Perancangan Aplikasi Pengamanan Data Text dengan Menggunakan Algoritma Simetri TEA (Tiny Encryption Algorithma)

Tini Octaviani Purba[1], Abdul Sani Sembiring [2]

[1,2] STMIK Budi Darma, Jl. Sisingamangaraja No.338 Simpang Limun Medan

ARTICLE INFORMATION

Received: Agustus 27,2019
Revised: September 20,2019
Available online: Oktober 05,2019

KEYWORDS

Kriptografi, Algortima TEA, Pengamanan Teks.

CORRESPONDENCE

Phone: +6285358014334

E-mail: tini.octaviani@gmail

A B S T R A K

Keamanan data merupakan salah satu aspek terpenting dalam teknologi informasi. Dengan tingkat keamanan yang tinggi, diharapkan informasi yang disajikan dapat terjaga keasliannya. Pada tugas akhir ini dibentuk suatu system yang mengamankan data dan informasi yang tersimpan pada computer dari gangguan para kriptanalis.Tahapan yang penulis lakukan untuk melakukan proses pembentukan sistem tersebut meliputi tahapan analisa permasalahan, algoritma dan flowchart beserta pemodelan struktur program dan desain antar muka aplikasi, sehingga aplikasi yang terbentuk menjadi mudah dipergunakan dan memiliki fungsi yang optimal. Dengan menggunakan Algoritma TEA yang merupakan algortim. Algoritma ini merupakan algoritma penyandian block cipher yang dirancang untuk penggunaan memori yang seminimal mungkin dengan kecepatan proses yang maksimal. Sistem ini dibangun menggunakan bahasa pemrograman Visual Basic .Net 2008..

1. PENDAHULUAN

Keamanan data adalah hal yang sangat penting, apalagi data yang dikirimkan adalah pesan yang sangat rahasia. Berbagai usaha dilakukan untuk menjamin agar pesan rahasia yang dikirimkan tersebut tidak dapat diakses oleh pihak lain. Hal tersebut tentu akan menimbulkan resiko apabila informasi yang sensitif dan berharga tersebut diakses oleh orang-orang yang tidak berhak[1]. Apabila hal tersebut sampai terjadi, kemungkinan besar akan merugikan bahkan membahayakan orang yang mengirim pesan atau menerima pesan, maupun organisasinya[2]. Informasi yang terkandung di dalamnyapun dapat saja berubah sehingga menyebabkan salah penafsiran oleh penerima pesan. Selain itu data yang di curi tersebut akan memiliki kemungkinan rusak bahkan hilang.

Kriptografi *(crypthography)* berasal dari bahasa Yunani yaitu dari dua suku kata *crypto* dan *graphia. Crypto* artinya menyembunyikan, sedangkan graphia artinya ilmu. Kriptografi, secara umum adalah ilmu dan seni untuk menjaga kerahasiaan berita yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi seperti kerahasiaan data, keabsahan data, integritas data, serta autentikasi data, yang dilakukan oleh seorang "Kriptographer"[3]. Kriptografi telah menjadi suatu bagian yang tidak dapat dipisahkan dari sistem keamanan jaringan.

Ada berbagai algoritma kriptografi yang sekarang ini telah dan sedang dikembangkan, satu diantaranya adalah algoritma kunci simetris ataupun asimetris (pembagian berdasarkan kunci). Kriptografi itu sendiri juga terbagi menjadi kriptografi klasik dan modern. Kriptografi klasik termasuk kedalam kriptografi kunci simetri. Data atau informasi yang akan diamankan menggunakan kriptografi disebut plaintext[4]. Proses menyandikan plaintext tersebut disebut enkripsi, yang menghasilkan pesan yang tersandikan disebut ciphertext. Proses untuk mengembalikan ciphertext menjadi plaintext disebut dekripsi. Proses enkripsi dan dekripsi biasanya menggunakan kunci atau key.

