Jurnal Sains Komputer & Informatika (J-SAKTI)
Volume (2) No.1 Maret 2018, pp. 12-22
ISSN:2548-9771/EISSN:2549-7200
http://tunasbangsa.ac.id/ejurnal/index.php/jsakti

Optimalisasi *Beaufort Cipher* Menggunakan Pembangkit Kunci RC4 Dalam Penyandian SMS

Mia Diana[1], Taronisokhi Zebua[2]
[1]Prorgram Studi Teknik Informatika STMIK Budidarma Medan
[2]Program Studi Teknik Informatika AMIK STIEKOM Sumatera Utara
[1]Jl. SM. Raja No. 338 Sp. Limun Medan
[2]Jl. H. Adam Malik No. 18 Rantauparapat, Labuhan Batu, Sumatera Utara
[1]miadiana@gmail.com, [2]taronizeb@gmail.com

Abstract

The key in the cryptography algorithm is a very important role in the process of encryption and decryption. The more random key numbers are used, the more random the ciphers are generated. The RC4 algorithm and the Beaufort cipher are algorithms of cryptographic techniques. The RC4 algorithm has advantages in generating random keys, whereas Beaufort ciphers have a disadvantage in terms of the number of keys used too much. This research describes how to optimize key formation in Beaufort algorithm by utilizing the key generation process on the RC4 algorithm which is implemented on encoding SMS which until now has not been point-to-point (not directly sent to destination). The results of this study provide convenience for users in the process of generating encryption and decryption keys and generate SMS cipher more random and difficult to understand by others.

Keywords : *SMS, Cryptography, Stream Cipher, RC4, Beaufort Cipher*

Abstrak

Kunci pada algoritma kriptografi sangat penting peranannya dalam proses enkripsi dan dekripsi. Semakin acak bilangan kunci yang digunakan, maka semakin acak pula cipher yang dihasilkan. Algoritma RC4 dan beaufort cipher merupakan algoritma dari teknik kriptografi. Algoritma RC4 memiliki kelebihan dalam membangkitkan kunci yang acak, sedangkan beaufort cipher memiliki kelemahan dalam hal jumlah kunci yang digunakan terlalu banyak. Penelitian ini menguraikan bagaimana mengoptimalkan pembentukan kunci pada algoritma beaufort dengan memanfaatkan proses pembangkitan kunci pada algoritma RC4 yang diimplementasikan pada menyandikan SMS yang sampai saat ini belum bersifat point-to-point (tidak langsung dikirim kepada tujuan). Hasil penelitian ini memberikan kemudahan bagi pengguna dalam proses pembangkitan kunci enkripsi maupun dekripsi serta menghasilkan cipher SMS yang lebih acak dan sulit dipahami oleh pihak lain.

Kata Kunci*: SMS, Kriptografi, Cipher Aliran, RC4, Beaufort Chiper*

1. PENDAHULUAN

Kriptografi merupakan salah satu teknik yang dapat digunakan dalam mengamankan data yang bersifat rahasia atau pribadi. Proses transformasi informasi yang berlangsung dua arah yang terdiri dari proses enkripsi dan dekripsi adalah ruanglingkup dari kriptografi[1]. Saat ini pemanfaatan teknik kriptografi pada bidang komunikasi sangat berkembang. Berdasarkan penelitian sebelumnya yang dilakukan oleh Setyaningsih mengatakan bahwa teknik kriptografi sangat penting diimplementasikan untuk melindungi data yang

ditransmisikan melalui suatu jaringan komunikasi[2]. Algoritma RC4 dan *beaufort cipher* merupakan contoh dari algoritma kriptografi.

Peranan kunci pada algoritma kriptografi sangat penting dalam melakukan proses enkripsi dan dekripsi data. Bilangan kunci yang acak akan menghasilkan cipher yang acak pula sehingga dapat menambah kerumitan bagi penyerang untuk memecahkan algoritma yang digunakan. Keamanan suatu pesan tidak tergantung pada algoritma yang digunakan dalam menyandikannya, namun tergantung pada kunci yang digunakan tanpa mengandung pola dan harus acak[1][2]. Oleh karena itu, maka pemlihan dan pemrosesan pembangkitan kunci dalam proses enkripsi dan dekripsi harus lebih baik.

