Jurnal Sains Komputer & Informatika (J-SAKTI)
Volume (2) No.1 Maret 2018, pp. 33-46
ISSN:2548-9771/EISSN:2549-7200
http://tunasbangsa.ac.id/ejurnal/index.php/jsakti

# Penerapan Algoritma *Skipjack* Untuk Menyandikan *Short Message Service*

**Nurainun Sinaga[1], Syarifah Aini[2], Bezatulo Gulo[3]**
[1,2,3]Mahasiswa Prorgram Studi Teknik Informatika STMIK Budidarma Medan
[1,2,3]Jln. Sisingamangaraja No. 338 Sp. Limun Medan
[1] nurainun267@gmail.com, [2]syarifahaini1995@gmai.com, [3]bezatulog@gmail.com

***Abstract***

*The security and confidentiality of Short Message Service (SMS) is very important for senders and recipients of messages. Generally, SMS is displayed in plain text form which can make it easy for the cryptanalyst to skipjack the message to be distributed. SMS security can be done by encrypting it based on one of the cryptographic algorithms in order to maintain the authenticity and security of messages and can make it difficult for others to know the original meaning of the SMS. Skipjack algorithm is one of cryptography algorithm that can be used for SMS encoding. The Skipjack algorithm is an electronic 64-bit codebook algorithm that converts 64-bit blocks into 64-bit blocks of output. The parameter used for encryption is 80-bit key and has 32 rounds of the encryption process and description. This study describes how SMS text encoding based on skipjack algorithm with the purpose of distributed SMS can be guaranteed to be safe and avoid misuse.*

***Keywords:*** *Cryptography, Algorithm, Skipjack, SMS.*

***Abstrak***

*Keamanan dan kerahasian Short Message Service (SMS) sangatlah penting bagi pengirim dan penerima pesan. Umumnya SMS ditampilkan dalam bentuk teks yang biasa yang dapat mempermudah kriptanalis untuk membajak pesan yang akan didistribusikan. Pengamanan SMS dapat dilakukan dengan menyandikannya berdasarkan salah satu dari algoritma kriptografi dengan tujuan menjaga keaslian dan keamanan pesan serta dapat mempersulit pihak lain untuk mengetahui makna asli dari SMS. Algoritma skipjack merupakan salah satu algoritma kriptografi yang dapat digunakan untuk penyandian SMS. Algoritma Skipjack merupakan algoritma elektronik codebook 64-bit yang merubah 64-bit blok masukkan menjadi 64-bit blok keluaran. Parameter yang digunakan untuk enkripsi adalah 80-bit kunci, dan mempunyai 32 putaran untuk proses enkripsi dan deskripsi. Penelitian ini menguraikan bagaimana penyandian teks SMS berdasarkan algoritma skipjack dengan tujuan SMS yang didistribusikan dapat terjamin keamanannya dan terhindar dari penyalahgunaan.*

***Kata Kunci****: Kriptografi, Algoritma, Skipjack,* SMS*.*

## 1. PENDAHULUAN

Pengiriman suatu data dan penyimpanan data menggunakan media elektronik memerlukan suatu proses yang menjamin keamanan dan keutuhan dari sebuah data misalnya saja *Short Message Service* (SMS). Data tersebut harus tetap bersifat rahasia selama proses pengirimannya hingga samapai ke tempat tujuan. Subhan dalam penelitiannya, mengatakan bahwa SMS sebagai layanan komunikasi pada saat didistribusikan masih harus melewati pusat penyedia layanan, artinya

belum bersifat *point-to-point*, sehingga dapat memungkinkan terjadinya penyadapan SMS[1]. Berdasarkan penelitian lainnya yang dilakukan oleh Purwaningsih bahwa komunikasi via SMS memiliki celah keamanan terbesar dimana pesan yang dikirimkan akan disimpan di SMSC (*Short Message Service Center*) sebelum dikirim ke tujuan[2].

Salah satu solusi untuk menyelesaikan masalah di atas adalah melakukan proses penyandiaan (enkripsi dan dekripsi) terhadap SMS yang akan dikirimkan. Enkripsi dilakukan untuk mengubah data asli menjadi data rahasia, sedangkan dekripsi dilakukan oleh penerima untuk mengetaui isi data asli dengan cara mengubah data rahasia menjadi data asli[3][4]. Proses ini disebut dengan teknik kriptografi dimana selama proses pengiriman data, data yang dikirim harus bersifat rahasia dan data asli hanya diketahui oleh pengirim dan penerima dengan menggunakan kunci rahasia.

Algoritma *Skipjack* merupakan salah satu dari algoritma kriptografi yang dapat digunakan dalam mengamanan data rahasia yang dikembangkan pada tahun 1987. *Skipjack* merupakan represantasi dari *family of encryption algorithms* sebagai bagian dari algoritma Tipe I yang dikembangkan Badan Kemanan Nasional Amerika Serikat. Berdasarkan penelitian terdahulu yang dilakukan oleh Suprianto mengatakan spesifikasi dari sebuah algoritma yang baik dan tahan terhadap setiap jenis serangan adalah mempunyai struktur yang sederhana serta tidak rumit dan algoritma *skipjack* termasuk dalam jenis ini[5].