Salah satu algoritma kriptografi data adalah Tiny Encryption Algorithma (TEA). Tiny Encryption Algorithma (TEA) merupakan suatu algoritma sandi yang diciptakan oleh David Wheeler dan Roger Needham (Computer Laboratory Cambridge University, England november 1994). Algoritma ini merupakan algoritma penyandian block cipher yang dirancang untuk penggunaan memori yang seminimal mungkin dengan kecepatan proses yang maksimal[5], [6].

2. LANDASAN TEORI

Kriptografi merupakan suatu ilmu yang mempelajari tentang bagaimana merahasiakan suatu informasi penting ke dalam suatu bentuk yang tidak dapat dibaca oleh siapapun serta mengembalikannya kembali menjadi informasi semula dengan menggunakan berbagai macam teknik yang telah ada sehingga informasi tersebut tidak dapat diketahui oleh pihak manapun yang bukan pemilik atau yang tidak berkepentingan. Sisi lain dari kriptografi ialah kriptanalisis (*Criptanaysis*) yang merupakan studi tentang bagaimana memecahkan mekanisme kriptografi[1].

Gambar 1 : Tulisan *Hieroglyph*[7]

Tiny Encription Algorithma (*TEA*) merupakan suatu algoritma sandi yang diciptakan oleh *David Wheeler* dan *Roger Needham* dari *Computer Loboratory*, *Cambridge Univercity, England* pada bulan November 1994. Algoritma ini merupakan algoritma penyandian *block cipher* yang dirancang untuk penggunaan memory yang seminimal mungkin dengan kecepatan proses yang maksimal[6], [8].

Sistem penyandian *TEA* menggunakan proses *feistel network* dengan menambahkan fungsi matematik berupa penambahan dan pengurangan sebagai operator pembalik selain *XOR*. Hal ini dimaksudkan untuk menciptakan sifat *non linearitas*. Pergeseran dua arah (ke kiri dan ke kanan) menyebabkan semua *bit* kunci dan data bercampur secara berulang ulang.

TEA memproses 64 *bit input* sekali waktu dan menghasilkan 64 *bit output*. *TEA* menyimpan 64 *bit input* kedalam L_0 dan R_0 masing masing 32 *bit*. Sedangkan 128 *bit* kunci disimpan kedalam k[0], k[1], k[2], dan k[3] yang masing masing berisi 32 *bit*. Diharapkan teknik ini cukup dapat mencegah penggunaan *teknik exshautive search* secara efektif[9]. Hasil *output*nya akan disimpan dalam L_{16} dan R_{16}.

Bilangan delta konstanta yang digunakan adalah 9E3779B9, Dimana Bilangan *delta* berasal dari *golden number*, digunakan *delta*=$((5/4^{1/2} -1/2 \cdot 0.618034)2^{32}$. Suatu bilangan *delta* ganda yang berbeda digunakan dalam setiap *roundnya* sehingga tidak ada *bit* dari perkalian yang tidak berubah secara teratur. Berbeda dengan sruktur *feistel* yang semula hanya mengoperasikan satu sisi yaitu sisi sebelah kanan dengan sebuah fungsi F, pada algoritma *TEA* kedua sisi dioperasikan dengan sebuah fungsi yang sama (Mugi, 2014, 7).

a. Proses Enkripsi Algoritma TEA

Proses diawali dengan input *bit* teks terang sebanyak 64 *bit*. Kemudian 64 *bit* teks terang tersebut dibagi, yaitu dua. Sisi kiri (L_0) sebanyak 32 *bit* dan sisi kanan (R_0) sebanyak 32 *bit*. Setiap bagianteks terang akan dioperasikan sendiri-sendiri. R_0 (z) akan digeser kekiri sebanyak empat (4) kali dan ditambahkan dengan kunci k[0]. Sementara itu z ditambah dengan sum *(delta)* yang merupakan konstanta. Hasil penambahan ini di *XOR*kan dengan penambahan sebelumnya. Kemudian di *XOR* kan dengan hasil penambahan antara Z yang digeser kekenan sebanyak lima (5) kali dengan kunci k[1]. Hasil tersebut kemudian ditambahkan dengan L_0 (y) yang akan menjadi R_1[5].

