsi maupun dekripsi serta melakukan pemindahan sejumlah bit pada posisi tertentu. Hasil penelitian ini adalah citra digital dengan nilai-nilai *pixel* yang jauh berbeda dengan nilai *pixel* aslinya dengan tujuan dapat mempersulit pihak lain dalam memanipulasi citra rahasia.

Kata kunci: *kriptografi, citra, algoritma, RC4, cipher aliran.*

Abstract

Digital images of a private and confidential verry extend to wiretapping by other parties, especially when the image is distributed over the internet. The tapping or misuse of the confidential private image of course, harm the image owner. One technique to minimize the above measures is the utilization of cryptographic techniques. Cryptography techniques can secure digital images by modifying the image pixel values so that the resulting image is different from the original image. RC4 algorithm can be used in realizing the purpose of cryptographic technique. But this algorithm has a weakness where the solving of this algorithm with know plaintext attack or know ciphertext only can be done easily. This research describes the security of digital images based on RC4 algorithm modification. The modification is to add a initial vector block in the process of encryption and decryption and shift a number of bits in a certain position. The results of this research is a digital image with pixel values that are much different from the original pixel value to make it difficult for others in manipulating the secret image.

Keywords: *cryptography, image, algorithm, RC4, stream cipher.*

1. PENDAHULUAN

Citra digital merupakan salah satu jenis data yang saat ini banyak digunakan dalam berkomunikasi baik secara langsung, maupun melalui media internet. Perkembangan pemanfaatan media sosial saat ini sudah semakin memudahkan orang-orang untuk saling berkomunikasi, bertukar informasi atau bertukar pesan baik berjenis teks, citra, audio maupun video. Namun, pada sisi lain pekermbangan tersebut menyebabkan semakin mudahnya pihak-pihak tertentu untuk melakukan penyerangan dan penyalahgunaan terhadap data-data atau informasi yang didistribusikan seperti tindakan penyadapan informasi, pemantauan informasi, manipulasi informasi atau menggunakan informasi tersebut pada kepentingan tertentu.

Berdasarkan penelitian Setianingsih, dalam penyandian citra sangat diperlukan sistem pengamanan untuk melindungi data yang ditransmisikan melalui suatu jaringan komunikasi. Salah satu cara yang dapat dilakukan untuk mengamankan data adalah dengan teknik kriptografi (Setyaningsih E., 2009). Sedangkan penelitian lain oleh Zebua dalam menyandikan *record database* mengatakan bahwa informasi penting untuk di jaga agar tidak dapat di akses dan disalahgunakan oleh orang-orang yang tidak berhak (Zebua T., 2013).

Salah satu algoritma kriptografi yang umum digunakan adalah algoritma RC4 (*Rivest Cipher 4*), namun kelemahan algoritma ini adalah mudah di serang dengan teknik *know-plaintext attack* dan

ciphertext-only attack (Hendarsyah D. & Wardoyo R., 2011).

Modifikasi algoritma RC4 yang diuraikan dalam penelitian ini adalah menambahkan sebuah nilai *intialization vector* (inisialisasi awal) yang saling berantai pada setiap operasi XOR antara biner *pixel* citra dan biner kunci serta melakukan pemindahan sejumlah bit-bit *plain* pada posisi tertentu baik pada proses enkripsi maupun pada proses dekripsi.

2. CITRA DIGITAL

Istilah lain dari gambar adalah citra digital dan merupakan bagian dari komponen multimedia yang sangat penting peranannya dalam menghasilkan informasi visual (Munir R., 2004). Gambar atau citra digital kaya dengan informasi sehingga karakteristik inilah yang membedakannya dengan teks. Citra di bagi menjadi dua jenis yaitu citra kotinu dan citra diskrit. Citra yang dihasilkan melalui proses digitalisasi terhadap citra kontinu di sebut dengan citra diskrit atau citra digital (*digital image*).

Citra digital tediri dari tiga jenis yaitu citra *monocrome* (hitam putih/citra biner), citra *grayscale* (abu-abu) dan citra *true color* (citra berwarna). Setiap *pixel* citra berwarna memiliki tiga elemen warna yang diistilahkan dengan RGB, yaitu *Red* (merah), *Green* (hijau) dan *Blue* (biru).