Penelitian ini menguraikan proses yang dilakukan untuk meng-implementasikan algoritma *skipjack* dalam sebuah aplikasi SMS. Hal yang dilakukan adalah sebelum SMS dikirimkan oleh pengirim, maka terlebih dahulu dilakukan proses penyandian (enkripsi) SMS tersebut, sehingga bila SMS tersebut berhasil disadap oleh orang lain, maka yang didapatkan adalah *cipher* SMS dalam bentuk sandi-sandi. Upaya ini dapat meminimalkan tindakan-tindakan pihak lain selain penerima SMS yang sah untuk melakukan manipulasi ataupun penyalahgunaan SMS.

## 2. METODOLOGI PENELITIAN

### 2.1 Short Message Service

Salah satu fasilitas telepon selular dalam proses pengiriman atau penerimaan pesan singkat baik yang bersifat rahasia maupun tidak rahasia adalah *Short Message Service* (SMS). Salah satu kelebihan yang dimiliki SMS adalah biaya yang relatif cukup murah[3]. Namun hingga saat ini layanan SMS masih belum bersifat *point-to-point*, melainkan harus singgah terlebih dahulu di SMS Center (SMSC) untuk diteruskan kepada tujuan (penerima). SMS dari pengguna ponsel dikirim melalui *wireless* dari menara seluler ke SMSC. Protocol akses menggunakan interaksi antara dua SMSC atau interaksi internal antara eksternal *Short Messege Entities* (SMEs) dan SMS Center[1]. Layanan SMS yang tidak bersifat *point-to-point* tersebut sering dimanfaatkan sebagai celah bagi para penyadap untuk mencuri dan memanfaatkan SMS yang sedang dikelolah di SMSC[2]. Bila hal ini terjadi, maka tentu akan merugikan pihak pengirim atau pemilik SMS.

## 2.2 Kriptografi

Saat ini, teknik kriptografi menjadi salah satu teknik yang umum digunakan dalam mengamankan data yang sifatnya pribadi atau rahasia. Pengamanna data berdasarkan 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[6][7]. 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. Tujuan yang harus dicapai bila teknik kriptografi diimplementasikan adalah kerahasiaan, integritas, otentikasi dan ketiadaan penyangkalan[4][8].

## 2.3 Algoritma Skipjack

Algoritma *Skipjack* merupakan salah satu dari algoritma kriptografi yang dapat digunakan dalam mengamanan data rahasia yang dikembangkan pada tahun 1987. *Skipjack* merupakan represantasi dari *family of encryption algorithms* sebagai bagian dari algoritma Tipe I yang dikembangkan Badan Kemanan Nasional Amerika Serikat. Berdasarkan penelitian terdahulu yang dilakukan oleh Suprianto mengatakan spesifikasi dari sebuah algoritma yang baik dan tahan terhadap setiap jenis serangan adalah mempunyai struktur yang sederhana serta tidak rumit dan algoritma *skipjack* termasuk dalam jenis ini[5].

Terdapat dua tipe putaran dalam *skipjack cipher* yang disebut dengan *stepping rule* [9]. Kedua tipe tersebut adalah :.

Tipe A :

1. Upablok W1 dienkripsi dengan fungsi permutasi G yang adalah empat putaran *feistel cipher* biasa.
2. Hasil enkripsinya dan nomor putaran yang bertambah dari satu sampai dengan 32, di xor dengan upablok W4.
3. Setiap upablok dirotasi W1 ke W2, W2 ke W3, W3 ke W4, dan W4 ke W1.

Tipe B :

1. Upablok W2 di-xor dengan W1 dan nomor putaran.
2. W1 dienkripsi dengan fungsi permutasi G.
3. Setiap upablok dirotasi W1 ke W2, W2 ke W3, W3 ke W4, dan W4 ke W1.

Ke-32 putaran *feistel* tak seimbang pada algoritma *skipjack* terdiri dari delapan putaran tipe A, delapan putaran tipe B, delapan putaran tipe A, dan delapan putaran tipe B. Upa-kunci pada setiap putaran digunakan di dalam fungsi *feistel* pada fungsi permutasi G. Fungsi *feistel* dalam fungsi permutasi G menerima *input* delapan *bit*. *Input* tersebut kemudian di-xor dengan upa-kunci. Kemudian hasilnya disubstitusi berdasarkan tabel *lookup* yang disebut tabel F.