Algoritma RC4 membangkitkan deretan kunci yang sangat acak, sehingga sangat sulit dipecahkan oleh pihak lain[3][4]. *Beaufort cipher* merupakan salah satu dari jenis kriptografi klasik dan merupakan varian algoritma *vigeneere cipher*. *Beaufort cipher* menggunakan kunci yang jumlah sama dengan jumlah *plain*[1]. Artinya bahwa bila pengguna memiliki panjang *plaintext* 200 karakter, berarti kunci yang digunakan harus 200 karakter. Hal inilah yang menjadi salah satu kelemahan dari algoritma *beaufort cipher*, karena akan sangat sulit bagi pengguna untuk menghafal kunci dengan panjang yang sama seperti *plaintext*.

Short Message Service (SMS) merupakan salah satu fasilitas yang disediakan oleh ponsel dan berfungsi sebagai salah satu media untuk berkomunikasi khususnya untuk melakukan pengiriman dan penerimaan data berupa pesan singkat. Hingga saat ini, pemanfaatan layanan ini sudah sangat berkembang di kalangan masyarakat. Namun layanan komunikasi dengan media SMS saat ini masih harus melewati pusat penyedia layanan dan belum bersifat *point-to-point*, sehingga hal ini dapat memungkinkan terjadinya penyadapan SMS yang sedang berada dalam proses distribusi[5]. Penelitian yang dilakukan oleh Purwaningsih mengatakan bahwa komunikasi via SMS memiliki celah keamanan terbesar dimana pesan yang dikirimkan akan disimpan di SMSC (*Short Message Service Center*) sebelum dikirim ke tujuan[6]. Tentu hal ini akan lebih berdampak negatif bila informasi atau pesan yang didistribusikan melalui SMS adalah pesan rahasia atau penting.

Penelitian ini menguraikan bagaimana mengoptimalkan proses pembangkitan kunci pada *beaufort cipher* dengan memanfaatkan proses pembangkitan kunci pada algoritma RC4 yang diimplementasikan pada penyandian teks SMS. Kunci yang dihasilkan akan dimanfaatkan sebagai kunci dalam proses enkripsi dan dekripsi teks SMS berdasarkan *beaufort cipher*, sehingga diperoleh *cipher* SMS yang lebih acak tidak lagi mudah dipahami.

2. METODOLOGI PENELITIAN

2.1 Short Message Service (SMS)

Fitur *Short Message Service* (SMS) merupakan salah satu aplikasi yang dikembangkan dan disediakan pada *handphone* sebagai salah satu fasilitas untuk bertukar pesan atau informasi[6]. Salah satu faktor perkembangan penggunaan SMS ini sebagai media berkomunikasi adalah layanan kemudahan serta biaya pengiriman SMS yang relatif murah. SMS yang dikirimkan melalui layanan SMS saat ini tidak langsung dikirimkan kepada penerima yang dituju, namun harus

melewati *Short Message Service Center* (SMSC) yang bertugas untuk mencatat komunikasi yang terjadi antara pengirim dan penerima[7]. Setelah SMS tercatat pada SMSC, barulah SMS tersebut diteruskan kepada penerima yang dituju.

2.2 Kriptografi

Kriptografi merupakan studi terhadap teknik matematis yang terkait dengan aspek keamanan suatu sistem informasi seperti kerahasiaan, integritas data, autentikasi dan ketiadaan penyangkalan[8]. Teknik kriptografi memiliki berbagai algoritma yang digunakan dalam mengamankan data, misalnya *Caesar Cipher, Vigeneere Cipher, Affine Cipher,* DES, RC4, RC6, AES, GOST, dan lain-lain. Hal yang harus dicapai dalam penerapan algoritma kriptografi adalah *confusion* (konfusi/pembingungan) yaitu harus mampu mempersulit pihak lain dalam merekonstruksi ulang *cipher* yang dihasilkan serta *diffusion* (peleburan) yaitu harus mampu menyembunyikan pola dari pesan asli[9].