3. KRIPTOGRAFI

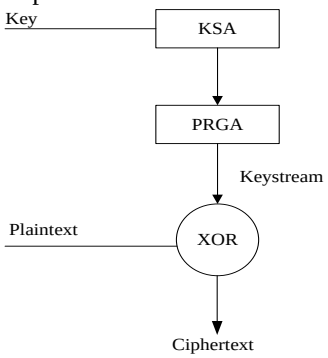
Teknik kriptografi merupakan salah satu teknik pengamanan data dengan melakukan proses penyandian terhadap data yang ingin diamankan sehingga makna asli dari data tidak lagi dapat dimengerti. Kriptografi merupakan salah satu teknik yang dapat digunakan dalam menjaga dan mengamankan informasi pada saat di distribusikan dari suatu tempat ke tempat lain (Ariyus D., 2008).

Adapun aspek atau tujuan yang harus dicapai dalam menerapkan teknik kriptografi adalah kerahasiaan, integritas, otentikasi dan *nir*-penyangkalan (Zebua T., 2015). Teknik kriptografi memiliki banyak algoritma dalam mencapai tujuan di atas, di antaranya algoritma *hill cipher*, *affine cipher*, DES, GOST, RC2, RC4 dan lainnya.

4. ALGORITMA RC4

Algoritma RC4 (*Riverst Cipher 4*) merupakan *stream cipher* yang dirancang di RSA *Security* oleh Ron *Rivest* tahun 1987. Sifat kunci dalam algoritma RC4 adalah simetris serta melakukan proses enkripsi *plain* per digit atau *byte* per *byte* dengan operasi biner (biasanya XOR) dengan sebuah angka semiacak. Namun algoritma ini memiliki kelemahan yaitu mudah diserang dengan teknik *know-plaintext attack* dan *ciphertext-only attack* (Setianingsih E., 2015). Serangan *know-plaintext attack* bisa diartikan jika kriptanalist memiliki potongan *plaintext dan ciphertext*, maka dengan mudah didapatkan aliran kunci dengan cara meng-XOR-kan *plaintext* dengan *ciphertext*.

Algortima RC4 bekerja dengan tiga tahap utama yaitu *Key Scheduling Algorithm (KSA), Pseudo Random Generation Algorithm (PRGA)* dan Proses Enkripsi dan Dekripsi (Setianingsih E., 2015) dan (Agung H. & Budiman, 2015). Proses di atas, dapat diperlihatkan pada Gambar 1 di bawah ini.



Gambar 1. Blok Diagaram Algoritma RC4

a. *Key Scheduling Algorithm* (KSA)

Proses KSA merupakan proses pembentukan tabel S-Box (Tabel Array S) dan Kunci (Tabel array [T]) yang di permutasi sebanyak 256 iterasi. *Pseudocode* untuk proses inisialisasi S-Box dan Array T:

```
for (i = 0 ; i<=255; i++){
    S-Box[i] = i
    T[i] = kunci[ i mod panjang_kunci]
}
```

Pseudocode untuk permutasi isi *array* S-Box :

```
j = 0
for (i = 0 ; i<=255; i++){
    j = (j + S-Box[i] + T[i]) mod 256
    Swap( S-Box[i], S[j] )
    j = j
}
```

setelah dua proses ini dilakukan, maka *array* S-Box dan *array* Kunci (T) telah terbentuk.

b. *Pseudo Random Generation Algorithm* (PRGA)

Tabel *array* S-Box akan digunakan pada proses ini untuk menghasilkan *key stream* yang jumlahnya sama dengan jumlah banyaknya karakter *plaintext* kemudian akan di-XOR dengan *plaintext*. Adapun *pseudocode* proses PRGA ini adalah :

```
i = 0; j = i
for (i = 0 ; i <= jlh_karakter_plaintext; i++){
    i = (i + 1) mod 256
    j = (j + S-Box[i]) mod 256
    Swap( S-Box[i], S-Box[j] )
    t = (S-Box[i] + S-Box[j]) mod 256
    Kunci[i] = S-Box[t]
}
```

c. Proses enkripsi atau dekripsi dengan operasi XOR. Proses enkripsi atau dekripsi di awali dengan merubah setiap nilai *plaintext* ke biner.