**Tabel 1. Tabel Feistel Skipjack**

| | x0 | x1 | x2 | x3 | x4 | x5 | x6 | x7 | x8 | x9 | xA | xB | xC | xD | xE | xF |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x | a3 | d7 | 09 | 83 | f8 | 48 | f6 | f4 | b3 | 21 | 15 | 78 | 99 | b1 | af | f9 |
| 1x | e7 | 2d | 4d | 8a | ce | 4c | ca | 2e | 52 | 95 | d9 | 1e | 4e | 38 | 44 | 28 |

| | x0 | x1 | x2 | x3 | x4 | x5 | x6 | x7 | x8 | x9 | xA | xB | xC | xD | xE | xF |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2x | 0a | df | 02 | a0 | 17 | f1 | 60 | 68 | 12 | b7 | 7a | c3 | e9 | fa | 3d | 53 |
| 3x | 96 | 84 | 6b | ba | f2 | 63 | 9a | 19 | 7c | ae | e5 | f5 | f7 | 16 | 6a | a2 |
| 4x | 39 | b6 | 7b | 0f | c1 | 93 | 81 | 1b | ee | b4 | 1a | ea | d0 | 91 | 2f | b8 |
| 5x | 55 | b9 | da | 85 | 3f | 41 | bf | e0 | 5a | 58 | 80 | 5f | 66 | 0b | d8 | 90 |
| 6x | 35 | d5 | c0 | a7 | 33 | 06 | 65 | 69 | 45 | 00 | 94 | 56 | 6d | 98 | 9b | 76 |
| 7x | 97 | fc | b2 | c2 | b0 | Fe | db | 20 | e1 | eb | d6 | e4 | dd | 47 | 4a | 1d |
| 8x | 42 | ed | 9e | 6e | 49 | 3c | cd | 43 | 27 | d2 | 07 | d4 | de | c7 | 67 | 18 |
| 9x | 89 | cb | 30 | 1f | 8d | c6 | 8f | aa | c8 | 74 | dc | c9 | 5d | 5C | 31 | a4 |
| Ax | 70 | 88 | 61 | 2c | 9f | 0d | 2b | 87 | 50 | 82 | 54 | 64 | 26 | 7d | 03 | 40 |
| Bx | 34 | 4b | 1c | 73 | d1 | c4 | fd | 3b | cc | Fb | 7f | ab | e6 | 3e | 5b | a5 |
| Cx | ad | 04 | 23 | 9c | 14 | 51 | 22 | f0 | 29 | 79 | 71 | 7e | ff | 8c | 0e | e2 |
| Dx | 0c | ef | bc | 72 | 75 | 6f | 37 | a1 | ec | d3 | 8e | 62 | 8b | 86 | 10 | e8 |
| Ex | 08 | 77 | 11 | be | 92 | 4f | 24 | c5 | 32 | 36 | 9d | cf | f3 | a6 | bb | Ac |
| Fx | 5e | 6c | a9 | 13 | 57 | 25 | b5 | e3 | bd | a8 | 3a | 01 | 05 | 59 | 2a | 46 |

Kunci rahasia pada algoritma skipjack memiliki panjang 80 bit, kemudian dikelola secara sederhana yyaitu mengelompokkan biner kunci menjadi 10 kelompok (CV0, CV1, CV2,....,CV9), artinya bahwa masing-masing kelompok terdiri dari 8 bit. Kelompok-kelompok bit kunci inilah yang digunakan dalam proses enkripsi maupun dekripsi.

Permutasi G yang merupakan 4 *round* dari struktur *Feistel* adalah fungsi permutasi pada algoritma *skipjack*. Tabel substitusi *byte* yang *fixed* merupakan fungsi *round*, yang dinamakan *F-Table*. Masing-masing *round* dari permutasi G juga memasukkan sebuah *cryptovariable* (CV). Permutasi G dilakukan pada proses enkripsi di awal setiap *rule* yakni *rule A dan rule B*. Proses pemutasi G pada dekripsi merupakan kebalikan (*inverse*) dari permutasi G yang disebut dengan permutasi G-1 yang dilakukan di awal setiap *rule A-1* dan *rule B-1*. Sebagai masukan (*input*) untuk melakukan proses permutasi adalah seperempat bagian dari blok *plaintext* ataupun blok *ciphertext* dalam bentuk heksadesimal yang berukuran 16 bit. Berikut ini adalah langkah-langkah dari permutasi G dan permutasi G-1 :

1. Permutasi G, G(Word = g1 || g2) = g5 || g6 di mana g1 adalah byte pertama dari *Word (high byte)* dan g2 adalah *byte* kedua dari *Word (low byte)* dan sebagai hasilnya (*output*) adalah gabungan antara g5 dengan g6. Untuk g3, g4, g5, dan g6, rumus yang berlaku adalah formula gi = F(gi-1⊕cv4k+i-3) ⊕gi-2. Dimana 3 ⊕ i ⊕ 6 (i awal = 3), k pada proses enkripsi putaran pertama adalah 0, F merupakan tabel substitusi atau *F-Table*, dan cv4k+i-3 adalah *cryptovariable* dengan indeks (4k+i-3) dalam cryptovariable schedule. Sesuai dengan rumus gi = F(gi-1⊕cv4k+i-3) ⊕gi-2 maka :
   g3 = F(g2⊕cv4k) ⊕g1
   g4 = F(g3⊕cv4k+1) ⊕g2
   g5 = F(g4⊕cv4k+2) ⊕g3
   g6 = F(g5⊕cv4k+3) ⊕g4
2. Permutasi G-1, G-1(Word = g5 || g6) = g1 || g2di mana g5 adalah *byte* pertama dari *word (high byte)* dan g6 adalah *byte* kedua dari *Word (low byte)* dan