Pengamanan data berdasarkan algoritma teknik kriptografi dilakukan dengan merubah pesan yang akan dirahasiakan *plaintext* menjadi sandi (*ciphertext*). Proses untuk mengkonversi *plaintext* menjadi *ciphertext* disebut dengan proses enkripsi sedangkan proses yang dilakukan untuk mengembalikan *ciphertext* menjadi *plaintext* disebut deskripsi[8][10]. Proses enkripsi dan dekripsi memerlukan sebuah kode dalam pelaksanaannya yang disebut dengan kunci. Kunci harus harus bersifat rahasia dan tidak boleh diberitahukan kepada orang lain yang tidak berhak untuk menerima pesan.

2.3 Algoritma RC4

Algoritma RC4 ditemukan pada tahun 1978 oleh Ronald Rivest dan merupakan pengembangan dari algoritma RC sebelumnya. Panjang kunci yang digunakan oleh RC4 adalah 1 *byte* hingga 256 *byte* dan digunakan untuk menginisialisasikan tabel sepanjang 256 *byte*[5]. Sebagai *stream cipher* (*cipher* aliran), maka algoritma RC4 melakukan proses enkripsi dan dekripsi secara satu persatu berdasarkan kunci yang telah dibangkitkan sebelumnya. Ada tiga proses utama pada algoritma RC4, yaitu proses *Key Schedulling Algorithm* (KSA), *Pseudo Random Generation Algorithm* (PRGA) dan proses enkripsi maupun dekripsi[3]. Kunci yang dibangkitkan berdasarkan algoritma RC4 sangat acak, namun kelemahan algoritma ini adalah mudahnya diserang oleh para kriptanalis dengan jenis *know-plain attack* maupun *cipher-only attack* [2][3] karena proses enkripsi dan dekripsinya yang sanngat sederhana.

2.3.1 Key Schedulling Algorithm (KSA)

Proses penjadwalan kunci (*key Schedulling*) dilakukan dengan tujuan membangkitkan kunci yang acak sejumlah 256 buah kunci. Penjadwalan kunci melibatkan dua tabel array yaitu *array* S dan *array* T. Proses pengacakan dilakukan dengan menukarkan nilai-nilai *array* S yang sebelumnya dikalkulasikan dengan nilai-nilai array T. Adapun *pseudocode* untuk melakukan pembentukan *array* S dan T adalah :

```
for (i = 0 ; i<=255; i++){
   S-Box[i] = i
```

```
    T[i] = kunci[ i mod panjang_kunci]
}
```

Pseudocode untuk melakukan permutasi *array* S adalah :

```
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
}
```