Formula untuk melakukan proses enkripsi dan dekripsi (Zebua T., 2015), adalah:
Formula proses enkripsi:

$$Ci = Pi \oplus Ki \quad (1)$$

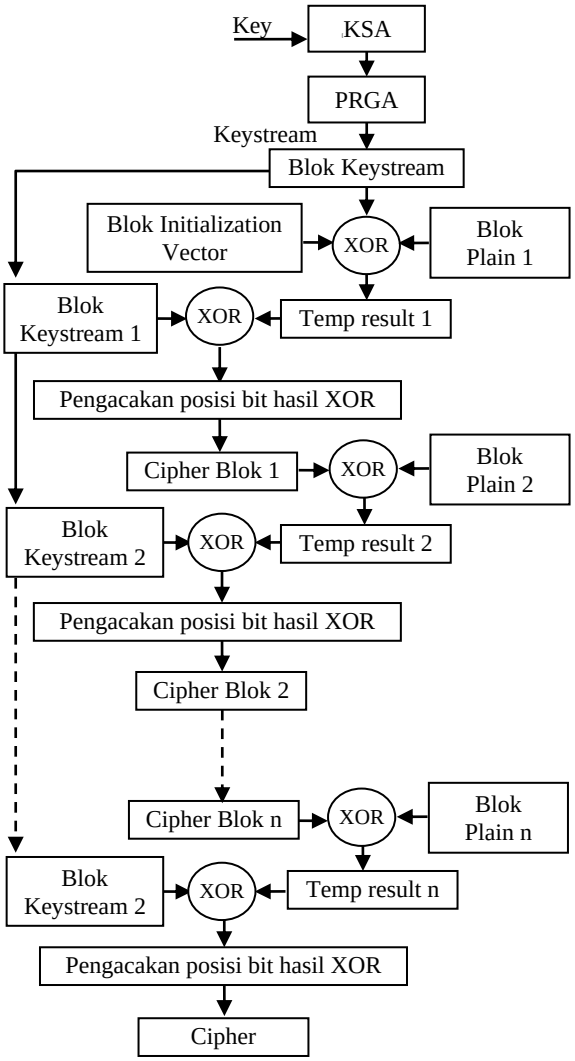
Formula proses dekripsi:

$$Pi = Ci \oplus Ki \quad (2)$$

5. PEMBAHASAN

Berdasarkan prosedur enkripsi dan dekripsi algoritma RC4, maka diketahui bahwa formulasi untuk melakukan proses enkripsi maupun dekripsi sangat sederhana, yaitu hanya dengan operasi XOR. Hal inilah yang menjadi salah satu kelemahan dari algoritma RC4 (Haji W. H. & Mulyono S., 2012).

Modifikasi pada algoritma ini adalah menambahkan sebuah nilai inisialisasi awal yang kemudian di-XOR-kan dengan masing-masing *plain* atau *cipher* secara berantai untuk mengoptimalkan ketahanannya dari teknik penyerangan. Hasil operasi inilah yang kemudian di-XOR-kan dengan kunci. *Cipher* yang dihasilkan merupakan rangkaian bit yang telah teracak. Diagram proses modifikasi RC4 ditunjukkan pada Gambar 2.



Gambar 2. Diagram Proses Modifikasi Algoritma RC4

Berdasarkan Gambar 2, dapat di ketahui bahwa setiap proses yang dilakukan untuk menghasilkan *cipher* berikutnya bergantung dari *cipher* sebelumnya.

Formulai enkripsi menjadi:

$$C[i] = (P[i] \oplus C[i-1]) \oplus K[i+1] \quad (3)$$

Formulai dekripsi menjadi :

$$P[i] = (C[i] \oplus C[i-1]) \oplus K[i-1] \quad (4)$$

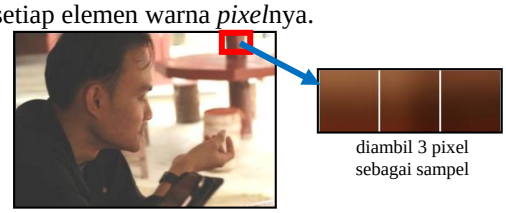
di mana:
P[i] adalah biner setiap blok plain
C[i] adalah biner cipher setelah dikembalikan bitnya
C[i-1] biner cipher sebelum dikembalikan bitnya (*cipher* asli)
C[0] adalah nilai *Initialization Vector* (IV)
K[i+1] dan K[i-1] adalah kunci

Proses dekripsi berdasarkan modifikasi algoritma RC4 ini di lakukan dengan cara yang sama seperti pada diagram proses enkripsi di atas.