sebagai hasilnya (*output*) adalah gabungan antara g1 dengan g2. Untuk g4, g3, g2, g1 , rumus yang berlaku adalah sebagai berikut :

gi-2 = F(gi-1⊕cv4(k - 1) + i - 3) ⊕gi

di mana 3 ⊕ i ⊕ 6 , (i awal = 6), k pada proses dekripsi putaran pertama adalah 32, F merupakan tabel substitusi atau *F-Table*, dan cv4(k-1)+i-3 adalah *cryptovariable* dengan indeks (4(k-1)+i-3) dalam *cryptovariable schedule*. Sesuai dengan rumus gi-2 = F(gi-1⊕cv4(k - 1) + i - 3) ⊕ gi, maka :

g4 = F(g5⊕cv4(k – 1) + 3) ⊕ g6

g3 = F(g4⊕cv4(k – 1) + 2) ⊕ g5

g2 = F(g3⊕cv4(k – 1) + 1) ⊕ g4

g1 = F(g2⊕cv4(k – 1)) ⊕g3

Panjang *cryptovariable* adalah 10 *bytes* dengan label cv0 … cv10, maka *schedule* yang diberikan pada defenisi permutasi G dan permutasi $G^{-1}$ selalu dibuat modulo 10.

### 2.3.1 Proses Enkripsi

Langkah – langkah dari *ruleA* adalah sebagai berikut :

1. Lakukan permutasi G dengan *inputW1k*.
2. *W1k+1* merupakan hasil dari operasi XOR antara *output* permutasi G, *W4k*, dan *counter*.
3. *W2k+1* merupakan *output* dari permutasi G.
4. *W3k+1 = W2k*.
5. *W4k+1 = W3k*.
6. *Counter* dan *k* ditambah satu.

Langkah – langkah dari *rule B* adalah sebagai berikut :

1. Lakukan permutasi G dengan *inputW1k*.
2. *W1k+1 = W4k*.
3. *W2k+1* merupakan *output* dari permutasi G.
4. *W3k+1* merupakan hasil dari operasi XORantara *W1k*, *W2k*, dan *counter*.
5. *W4k+1 = W3k*.
6. *Counter* dan *k* ditambah satu.

### 2.3.2 Proses Dekripsi Skipjack

Langkah-langkah dari *ruleA-1* adalah sebagai berikut :

1. Lakukan permutasi G-1 dengan *inputW2k*.
2. *W1k-1* merupakan *output* dari permutasi G-1.
3. *W2k-1 = W3k*.
4. *W3k-1 = W4k*.
5. *W4k-1*= merupakan hasil dari operasi XOR antara *W1k*, *W2k* dan *counter*.
6. *Counter* dan *k* dikurangi satu.

Langkah – langkah dari *rule B-1* adalah sebagai berikut :

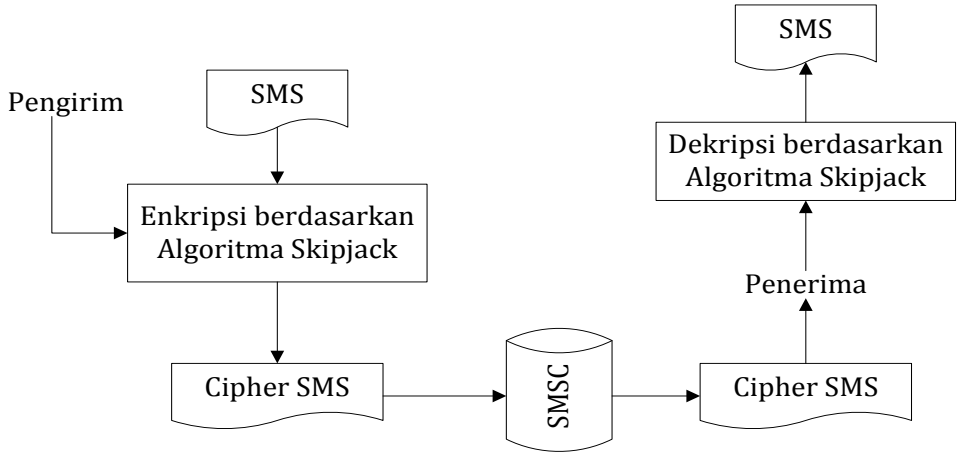
1. Lakukan permutasi G-1 dengan *inputW2k*.
2. *W1k-1* merupakan output dari permutasi G-1.
3. *W2k-1* merupakan hasil dari operasi XOR antara *output* permutasi G-1, *W3k*, dan *counter*.
4. *W3k-1 = W4k*.

5. *W4k-1 = W1k*.
6. *Counter* dan *k* dikurangi satu.