2.3.2 Pseudo Random generation Algorithm (PRGA)

Proses *pseudo random generation* merupakan proses yang dilakukan untuk membangkitkan kunci sebanyak elemen *plaintext* yang akan dienkripsi. Proses ini melibatkan nilai-nilai pada tabel *array* S yang telah dipermutasi (diacak). Kunci-kunci inilah nantinya yang akan di XOR-kan dengan *plaintext*. Adapun *pseudo code* untuk melakukan PRGA 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]
}
```

2.3.3 Enkripsi dan Dekripsi

Proses enkripsi dan dekripsi dilakukan dengan cara yang sederhana yaitu melakukan proses XOR antara biner *plaintext* dengan biner kunci yang telah dibangkitkan dari proses PRGA.

Formula enkripsi adalah :

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

Formula dekripsi adalah :

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

2.4 Beaufort Cipher

Beaufort cipher merupakan salah satu algoritma dalam teknik keamanan kriptografi klasik. Kunci (K) pada *beaufort cipher* adalah urutan karakter-karakter $K = k_1 \ldots k_d$ dimana k_1 didapat dari banyaknya pergeseran dari alfabet ke-*i* sama seperti *viginere cipher*[1]. Artinya bahwa jumlah kunci yang dibangkitkan harus sama dengan jumlah karakter *plaintext* yang diamankan. Algoritma ini melakukan proses enkripsi dan dekripsi secara *stream* (masing-masing karakter *plaintext* harus memiliki pasangan kunci). Hal ini yang menyebabkan algoritma ini sama hampir sama dengan algoritma *vigeneere cipher*. Adapun formulasi yang digunakan dalam proses enkripsi dan dekripsi [2][11] adalah :

Formula proses enkripsi :

$C_i = E_k(M_i) = (K_i - M_i)$ Mod 26 .....(3)

Formula proses dekripsi :
$M_i = D_k(C_i) = (K_i - C_i)$ Mod 26 .....(4)
Keterangan :
Mi = Pesan yang akan dienkripsi (*plain*)
Ci = Sandi (*cipher*) Ki = Kunci
Ek = Fungsi Enkripsi Dk = Fungsi Dekripsi

Nilai mod 26 di atas tergantung dari jumlah kebutuhan karakter yang digunakan, pada awalnya *beaufort cipher* hanya menggunakan 26 karakter, namun seiring dengan perkembangan teknologi komputer saat ini, maka dapat menggunakan mod 256 (mengguakan seluruh tabel ASCII).

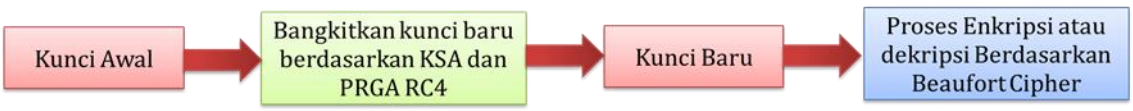
3. HASIL DAN PEMBAHASAN

3.1 Analisa Masalah

Proses penentuan kunci dalam enkripsi dan dekripsi data berdasarkan *beaufort cipher* sangat sederhana, namun jumlah kunci yang digunakan menjadi salah satu hal yang menyita waktu banyak bagi pengguna. Berdasarkan teoritis bahwa jumlah kunci yang digunakan pada *beaufort cipher* berbanding lurus (sama dengan) jumlah karakter *plaintext*, menyebabkan pengguna kesulitan mengingat kata kunci yang digunakan pada proses enkripsi maupun dekripsi. Umumnya hal ini dapat diatasi dengan melakukan pengulangan kunci yang jumlahnya lebih sedikit hingga jumlanya sama dengan banyaknya *plaintext*. Namun hal ini tidaklah memenuhi kaidah kunci kriptografi yang baik.