Berikut ini diuraikan contoh penerapan algoritma RC4 yang telah di modifikasi dalam menyandikan sebuah citra berwarna berekstensi *bmp* dengan resolusi 187 x 314 dan *bitdepth* adalah 24 bit dengan kunci yang digunakan adalah ZEBUA dan biner *Inisialization Vector* adalah 01010100.

Gambar 4. *Plainimage* dengan Resolusi 187 x 314

Berdasarkan *plainimage* di atas, akan diambil 3 *pixel* sebagai sampel dalam perhitungan manual. Tiga *pixel* tersebut akan di ambil nilai desimal warna pada setiap elemen warna *pixel*nya.



Gambar 5. *Plainimage* Sampel sebanyak 3 *Pixel*

Nilai elemen warna dari tiga *pixel plainimage* sampel di atas di ambil dengan menggunakan *software* matlab, sehingga diperoleh:

Tabel 1. Nilai RGB Citra Sampel

Pixel 1			Pixel 2			Pixel 3		
R	G	B	R	G	B	R	G	B
154	96	60	113	59	33	71	26	28

Berdasarkan Tabel 1 di atas, maka nilai desimal *plainimage* adalah 154,96,60,133, 59, 33, 71, 26, 28.

5.1 Proses Enkripsi

Proses enkripsi berdasarkan modifikasi algoritma RC4, meliputi:

a. Proses *Key Scheduling Algorithm* (KSA)

Pembentukan Tabel S-Box, dilakukan berdasaran pseudocode-nya dan menghasilkan array dengan nilai 0 sampai dengan 255, sehingga tabel S-Box :

Tabel 2. Tabel S-Box

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
:::	:::	:::	:::	:::	:::	:::	:::	:::	:::	:::	:::	:::	:::	:::	:::
240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

Karakter kunci di buat dalam bentuk array :

Index0	Index1	Index2	Index3	Index4
Z	E	B	U	A

Kemudian bentuklah *array* Tabel T sesuai dengan *pseudocode*-nya :

untuk i = 0
T[0] = Kunci[0 mod 5]
T[0] = Kunci[0] ➔ Z (dec 90)

untuk i = 1
T[1] = Kunci[1 mod 5]
T[1] = Kunci[1] ➔ E (dec 69)

proses ini dilakukan hingga nilai i = 255
untuk i = 255
T[255] = Kunci[255 mod 5]
T[255] = Kunci[0] ➔ Z (dec 90)

Sehingga dihasilkan nilai tabel T keseluruhan adalah :

Tabel 3 : Tabel Array T

90	69	66	85	65	90	69	66	85	65	90	69	66	85	65	90
69	66	85	65	90	69	66	85	65	90	69	66	85	65	90	69
:::	:::	:::	:::	:::	:::	:::	:::	:::	:::	:::	:::	:::	:::	:::	:::
90	69	66	85	65	90	69	66	85	65	90	69	66	85	65	90

Setelah tabel *array* S-Box dan Array T di dapatkan, maka langkah selanjutnya adalah melakukan permutasi nilai-nilai tabel *array* S-Box. Permutasi dilakukan sebanyak 256 kali iterasi dengan menukarkan nilai S-Box dalam *array* i dengan nilai S-Box pada *array* j.

untuk nilai i = 0; j= 0
j = (0 + S-Box[0] + T[0]) mod 256
j = (0 + 0 + 90) mod 256
j = 90
Swap(S-Box[0], S-Box[90])
j = 90, nilai j berikutnya adalah 90

Tukarkan nilai tabel S-Box pada *array* 0 dengan nilai tabel S-Box pada array 90 dan begitu sebaliknya.