## 3. HASIL DAN PEMBAHASAN

### 3.1 Analisa Masalah

Layanan *Short Message Service* (SMS) yang belum bersifat *point-to-point* (belum langsung dikirimkan ke tujuan), menjadi salah satu celah bagi para penyerang untuk mencuri atau membaca SMS yang dikirim oleh pengirim kepada penerima. Penyadapan SMS yang isinya merupakan informasi penting atau bersifat rahasia akan merugikan pemilik SMS bila aktivitas penyadapan ini terjadi. Implemenasi teknik penyandian (kriptografi) berdasarkan algoritma *skipjack* mampu meminimalisir kelemahan tersebut di atas, sehingga SMS yang dikirimkan benar-benar aman. Tindakan minimalisir yang dilakukan dalah menyandikan teks SMS yang akan dikirimkan berdasarkan algoritma *skipjack*, sehingga yang didistribusikan adalah *cipher* dari teks SMS. Secara umum skema proses penerapan algoritma skipjack dalam menyandikan SMS dapat diilustrasikan pada diagram berikut.




**Gambar 1. Skema Penerapan Algoritma Skipjack pada SMS**

Berdasarkan gambar 1 di atas, terlihat bahwa sms pengirim terlebih dahulu melakukan proses enkripsi SMS. *Cipher* SMS yang dihasilkan akan dikirim dan dikelolah oleh SMS Center (SMSC) untuk dilanjutkan pengirimannya kepada penerima. *Cipher* SMS (bukan teks SMS asli) inilah yang sampai kepada penerima. Oleh karena itu, maka penerima harus melakukan proses dekripsi sehingga dihasilkan teks asli dari SMS dari pengirim.

### 3.2 Implementasi

Berikut ini akan diimplementasikan penyandian SMS berdasarkan prosedur algoritma *skipjack*. Diasumsikan teks SMS yang dikirim adalah **COMPUTER** dengan kunci enkripsi dan dekripsi adalah *CRYPTOLOGI*.

a. Proses Pengolahan Kunci
   Kunci = “CRYPTOLOGY”
   Bentuk heksadesimal = 43525950544F4C4F4759

Bagi kunci menjadi 10 subkunci (*cryptovariable*) masing – masing 8 bit sebagai berikut :
cv[0] = 43; cv[1] = 52; cv[2] = 59; cv[3] = 50;
cv[4] = 54 cv[5] = 4F; cv[6] = 4C; cv[7] = 4F;
cv[8] = 47; cv[9] = 59

Selanjutnya dilakukan proses permutasian
Permutasian G :
Input : 434F ; K = 0
g1 = Mid(Input,1,2) = 43
g2 = Mid(Input,3,2) = 4F

g3= F(g2 ⊕ cv[(4*k)mod 10]) ⊕ g1
cv[(4*0)mod 10] = cv[0] = 43

g2 = 4F : 0100 1111
cv[0] = 43 : 0100 0011 ⊕
0000 1100

F(0000 1100) = F(0C) = 99

F(0C) = 99 : 1001 1001
g1 = 43 : 0100 0011 ⊕
g3 = 1101 1010 (**DA**)

g4= F(g3⊕ cv[((4*k)+1)mod 10]) ⊕g2
cv[((4*0)+1)mod 10] = cv[1] = 52

g3 = DA : 1101 1010
cv[1] = 52 : 0101 0010 ⊕
1000 1000

F(1000 1000) = F(88) = 27

F(88) = 27 : 0010 0111
g2 = 4F : 0100 1111 ⊕
g4 = 0110 1000 (**68**)

g5= F(g4⊕ cv[((4*k)+2)mod 10]) ⊕g3
cv[((4*0)+2)mod 10] = cv[2] = 59

g4 = 68 : 0110 1000
cv[2] = 59 : 0101 1001 ⊕
0011 0001

F(0011 0001) = F(31) = 84

F(31) = 84 : 1000 0100
g3 = DA : 1101 1010 ⊕
g5 = 0101 1110 (**5E**)

g6= F(g5⊕ cv[((4*k)+3)mod 10]) ⊕g4
cv[((4*0)+3)mod 10] = cv[3] = 50

g5 = 5E : 0101 1110
cv[3] = 50 : 0101 0000 ⊕
0000 1110

F(0000 1110) = F(0E) = AF

F(0E) = AF : 1010 1111
g4 = 68 : 0110 1000 ⊕
g6 = 1100 0111 (**C7**)

**G = g5 + g6 = 5EC7**

Permutasi G-1 merupakan kebalikan (inverse) dari permutasi G.
Input = 31EE ; k = 32
g5 = Mid(Input,1,2) = 31
g6 = Mid(Input,3,2) = EE
g6= F(g5⊕ cv[(4*(k-1)+3)mod 10]) ⊕g4
cv[(4*(32-1)+3)mod 10] = cv[7] = 4F

g5 = 31 : 0011 0001
cv[7] = 4F : 0100 1111 ⊕
0111 1110

F(0111 1110) = F(7E) = 4A

$$\begin{array}{l} F(7E) = 4A: 0100\ 1010 \\ g6 = EE: 1110\ 1110 \\ \hline g4 = 1010\ 0100\ (\mathbf{A4}) \end{array} \oplus$$

g3= F(g4⊕ cv[(4*(k-1)+2)mod 10]) ⊕g5
cv[(4*(32-1)+2)mod 10] = cv[6] = 4C

$$\begin{array}{l} g4 = A4: 1010\ 0100 \\ cv[6] = 4C: 0100\ 1100 \\ \hline 1110\ 1000 \end{array} \oplus$$