Proses pembangkitan kunci berdasarkan algoritma RC4 mampu menghasilkan bilangan-bilangan kunci yang acak sejumlah *plaintext*. Artinya proses *key schedulling* dan *pseudo random generation* akan membangkitkan kunci yang baru dengan jumlah yang sama seperti jumlah *plaintext*. Pemanfaatan kunci yang dibangkitkan berdasarkan algoritma RC4 sebagai kunci dalam proses enkripsi maupun dekripsi berdasarkan *beaufort cipher* tentu akan mengoptimalkan ketahanan dan memudahkan proses pembangkitan kunci pada *cipher* ini khususnya untuk menyandikan SMS.

Adapun skema optimasi yang dilakukan pada *beaufort cipher* dalam menyandikan SMS ditunjukkan pada gambar 1.

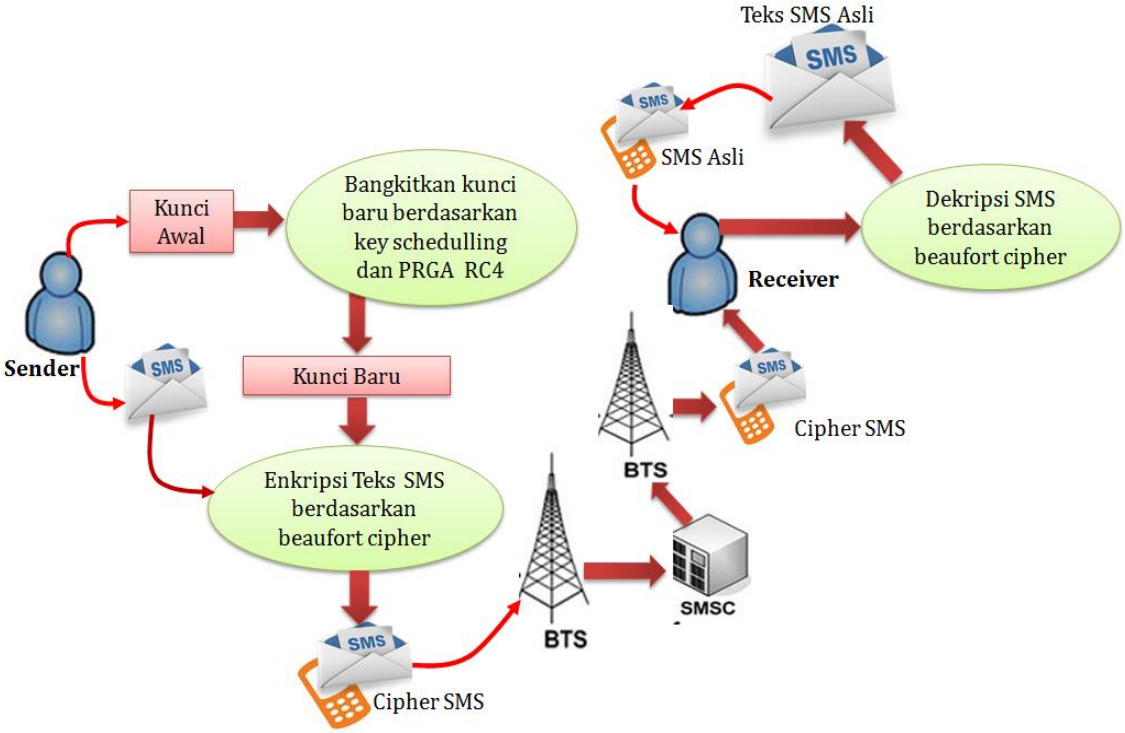


Gambar 1. Skema Otimalisasi Pembangkitan Kunci pada Beaufort Cipher

Berdasarkan gambar 1 di atas, terlihat bahwa kunci yang digunakan pada proses enkripsi maupun dekripsi bukan lagi kunci yang diberikan pada awal, melainkan akan dibangkitkan kunci-kunci yang baru yang lebih acak dan berbeda dengan kunci awal. Kelebihan teknik ini adalah pengguna tidak lagi harus menginputkan kunci sejumlah *plaintext*. pengguna dapat saja menginpukan jumlah kunci yang jauh lebih kecil dari jumlah *plaintext* karena secara otomatis akan dibangkitkan kunci-kunci baru sejumlah *plaintext* berdasarkan proses *key schedulling* dan *pseudo random generation* berdasarkan algoritma RC4. Kunci baru

yang dihasilkan pada proses inilah yang digunakan untuk melakukan enkripsi dan dekripsi SMS berdasarkan algoritma *beaufort cipher*.

Skema penerapan algoritma *beaufort cipher* yang telah teroptimalisasi dalam penyandian SMS ditunjukkan pada gambar di bawah ini :



Gambar 2. Skema Implementasi Penyandian SMS Berdasarkan Beaufort Cipher yang telah dioptimalisasikan

Berdasarkan gambar 2 di atas, terlihat bahwa SMS yang dikirim melalui handphone pengirim kepada penerima adalah SMS yang telah disandikan terlebih dahulu berdasarkan *beaufort cipher* yang telah dioptimalisasikan proses pembentukan kunci yang digunakan. Walaupun proses pengiriman SMS belum bersifat *point-to-point*, maka SMS tersebut tetap tidak dapat dipahami oleh pihak lain yang berhasil menyadapnya.

Proses pengamanan ini dilakukan dengan membangun aplikasi pengiriman SMS pada handphon yang dilengkapi dengan fitur enkripsi dan dekripsi SMS. Proses penyandian SMS dilakukan oleh penerima berdasarkan algoritma yang sama dan kunci yang sama. Oleh karenanya sebelum kedua belah pihak melakukan distribusi SMS, maka kedua belah pihak harus telah menyepakati kunci yang digunakan.

3.2 Implementasi

Contoh kasus berikut ini akan mengimplementasikan pembangkitan kunci *beaufort cipher* berdasarkan proses key *schedulling algorithm* dan *pseudo random generation algorithm* pada algoritma RC4. Asumsikan teks SMS yang dijadikan sebagai *plaintext* adalah **Hai Mia...** Kemudian ditetapkan kunci awal adalah **Diana**.

Berdasarkan skema optimalisasi pembentukan kunci (gambar 1) terhadap *beaufort cipher*, maka sebelum proses enkripsi dan dekripsi dilakukan terlebih dahulu dilakukan proses KSA dan PRGA berdasarkan algoritma RC4.

1. Proses *Key Schedulling Algorithm* (KSA)
   Teks SMS = Hai Mia...
   Kunci Awal = Diana