Berdasarkan tabel 2, maka nilai S-Box[0] = 0 dan nilai S-Box[90] = 90. Jadi setelah ditukarkan, maka nilai S-Box[0] = 90 dan S-Box[90] = 0.

untuk nilai i = 1; j= 90 (nilai akhir j pada iterasi i=0)
j = (90 + S-Box[1] + T[1]) mod 256
j = (90 + 1 + 69) mod 256
j = 160 mod 256 = 160
Swap(S-Box[1], S-Box[160])
j = 90, nilai j berikutnya adalah 90

Tukarkan nilai tabel S-Box pada *array* 1 (nilai i) dengan nilai tabel S-Box pada *array* 160 (nilai j) dan begitu sebaliknya.

Berdasarkan Tabel 2 setelah proses permutasi sebelumnya, maka nilai S-Box[1] = 1 dan nilai S-Box[160] = 160. Jadi setelah ditukarkan, maka nilai S-Box[1] = 160 dan S-Box[160] = 1.

Pada proses permutasi ini, nilai tabel S-Box yang digunakan pada setiap proses *swap* (penukaran) adalah nilai S-Box setelah dipermutasikan.

Proses permutasi ini dilakukan hingga nilai i = 255, sehingga proses ini dapat menyebabkan nilai array S-Box dapat di tukar secara berulang atau lebih dari satu kali. Hasil proses permutasi Tabel S-Box keseluruhan adalah:

Tabel 4. Hasil Permutasi S-Box

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

b. *Pseudo Random Generation Algorithm* (PRGA)

Proses PRGA menggunakan hasil permutasi S-Box pada Tabel 4 di atas. Proses ini dilakukan untuk menghasilkan *key stream* yang akan digunakan pada proses enkripsi ataupun dekripsi.

Proses iterasi pada PRGA dilakukan sebanyak jumlah nilai-nilai elemen warna citra. Bila citra sampel yang digunakan di atas memiliki 9 nilai, maka iterasi proses PRGA akan di lakukan sebanyak 9 kali.

Pada Proses PRGA menyebabkan tabel permutasi S-Box akan dipermutasikan kembali sebanyak iterasi yang dilakukan pada proses ini.

untuk iterasi 1 → i = 0; j = i
i = (0 + 1) mod 256 = 1
j = (0 + S-Box[1]) mod 256
j = (0 + 90) mod 256
j = 90
Swap (S-Box[1], S-Box[90])

Berdasarkan tabel hasil permutasi S-Box (tabel 4), maka nilai S-Box[1] = 160; S-Box[90] = 116
setelah ditukarkan menjadi :
S-Box[0] = 116; S-Box[90] = 160

t = (S-Box[1] + S-Box[90]) mod 256
t = (116 + 160) mod 256
t = 276 mod 256
t = 20
Kunci[0] = S-Box[20] → 154 (char š)

untuk iterasi 2 → i = 1; j = 1
i = (1 + 1) mod 256 = 2
j = (1 + S-Box[2]) mod 256
j = (1 + 239) mod 256
j = 240
Swap (S-Box[2], S-Box[240])

Berdasarkan tabel S-Box (setelah permutasi iterasi 2 di atas), maka : nilai S-Box[2] = 239 dan S-Box[240] = 99
setelah ditukarkan menjadi :
S-Box[2] = 99; S-Box[239] = 239

t = (S-Box[2] + S-Box[239]) mod 256
t = (99 + 239) mod 256
t = 338 mod 256; t = 82
Kunci[1] = S-Box[82] → 131 (char *ƒ*)

Iterasi selanjutnya di cari berdasarkan cara yang sama seperti di atas, sehingga di peroleh hasil dari *key stream* seperti ditunjukkan pada Tabel 5.

Tabel 5. Hasil Pseudorandom Key

Kunci	Desimal	Char
K[0]	154	Š
K[1]	131	Ƒ
K[2]	107	×
K[3]	31	
K[4]	151	—
K[5]	204	Ì
K[6]	141	R
K[7]	167	§
K[8]	123	{

key stream yang dihasilkan dari proses PRGA di atas akan digunakan sebagai kunci pada proses enkripsi maupun dekripsi.