F(1110 1000) = F(E8) = 32

$$\begin{array}{l} F(E8) \\ = 32: 0011\ 0010 \\ g5 = 31: 0011\ 0001 \\ \hline g3 = 0000\ 0011\ (\mathbf{03}) \end{array} \oplus$$

g2= F(g3⊕ cv[(4*(k-1)+1)mod 10]) ⊕g4
cv[(4*(32-1)+1)mod 10] = cv[5] = 4F

$$\begin{array}{l} g3 = 03: 0000\ 0011 \\ cv[5] = 4F: 0100\ 1111 \\ \hline 0100\ 1100 \end{array} \oplus$$

F(0100 1100) = F(4C) = D0

$$\begin{array}{l} F(4C) = D0: 1101\ 0000 \\ g4 = A4: 1010\ 0100 \\ \hline g2 = 0111\ 0100\ (\mathbf{74}) \end{array} \oplus$$

g1= F(g2⊕ cv[(4*(k-1)mod 10]) ⊕g3
cv[(4*(32-1))mod 10] = cv[4] = 54

$$\begin{array}{l} g2 = 74: 0111\ 0100 \\ cv[4] = 54: 0101\ 0100 \\ \hline 0010\ 0000 \end{array} \oplus$$

F(0010 0000) = F(20) = 0A

$$\begin{array}{l} F(20) = 0A: 0000\ 1010 \\ g3 = 03: 0000\ 0011 \\ \hline g1 = 0000\ 1001\ (\mathbf{09}) \end{array} \oplus$$

**$G^{-1}$ = g1 + g2 = 0974**

b. Proses Enkripsi
Plaintext = NURAINUN
Kunci = CRYPTOLOGY
Pengolahan Kunci :
Ubah dalam : 43525950544F4C4F4759
Bagi kunci menjadi 10 bagian sebagai berikut:
cv(0) = 43; cv(1) = 52; cv(2) = 59; cv(3) = 50;

cv(4) = 54 cv(5) = 4F; cv(6) = 4C; cv(7) = 4F;
cv(8) = 47; cv(9) = 59

Proses Enkripsi :
Ubah plaintext dalam hexadecimal: 4E555241494E554E
Bagi plaintext menjadi 4 bagian (W1, W2, W3, W4) sebagai berikut :
W1(0) = 4E55; W2(0) = 5241; W3(0) = 494E;
W4(0) = 554E

**Putaran 1 (Rule A, K = 0, Counter = 1)**
G(W1(0)) = G(4E55) = g5 + g6 = 5EC7
g1 = Mid(W1(0),1,2) = 4E
g2 = Mid(W1(0),3,2) = 55
g3 = F(g2⊕ cv[(4*k) mod 10]) ⊕ g1
cv[(4*0)mod 10] = cv[0] = 43

g2 = 55 : 0101 0101
cv[0] = 43 : 0100 0011 ⊕
0000 0110

F(0000 0110) = F(06) =16

F(06) = 16 : 0001 0110
g1 = 43 : 0100 0011 ⊕
g3 = 0101 0110 (**56**)

g4= F(g3⊕ cv[((4*k)+1)mod 10]) ⊕g2
cv[((4*0)+1)mod 10] = cv[1] = 52

g3 = 56 : 0101 0110
cv[1] = 52 : 0101 0010 ⊕
0000 0100

F(0000 0100) = F(04) = 18

F(04) = 18 : 0001 1000
g2 = 55 : 0101 0101 ⊕
g4 = 0100 1101 (**4D**)

g5= F(g4⊕ cv[((4*k)+2)mod 10]) ⊕g3
cv[((4*0)+2)mod 10] = cv[2] = 59

g4 = 4D : 0100 1101
cv[2] = 59 : 0101 1001 ⊕
0001 0100

F(0001 0100) = F(14) = CE

F(14) = CE : 1100 1110
g3 = 56 : 0101 0110 ⊕
g5 = 1001 1000 (**98**)

g6= F(g5⊕ cv[((4*k)+3)mod 10]) ⊕g4
cv[((4*0)+3)mod 10] = cv[3] = 50

g5 = 98 : 1001 1000
cv[3] = 50 : 0101 0000 ⊕
1100 1000

F(1100 1000) = F(C8) = 29
F(C8) = 29 : 00101001
g4 = 4D : 0100 1101 ⊕
g6 = 0110 0100 (**64**)