   Sebelum dilanjutkan pada proses selanjutnya, maka terlebih dibuat tabel array untuk kunci awal.

Tabel 1. Tabel Array Kunci Awal

Index	Char Kunci	Nilai ASCII
0	D	68
1	i	105
2	a	97
3	n	110
4	a	97

Proses pembentukan *array* S, yaitu tabel berisi *array* dengan jumlah 255 dan isi *array* dimulai dari 0-255. Proses ini dilakukan berdasarkan *pseudocode* KSA, sehingga dihasilkan *array* S sebagai berikut :

Tabel 2. Nilai Array S

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
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
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

Setiap kolom diberi *index* dari kiri ke kanan dengan *index* yang dimulai dari 0 hingga 255. Kolom pertama baris pertama berarti *index* 0 nilainya 0, kolom kedua baris pertama adalah *index* 1 dengan nilai 1, demikian seterusnya.

Langkah selanjutnya adalah membentuk tabel *array* T. *Array* T terdiri dari 256 *array* yang terdiri dari *index* 0 hingga 255. Nilai *array* diambil dari nilai-nilai desimal kunci awal berdasarkan *pseudocode* pembangkitan *array* T. Proses ini melibatkan nilai-nilai pada tabel 1.

Iterasi i = 0, maka :

T[0] = Kunci[0 mod 5]

Artinya bahwa isi *array* T pada *index* 0 adalah nilai *array* kunci pada *index* 0 mod 5. Nilai 5 adalah jumlah karakter (panjang) kunci awal.

Sehingga didapatkan :

T[0] = Kunci[0] = 68, artinya ambil nilai *array* T pada *index* ke-0 dan jadikan menjadi isi *array* T pada index ke-0.

Untuk iterasi i = 1, maka :

T[1] = Kunci[1 mod 5]

T[1] = Kunci[1] = 105

artinya ambil nilai *array* kunci awal pada *index* ke-1 dan jadikan menjadi isi *array* T pada index ke-1.

Proses ini dilakukan hingga iterasi ke 255, sehingga dihasilkan susunan *array* T sebagai berikut :

Tabel 3. Nilai Array T

68	105	97	110	97	68	105	97	110	97	68	105	97	110	97	68
105	97	110	97	68	105	97	110	97	68	105	97	110	97	68	105
97	110	97	68	105	97	110	97	68	105	97	110	97	68	105	97
110	97	68	105	97	110	97	68	105	97	110	97	68	105	97	110
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
110	97	68	105	97	110	97	68	105	97	110	97	68	105	97	110
97	68	105	97	110	97	68	105	97	110	97	68	105	97	110	97

Langkah berikutnya adalah proses permutasi (pertukaran) nilai-nilai *array* S serta pembentukan tabel *array* T sebagai acuan untuk membangkitkan kunci yang baru.

Iterasi i = 0; j = 0

j = (0 + S[0] + T[0]) mod 256

nilai *array* S[0] = 0 dan T[0] = 68, sehingga :

= (0 + 0 + 68) mod 256

j = 68

swap (S[0], S[68]); i = 0 dan j = 68

artinya tukarkan isi *array* index ke-0 dengan isi *array* ke-68 dan begitu sebaliknya.

Iteasi i = 1; j = 68

j = (68 + S[1] + T[1]) mod 256

nilai j sebelumnya adalah 68, *array* S[1] = 1 dan T[1] = 105, sehingga :

= (68 + 1 + 105) mod 256

j = 174

swap (S[1], S[174]); i = 1 dan j = 175

artinya tukarkan isi *array* S index ke-1 dengan isi *array* S ke-174 dan begitu sebaliknya. Proses ini dilakukan hingga iterasi ke 255 (seluruh nilai *array* S akan teracak). Proses ini memungkinkan nilai *array* tertukar lebih dari sekali. Hasil permutasi *array* S, ditunjukkan pada tabel 4 di bawah ini.