Sisi sebelah kiri akan mengalami proses yang sama dengan sisi sebelah kanan. L_0 (y) akan digeser kekiri sebanyak empat (4) kali lalu ditambahkan dengan kunci k[2]. Sementara itu, Y ditambah dengan *sum (delta)*. Hasil penambahan ini di *XOR* kan dengan penambahan sebelumnya. Kemudian di *XOR* kan dengan hasil penambahan antara Y yang digeser ke kanan sebanyak lima (5) kali dengan kunci k[3]. Hasil tersebut kemudian ditambahkan dengan R_0 (Z) yang akan menjadi L_1 [5].

Berikut ini langkah-langkah penyandian dengan algoritma *TEA* dalam satu *cycle/dua round* [6]yaitu :

1. Pergeseran*(shift)*
   Blok teks terang pada kedua sisi yang masing masing sebanyak 32 *bit* akan digeser kekiri sebanyak empat (4) kali dan digeser ke kanan sebanyak lima (5) kali.
2. Penambahan
   Setelah digeser kekiri stsu kekanan, maka *Y* dan *Z* yang sedah digeser akan ditambahkan dengan kunci *k[0]-k[3]*. Sedangkan *Y* dan *Z* awal akan ditambahkan dengan *sum (delta)*.
3. Peng-*XOR*-an
   Setelah dioperasikan dengan penambahan pada masing-masing register maka akan dilakukan peng *XOR*an dengan rumus untuk satu *round*.
   dalam hal ini *sum=sum+delta*.
   Hasil penyandian dalam satu *cycle* satu *blok* teks terang 64 *bit* menjadi 64 *bit* teks sandi adalah dengan menggabungkan *y* dan *z*. Untuk penyandian pada *cycle* berikutnya *y* dan *z* ditukar posisinya, sehingga y_1 menjadi z_1 dan z_1 menjadi y_1 terus dilanjutkan proses seperti langkah-langkah di atas sampai dengan 16 *cycle* (32 *round*).
4. *Key Schedule*
   Pada algoritma *TEA*, *key schedulenya* sangat sederhana. Yaitu kunci k[0] dan k[1] konstan digunakan untuk *round* ganjil sedangkan kunci k[2] dan k[3] konstan digunakan untuk *round* genap.

b. Proses Dekripsi Algorithm TEA

Proses dekripsi pada algoritma *TEA*sama halnya dengan proses enkripsi. Hanya saja terjadi perbedaan pada penjadwalan kuncinya yaitu pada proses enkripsi untuk *cipher R* yang mengalami pergeseran *bit* ke kiri sebanyak 4 *bit* digunakan kunci k[0] pada proses dekripsi digunakan kunci k[1], untuk *cipher R* yang mengalami pergeseran ke kanan sebanyak 5 *bit*

menggunakan kunci [1] pada proses dekripsi menggunakan kunci k[0]. begitu juga hal nya dengan *cipher L*, pada proses enkripsi untuk *cipher L* yang mengalami pergeseran ke kiri sebanyak 4 *bit* menggunakan kunci k[2] pada proses dekripsi digunakan kunci k[3]. Untuk *cipher L* yang mengalami pergeseran ke kanan sebanyak 5 *bit* digunakan kunci k[3] pada proses dekripsi digunakan kunci k[2].