W1(1) = G(W1(0)) ⊕ W4(0) ⊕ Counter

G(W1(0)) = 9864 : 1001 1000 0110 0100
W4(0) = 554E : 0101 0101 0100 1110 ⊕
1100 1101 0010 1010
Counter = 1 : 0000 0000 0000 0001
1100 1101 0010 1011 (CD2B)

W2(1) = G(W1(0)) = G(4E55) = 9864
W3(1) = W2(0) = 5241
W4(1) = W3(0) = 494E
K = 1 ; Counter = 2
Ciphertext : W1(1) + W2(1) + W3(1) + W4(1)
= CD2B 9864 5241 494E

**Putaran ke-2 (Rule A, K = 1, Counter = 2)**
G(W1(1)) = G(CD2B) = g5 + g6 = 21A7
g1 = Mid(W1(1),1,2) = CD
g2 = Mid(W1(1),3,2) = 2B
g3 = F(g2 ⊕ cv[(4*k) mod 10]) ⊕ g1
cv[(4*1)mod 10] = cv[4] = 54
g2 = 2B : 0010 1011
cv[4] = 54 : 0101 0100 ⊕
0111 1111

F(0111 1111) = F(7F) = 1D
F(7F) = 1D : 0001 1101
g1 = CD : 1100 1101 ⊕
g3 = 1101 0000(D0)

g4 = F(g3 ⊕ cv[((4*k)+1)mod 10]) ⊕ g2
cv[((4*1)+1)mod 10] = cv[5] = 4F
g3 = D0 : 1101 0000
cv[5] = 4F : 0100 1111 ⊕
1001 1111

F(1001 1111) = F(9F) = A4
F(9F) = A4 : 1010 0100
g2 = 2B : 0010 1011 ⊕
g4 = 1000 1111 (8F)

g5 = F(g4 ⊕ cv[((4*k)+2)mod 10]) ⊕ g3
cv[((4*1)+2)mod 10] = cv[6] = 4C

```
  g4  =  8F :  1000 1111
cv[6] =  4C :  0100 1100
-------------------------- ⊕
         1100 0011
```

F(1100 0011) = F(C3) = 9C

```
F(C3) =  9C :  1001 1100
g3    =  D0 :  1101 0000
-------------------------- ⊕
g5    =  0100 1100 (4C)
```

g6 = F(g5 ⊕ cv[((4*k)+3)mod 10]) ⊕ g4
cv[((4*1)+3)mod 10] = cv[7] = 4F

```
  g5  =  4C :  0100 1100
cv[7] =  4F :  0100 1111
-------------------------- ⊕
         0000 0011
```

F(0000 1001) = F(03) = 83

```
F(03) =  83 :  1000 0011
g4    =  8F :  1000 1111
-------------------------- ⊕
g6    =  0000 1100 (0C)
```

W1(2) = G(W1(1)) ⊕ W4(1) ⊕ Counter

```
G(W1(1)) =  4C0C :  01001100 0000 1100
W4(1)    =  494E :  0100 1001 0100 1110
-------------------------------------- ⊕
                    0000 0101 0100 0010
Counter  =     2 :  0000 0000 0000 0010
-------------------------------------- ⊕
                    0000 0101 0100 0000 (0540)
```

W2(2) = G(W1(1)) = G(CD2B) = 4C0C
W3(2) = W2(1) = 9864
W4(2) = W3(1) = 5241
K = 2 ; Counter = 3
Ciphertext : W1(2) + W2(2) + W3(2) + W4(2)
= 0540 4C0C 9864 5241
Hasil akhir pada putaran ke-32 adalah sebagai berikut :
8A F0 31 EE 1104 C2 C2 atau dalam karakter adalah Š ð1 î ◄ ˩ Â Â

c. Proses Dekripsi
Proses dekripsi dilakukan oleh penerima SMS. Proses dekripsi dilakukan berdasarkan kaidah-kaidah dekripsi yang berlaku pada algoritma *skipjack*.

*Ciphertext* SMS = Š ð1 î ◄ ˩ Â Â
Kunci = CRYPTOLOGY
Pengolahan Kunci :
Ubah dalam hexadecimal : 43525950544F4C4F4759
Bagi key menjadi 10 bagian sebagai berikut :

cv(0) = 43; cv(1) = 52; cv(2) = 59; cv(3) = 50;
cv(4) = 54 cv(5) = 4F; cv(6) = 4C; cv(7) = 4F;
cv(8) = 47; cv(9) = 59

Ubah *cipher* SMS dalam hexadesimal : 8AF031EE1104C2C2
Bagi *Ciphertext* menjadi 4 bagian(W1,W2,W3,W4) sebagai berikut :
W1(32) = 8AF0; W2(32) = 31EE; W3(32) = 1104;
W4(32) = C2C2