Kunci Awal	Char	Z	E	B	U	A
	Dec	90	69	66	85	65

Key Stream	Char	š	*ƒ*	×		—	Ì	r	§	{
	Dec	154	131	107	31	151	204	141	167	123

Berdasarkan jumlah *key stream* yang dihasilkan di atas, maka disimpulkan bahwa jumlah *key stream* yang dibangkitkan berbanding lurus atau sama dengan jumlah banyaknya *pixel* citra digital yang akan disandikan. Hal ini terjadi karena RC4 mengenkripsi setiap *pixel* citra dengan kunci yang berbeda. Konsep ini hampir sama dengan konsep algoritma *vegeneere cipher*.

c. Proses Enkripsi

Proses enkripsi di awali dengan mengkonversi nilai-nilai warna elemen pixel citra ke bilangan biner. Hal yang sama juga dilakukan untuk karakter *key stream*.

Tabel 6. Hasil konversi nilai pixel citra ke biner

Pixel	Warna	Dec	Biner	P[i]
1	R	154	10011010	P[1]
	G	96	01100000	P[2]
	B	60	00111100	P[3]
2	R	133	10000101	P[4]
	G	59	00111011	P[5]
	B	33	00100001	P[6]
3	R	71	01000111	P[7]
	G	26	00011010	P[8]
	B	28	00011100	P[9]

Tabel 7. Hasil Konversi Key Stream Ke Biner

Key	Char	Dec	Biner
K[0]	Š	154	10011010
K[1]	Ƒ	131	10000011
K[2]	×	107	01101011
K[3]		31	00011111
K[4]	—	151	10010111
K[5]	Ì	204	11001100
K[6]	R	141	10001101
K[7]	§	167	10100111
K[8]	{	123	01111011

Selanjutnya melakukan proses enkripsi berdasarkan Persamaan (3).

Nilai bit Inisialization Vector (IV) atau C0 : 01010100

Karena jumlah bit IV adalah 8 bit, maka biner-biner *plainimage* dikelompokkan menjadi 8 bit (seperti kolom Pi pada Tabel 6).

Enkripsi Blok P1 : i = 1

C1 = (P[1] ⊕ C[1-1]) ⊕ K[1-1]
= (P[1] ⊕ C[0]) ⊕ K[0]
= (10011010 ⊕ 01010100)⊕10011010
= 11001110 ⊕ 10011010
= 01010100

Shift 2 bit dari kiri ke kanan (pemindahan 2 bit kiri ke kanan)

C1 = 01010001 (dec **81**)

Enkripsi Blok P2 : i = 2

C2 = (P[2] ⊕ C[2-1]) ⊕ K[2-1]
= (P[2] ⊕ C[1]) ⊕ K[1]
= (01100000 ⊕ 01010001)⊕ 10000011
= 00110001 ⊕ 10000011
= 10110010

Shift 2 bit dari kiri ke kanan (pemindahan 2 bit kiri ke kanan)

C2 = 11001010 (dec **202**)

Proses yang sama dilakukan untuk mencari *Cipher* dari blok *plainimage* lainnya, sehingga *cipher* seluruhnya seperti pada Tabel 8.

Tabel 8. Nilai Desimal Hasil Enkripsi

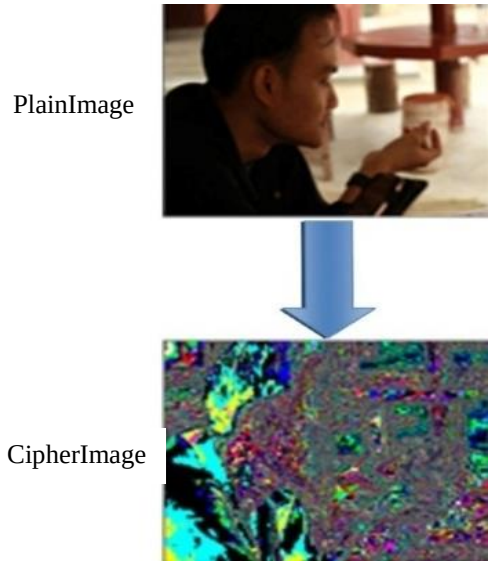
Block Ci	Nilai Desimal
C1	81
C2	202
C3	118
C4	179
C5	124
C6	70
C7	50
C8	62
C9	101

Sehingga, nilai warna citra asli (*plainimage*) setelah di enkripsi dunjukkan pada Tabel 9.