Tabel 4. Nilai Permutasi Array S

35	175	176	195	16	171	192	23	29	28	205	169	89	70	12	83
204	47	255	19	123	85	101	206	165	162	36	184	74	201	42	127
51	96	69	234	41	18	91	166	231	159	198	92	5	218	113	225
7	136	79	216	34	191	124	132	137	156	39	187	9	141	105	24
135	130	31	146	177	168	238	13	160	55	82	251	209	250	182	90
221	151	88	199	40	131	147	21	80	22	66	110	220	246	58	121
194	227	44	219	6	188	68	183	32	108	33	143	115	52	157	63
172	11	129	240	50	134	253	104	8	4	14	247	232	243	45	186

144	94	99	189	155	142	193	122	30	150	98	28	116	133	72	139
25	228	106	59	185	37	20	126	61	93	86	148	236	154	118	43
213	102	111	38	212	100	203	164	109	26	140	233	190	1	174	71
87	27	170	48	158	226	10	2	120	84	178	103	161	78	145	17
114	202	254	81	214	153	46	241	173	200	249	60	207	224	235	217
245	77	75	97	15	239	208	223	180	125	244	149	215	95	196	211
53	248	163	49	28	252	67	76	54	119	237	62	229	197	112	242
210	230	65	117	222	0	57	3	167	107	181	152	56	128	138	179

2. Proses *Pseudo Random Generation Algorithm* (PRGA)
   Proses ini merupakan proses yang dilakukan untuk membangkitkan *key stream* (aliran kunci) yang digunakan pada proses enkripsi dan dekripsi SMS berdasarkan *beaufort cipher*. Jumlah kunci yang dibangkitkan sesuai dengan jumlah karakter teks SMS yang disandikan. Contoh kasus pada penelitian ini memiliki jumlah karakter SMS adalah 10, sehingga akan dibangkitkan 10 kunci yang baru. Pembangkitan kunci yang acak menggunakan nilai-nilai array S hasil permutasi.

   Bila iterasi i = 0; j = 0, maka :
   i = (0 + 1) mod 256
   i = 1
   j = (0 + S[1]) mod 256
   = (0 + 175) mod 256
   j = 175
   swap (S[1], S[175])
   t = (S[1] + S[175]) mod 256
   = (175 + 71) mod 256, nilai array S pada index 175 adalah 71
   = 246 mod 256
   t = 246, sehingga didapatkan :
   Kunci [1] = S[246]
   Kunci [1] = 57, nilai array S pada index 246 adalah 57
   Artinya kunci yang digunakan untuk mengenkripsi karakter pertama adalah nilai desimal 57.
   Bila iterasi i = 1; j = 175 (nilai j iterasi sebelumnya), maka :
   i = (1 + 1) mod 256
   i = 2
   j = (175 + S[2]) mod 256
   = (175 + 176) mod 256
   j = 95
   swap (S[2], S[95])
   t = (S[2] + S[95]) mod 256
   = (176 + 121) mod 256, nilai *array* S pada *index* 95 adalah 121
   = 41 mod 256
   t = 41, sehingga didapatkan :
   Kunci [2] = S[41]
   Kunci [2] = 159, nilai *array* S pada *index* 41 adalah 159

Artinya kunci yang digunakan untuk menyandikan karakter SMS (huruf ke dua) adalah nilai desimal 159.
Proses ini dilakukan hingga iterasi ke 10 (jumlah karakter SMS), sehingga nilai bilangan kunci yang dibangkitkan secara acak adalah 57, 159, 41, 94, 42, 189, 78, 163, 200, 201 (terlihat bahwa bukan lagi nilai-nilai kunci awal).

3. Proses Enkripsi dan Dekripsi
Proses enkripsi dilakukan berdasarkan formula enkripsi *beuafort cipher* (persamaan 3).
C_1 = (K_1 – M_1) mod 256
= (57 – H) mod 256
= (57 – 72) mod 256
= 241 (char ñ)
C_2 = (K_2 – M_2) mod 256
= (159 – a) mod 256
= (159 – 97) mod 256
= 62 (char >)