Putaran ke-1 (Rule B-1 , K = 32, Counter = 32)
$G^{-1}$ (W2(32)) = $G^{-1}$ (31EE) = g1 + g2 = 0974
g5 = Mid(W2(32),1,2) = 31
g6 = Mid(W2(32),3,2) = EE

g4 = F(g5 ⊕ cv[(4*(k-1)+3)mod 10]) ⊕ g6
cv[(4*(32-1)+3)mod 10] = cv[7] = 4F

g5 = 31 : 0011 0001
cv[7] = 4F : 0100 1111 ⊕
0111 1110

F(0111 1110) = F(7E) = 4A

F(7E) = 4A : 0100 1010
g6 = EE : 1110 1110 ⊕
g4 = 1010 0100 (A4)

g3 = F(g4 ⊕ cv[(4*(k-1)+2)mod 10]) ⊕ g5
cv[(4*(32-1)+2)mod 10] = cv[6] = 4C

g4 = A4 : 1010 0100
cv[6] = 4C : 0100 1100 ⊕
1110 1000

F(1110 1000) = F(E8) = 32

F(E8) = 32 : 0011 0010
g5 = 31 : 0011 0001 ⊕
g3 = 0000 0011 (03)

g2 = F(g3 ⊕ cv[(4*(k-1)+1)mod 10]) ⊕ g4
cv[(4*(32-1)+1)mod 10] = cv[5] = 4F

g3 = 03 : 0000 0011
cv[5] = 4F : 0100 1111 ⊕
0100 1100

F(0100 1100) = F(4C) = D0

F(4C) = D0 : 1101 0000
g4 = A4 : 1010 0100 ⊕
g2 = 0111 0100 (74)

g1 = F(g2 ⊕ cv[(4*k-1)mod 10]) ⊕ g3
cv[(4*(32-1))mod 10] = cv[4] = 54

g2 = 74 : 0111 0100
cv[4] = 54 : 0101 0100 ⊕
0010 0000

F(0010 0000) = F(20) = 0A

F(20) = 0A : 0000 1010
g3 = 03 : 0000 0011 ⊕
g1 = 0000 1001 (09)

W1(31) = $G^{-1}$ (W2(32)) = $G^{-1}$ (31EE) = 0974
W2(31) = $G^{-1}$ (W2(32)) ⊕W3(32) ⊕ Counter = 1850

$G^{-1}$ (W2(32)) = 0974 : 0000 1001 0111 0100
W3(32) = 1104 : 0001 0001 0000 0100 ⊕
0001 1000 0111 0000
Counter = 32 : 0000 0000 0010 0000 ⊕
0001 1000 0101 0000 (1850)

W3(31) = W4(32) = C2C2
W4(31) = W1(32) = 8AF0
K = 31 ; Counter = 31
Ciphertext : W1(31) + W2(31) + W3(31) + W4(31)
= 0974 1850 C2C2 8AF0

Hasil akhir proses dekripsi pada putaran ke-32 nilai-nilai hexadecimal SMS asli, yaitu 434F 4D50 5554 4552 atau dalam karakter : **COMPUTER**

## 4. KESIMPULAN

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

a. Aplikasi Pengamanan SMS menggunakan algoritma *Skipjack* mempunyai dua teknik pembacaan yaitu teknik enkripsi (mengubah teks SMS asli menjadi *teks* SMS yang tidak dapat dipahami makananya) dan teknik dekripsi (mengubah *cipher* SMS menjadi teks SMS asli).
b. Penyandian SMS berdasarkan algoritma *Skipjack,* dapat dapat mengoptimalkan keamanan SMS karena algoritma ini melakukan proses penyandian sebanyak 32 kali putaran algoritma dibandingkan dengan algoritma yang lain sehingga data sulit untuk dibuka kerahasiaannya.
c. Algoritma *Skipjack* menyadiakan suatu layanan penukaran data dengan tingkat keamanan yang cukup tinggi.

## DAFTAR PUSTAK

[1] 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.
[2] F. Purwaningsih and M. Badrul, "Penerapan algoritma huffman untuk aplikasi pengamanan sms berbasis android," *J. PROSISKO*, vol. 4, no. 2, pp. 60–66, 2017.

[3] G. P. R. S. Fettiana, Elvina, "PEMBUATAN ALGORITMA ENKRIPSI DES SMS BERBASIS MOBILE," vol. VI, no. 2, pp. 151–166, 2015.
[4] E. Setyaningsih, *Kriptografi & Implementasinya Menggunakan Matlab*, Yogyakarta:Andi, 2015.
[5] Suprianto, "SISTEM PENGKODEAN DATA PADA FILE TEKS PADA KEAMANAN INFORMASI DENGAN MENGGUNAKAN METODE SKIPJACK," vol. 1, no. 2, pp. 105–118, 2007.
[6] E. Setyaningsih, *Kriptografi & Implementasinya Menggunakan Matlab*, Yogyakarta:Andi, 2015.
[7] Rifki Sadikin, *Kriptografi Untuk Keamanan Jaringan dan Implementasinya dalam Bahasa Java*, Yogyakarta: Andi, 2012.
[8] 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.
[9] Hartono, "Aplikasi Pengamanan Data Menggunakan Metode Skipjack", *STMIK IBBI*, pp.39-50, 2009.