Tabel 9. Nilai Pixel CipherImage

Pixel	Warna	Plain	Cipher
		Dec	Dec
1	R	154	81
	G	96	202
	B	60	118
2	R	133	179
	G	59	124
	B	33	70
3	R	71	50
	G	26	62
	B	28	101

Sehingga, warna citra yang dihasilkan akibat perubahan nilai setiap elemen warna *pixel plainimage* adalah:



Gambar 6. Tampilan CipherImage

5.2 Proses Dekripsi

Proses dekripsi di awali dengan proses KSA dan proses PRNG. Prosesnya sama dengan yang dilakukan pada proses enkripsi. Sehingga *key stream* yang dihasilkan sama seperti *key sream* pada proses enkripsi (tabel 5).

Sebelum proses dekripsi dilakukan, *cipherimage* dan nilai-nilai karakter kunci sebagai input harus dikonversi menjadi biner, kemudian dilakukan pengelompokkan biner *cipher* sepanjang jumlah bit *initialization vector* (IV/C0).

Selanjutnya, dilakukan pengembalian 2 bit kanan ke posisi kiri pada setiap kelompok *cipherimage*. Hal dilakukan agar posisi-posisi bit yang telah di acak kembali pada posisi semula.

Tahap terakhir adalah melakukan proses dekripsi berdasarkan Persamaan (4).

Tabel 10. Nilai Desimal Pixel Cipher Image

Pixel	Warna	Decimal Pixel Cipher	BinerPixel Cipher	Ci
1	R	81	01010001	C[1]
	G	202	11001010	C[2]
	B	118	01110110	C[3]
2	R	179	10110011	C[4]
	G	124	01111100	C[5]
	B	70	01000110	C[6]
3	R	50	00110010	C[7]
	G	62	00111110	C[8]
	B	101	01100101	C[9]

Nilai *key stream* sama dengan nilai pada Tabel 7. IV/C0 sama dengan nilai yang IV yang digunakan pada proses enkripsi yaitu 01010100. 2 bit di posisi kanan pada masing-masing kelompok *cipherimage* dikembalikan ke posisi kiri, sehingga di peroleh hasilnya seperti pada Tabel 11.

Tabel 11. Hasil Pengembalian Bit CipherImage

Ci	BinerPixel Cipher Sebelum dikembalikan	BinerPixel Cipher Setelah dikembalikan
C[1]	01010001	01010100
C[2]	11001010	10110010
C[3]	01110110	10011101
C[4]	10110011	11101100
C[5]	01111100	00011111
C[6]	01000110	10010001
C[7]	00110010	10001100
C[8]	00111110	10001111
C[9]	01100101	01011001

untuk mencari *PlainImage* Blok 1 : i = 1

P[1] = (C[1] ⊕ C[1-1]) ⊕ K[1-1]
= (C[1] ⊕ C[0]) ⊕ K[0]
= (01010100 ⊕ 01010100) ⊕ 10011010
= 00000101 ⊕ 10011010
P[1] = 10011010 (decimal 154)

Untuk mencari PlainImage Blok 2 : i = 2

P[2] = (C[2] ⊕ C[2-1]) ⊕ K[2-1]
= (C[2] ⊕ C[1]) ⊕ K[1]
= (10110010 ⊕ 01010001) ⊕ 10000011
= 11100011 ⊕ 10000011
P[2] = 01100000 (decimal 96)

Untuk mencari PlainImage Blok 3 : i = 3

P[3] = (C[3] ⊕ C[3-1]) ⊕ K[3-1]
= (C[3] ⊕ C[2]) ⊕ K[2]
= (10011101 ⊕ 11001010) ⊕ 01101011
= 01010111 ⊕ 01101011
P[3] = 00111100 (decimal 60)

untuk mendapatkan nilai pixel *plainimage* dari blok *cipherimage* lainnya dilakukan dengan cara yang sama seperti di atas. Nilai yang dihasilkan sama seperti nilai setiap elemen warna *pixel* citra asli.

Tabel 12. Nilai Pixel Citra Hasil Dekripsi

Pixel	Warna	Plain Dec	Blok PlainImage
1	R	154	P[1]
	G	96	P[2]
	B	60	P[3]
2	R	133	P[4]
	G	59	P[5]
	B	33	P[6]
3	R	71	P[7]
	G	26	P[8]
	B	28	P[9]

Nilai-nilai elemen warna *pixel* citra tersebut di atas dipetakan kembali menjadi citra yang baru sehingga menghasilkan citra yang sama seperti citra aslinya (*plainimage*).