Cipher lainnya dicari dengan cara yang sama, sehingga sandi SMS **Hai Mia...** adalah **ñ>À>Ý—ŸÒÒÒ**

Proses dekripsi oleh penerima SMS yang diawali oleh proses pembangkitan kunci berdasarkan algoritma RC4 (proses KSA dan PRGA), kemudian melakukan proses dekripsi berdasarkan persamaan 4.
P_1 = (K_1 – C_1) mod 256
= (57 – ñ) mod 256
= (57 – 241) mod 256
= 72 (char H)
P_2 = (K_2 – C_2) mod 256
= (159 – >) mod 256
= (57 – 62) mod 256
= 97 (char a)
demikian seterusnya, hingga dihasilkan *plaintext* SMS adalah **Hai Mia...**

4. KESIMPULAN

Berdasarkan uraian analisa dan pembahasan, maka disimpulkan beberapa hal sebagai berikut :

a. Optimaliasi pembangkitan kunci sejumlah *plaintext* berdasarkan proses KSA dan PRGA dari algoritma RC4 terhadap *beaufort cipher* sangat efektif, karena pengguna dapat menggunakan jumlah karakter kunci yang jauh lebih kecil dibandingkan jumlah *plaintext* sehingga mudah mengingatnya.
b. Kunci yang dihasilkan cukup acak dan bahkan menghasilkan karakter-karakter kunci yang baru sehingga prinsip *diffusion* pada kunci terwujudkan.
c. Pengamanan SMS berdasarkan *beaufort cipher* yang telah dioptimalisasikan cukup baik yang dapat dibuktikan dengan tidak terlihatnya karakteristik dan pola *plain* SMS dengan *cipher* SMS yang dihasilkan sehingga hal ini dapat mengelabui dan mempersulit para penyerang.

DAFTAR PUSTAKA

[1] N. Widyastuti, "Pengembangan Metode Beaufort Cipher Menggunakan Pembangkit Kunci Chaos," *J. Teknol.*, vol. 7, no. 1, pp. 73–82, 2014.
[2] E. Setyaningsih, "Penyandian Citra Menggunakan Metode Playfair Cipher", J. Teknol., vol. 2, no. 2, pp. 213–219, 2009.
[3] T. Zebua and E. Ndruru, "PENGAMANAN CITRA DIGITAL BERDASARKAN MODIFIKASI ALGORITMA RC4," *J. Teknol. Infomasi dan Ilmu Komput.*, vol. 4, no. 4, pp. 275–282, 2017.
[4] H. Agung and Budiman. "Implementasi Affine Chiper dan RC4 Pada Enkripsi File Tunggal," Prosiding SNATIF, pp. 243–250, 2015
[5] S. Subhan, S. Amini, and P. F. Ariyani, "Implementasi Pengamanan Data Enkripsi Sms Dengan Algoritma RC4 Berbasis Android," in *Seminar Nasional Inovasi Dan Aplikasi Teknologi Di Industri 2017 ITN Malang*, 2017, pp. 1–6.
[6] F. Purwaningsih and M. Badrul, "Penerapan algoritma huffman untuk aplikasi pengamanan sms berbasis android," *J. PROSISKO*, vol. 4, no. 2, pp. 60–66, 2017.
[7] W. P. Atmojo, R. R. Isnanto, and R. Kridalukmana, "Implementasi Aplikasi Kriptografi Pada Layanan Pesan Singkat (SMS) Menggunakan Algoritma RC6 Berbasis Android," *J. Teknol. dan Sist. Komput.*, vol. 4, no. 3, p. 450, 2016.
[8] E. Setyaningsih, *Kriptografi & Implementasinya Menggunakan Matlab*, Yogyakarta:Andi, 2015.
[9] T. Zebua, "ANALISA DAN IMPLEMENTASI ALGORITMA TRIANGLE CHAIN PADA PENYANDIAN RECORD DATABASE," *Pelita Inform. Budi Darma*, vol. 3, no. 2, pp. 37–49, 2013.
[10] R. Sadikin, *Kriptografi Untuk Keamanan Jaringan dan Implementasinya dalam Bahasa Java*, Yogyakarta: Andi, 2012.
[11] Wikipedia, "Beaufort Cipher," [online]. Available: *https://en.wikipedia.org/wiki/Beaufort_cipher*, 2017 [Accessed 30 Januari 2018].