5.3 Ketahanan Terhadap Know Plain dan Know Cipher Attack

Jenis serangan *know plain attack* maupun *know cipher attack* dapat dilakukan dengan mengetahui potongan-potongan dari *plain* dan *cipher* kemudian kedua potongan tersebut di-XOR-kan untuk mendapatkan aliran kunci yang digunakan.

Bila seorang kriptanalis berhasil mendapatkan potongan biner *plain* dan *cipher* data yang telah di enkripsi berdasarkan algoritma RC4 tanpa modifikasi, kemudian dilakukan operasi XOR.

potongan biner *plain* = 10011001
potongan biner *cipher* = 11001001 ⊕
maka, kunci adalah = *01010000*

Bila potongan biner *plain* maupun *cipher* di-XOR-kan dengan kunci yang telah ditemukan, maka *cipher* atau plain adalah benar.

potongan biner *plain* = 10011001
potongan biner *kunci* = 01010000 ⊕
maka, cipher adalah = *11001001*

hasil operasi XOR yang didapatkan sama dengan biner *cipher*.

Ketahanan data yang telah di enkripsi berdasarkan modifikasi algoritma RC4 terhadap serangan jenis *know plain* dan *know cipher* terletak pada rumitnya untuk menemukan aliran kunci yang digunakan baik pada proses enkripsi maupun dekripsi. Kekuatan RC4 termodifikasi terletak pada operasi XOR yang tidak hanya dilakukan dengan kunci, namun dilakukan pada blok biner inisialisasi serta adanya proses pemindahan sejumlah bit dari posisi kiri ke kanan untuk mengacak posisi-posisi biner hasil operasi XOR dan proses ini dilakukan secara berantai pada setiap blok *plain* maupun *cipher*.

6. KESIMPULAN

Berdasarkan pembahasan di atas, maka disimpulkan bahwa :

1. Penambahan blok *initialization vector* dan pergeseran bit pada proses enkripsi maupun dekripsi pada modifikasi algoritma RC4 sangat efektif untuk mempersulit pihak penyerang baik dengan cara *know plain attack* maupun *cipher-only attack*.
2. Modifikasi yang dilakukan pada algoritma RC4 ini sangat efektif dalam mengaburkan pola dan warna citra asli sehingga *cipherimage* yang dihasilkan sangat berbeda dengan *plainimage*.
3. Pengubahan *size cipherimage* sangat berpengaruh terhadap gagalnya proses dekripsi.

7. DAFTAR PUSTAKA

SETYANINGSIH, E. 2009. Penyandian Citra Menggunakan Metode Playfair Cipher, J. Teknol., vol. 2, no. 2, pp. 213–219.

ZEBUA, T. 2013. Analisa dan Implementasi Algoritma Triangle Chain pada Penyandian Record Database. Pelita Inform. Budi Darma, vol. 3, no. 2, pp. 37–49.

HENDARSYAH, D. & WARDOYO, R. 2011. Implementasi Protokol Diffie-Hellman dan Algoritma RC4 untuk Keamanan Pesan SMS. IJCCS, vol. 5, no. 1, pp. 14–25.

MUNIR, R. 2004. Pengolahan Citra Digital dengan Pendekatan Algoritmik. Bandung: Informatika.

ARIYUS, D. 2008. Pengantar Ilmu Kriptografi, Yogyakarta: Andi.

ZEBUA, T. 2015. Penerapan Metode LSB-2 untuk Menyembunyikan Ciphertext pada Citra Digital. Pelita Inform. Budi Darma, vol. 10, no. 3, pp. 135–140.

SETIANINGSIH, E. 2015. Kriptografi dan Implementasi Menggunakan Matlab, Yagyakarta: Andi.

AGUNG, H. & BUDIMAN. 2015. Implementasi Affine Chiper dan RC4 Pada Enkripsi File Tunggal. Prosiding SNATIF, pp. 243–250.

HAJI, W. H. & MULYONO, S. 2012. Implementasi Rc4 Stream Cipher Untuk Keamanan Basis Data. Seminar Nasional Aplikasi Teknologi Informasi (SNATI). pp. 15–16.