Adapun beberapa keunggulan dari algoritma *Tiny Encryption Algorithma* (TEA) ini [5]adalah :

1. pada algoritma *Tiny Encryption Algorithma* (TEA) panjang kuncinya 128 *bit*, merupakan jumlah kunci yang cukup panjang untuk algoritma kriptografi modern saat ini yang dapat menahan serangan *kriptanalis.*
2. Teknik yang digunakan *TEA* cukup baik, yaitu pada setiap prosesnya menggunkan jaringan *Feisel* yang memuat Operasi permutasi, subtitusi dan modular arithmatic berupa *XOR* dan penambahan bilangan *delta* yang diharapkan dari operasi tersebut menciptakan efek difusi dan konfusi yang baik, karena semakin baik efek difusi dan konfusi yang dihasilkan suatu algoritma makin semakin baik pula tingkat keamanannya.
3. Ukuran *blok input TEA* yaitu 64 *bit,* sebuah jumlah yang cukup panjang untuk menghindari analisis pemecahan kode dan cukup kecil, agar bekerja dengan cepat.
4. Tidak membutuhkan *S-Box* dan *P-Bok* tersebut hanya diketahui oleh NSA *(National Security Agency)* dan diubah menurut saran dari NSA, sehingga jika *S-Box* dan *P-Box* tersebut diubah maka sangat mungkin sekali algoritma yang digunakan akan mudah dibobol. Selain itu, juga dapat meminimalkan penggunaan *memory* pada saat melakukan proses enkripsi dan dekripsi sehingga dapat memaksimalkan proses.
5. Algoritma *TEA* diketahui sangat kuat terhadap metode penyerangan berupa hanya *ciphertext* yang diketahui, *plaintext* yang diketahui dan *plaintext* terpilih

Sedangkan kelemahan dari Algoritma *Tiny Encryption Algorithma* (TEA) ini adalah karena *TEA* ini termasuk kedalam kelompok algoritma simetri, maka masih rentan untuk dibobol, karena dalam algoritma simetri masalah utama memang terletak dari segi pendistribusian kuncinya, dimana harus benar-benar aman pada saat mendistribusikan kunci yang akan digunakan berdasarkan data yang didapat, *estimasi* proses enkripsi dan dekripsi algoritma *TEA* yang dibandingkan dengan algoritma simetri lainnya.

3. HASIL DAN PEMBAHASAN

Sebelum proses perancangan dimulai, maka diperlukanlah beberapa analisis terhadap sistem, metode ataupun teknik-teknik yang digunakan dalam tahap perancangan. Analisa dapat memberi uraian secara utuh tentang masalah yang sedang dianalisa dengan melakukan identifikasi dan evaluasi terutama hambatan-hambatan yang terjadi serta kebutuhkan dalam memberi solusi penyelesaian masalah yang sedang dibahas dalam melakukan analisa terhadap data teks dan proses pengamanan data baik enkripsi maupun dekripsi.

a. Proses Enkripsi Algoritma TEA

Berikut ini dijelaskan proses enkripsi pada data teks menggunakan algoritma *TEA*.

1. *Input* Plainteks
   Plainteks =TINIOKTA
2. Partisi Plainteks
   Bagi plainteks menjadi 2 blok ke dalam blok R dan blok *L*
   L_0 = TINI
   R_0 = OKTA
   T 84 = 01010100
   I 73 = 01001001
   N 78 = 01001110
   I 73 = 01001001
   O 79 = 01001111
   K 75 = 01001011
   T 84 = 01010100
   A 65 = 01000001
   Sehingga didapat :
   Cipher $L_0(z)$ = 01010100 01001001 01001110 01001001
   Cipher $R_0(y)$ = 01001111 01001011 01010100 01000001
3. *Input* Kunci
   Kunci = VIAPIP BUDIDARMA
4. Partisi Kunci
   Bagi kunci menjadi 4 blok ke dalam blok k[0], k[1], k[2] dan k[4] :
   k[0] =VIAP
   k[1] = IPspcB
   k[2] = UDID
   k[3] = ARMA
   V 86 = 01010110
   I 73 = 01001001
   A 65 = 01000001
   P 80 = 01010000

I 73 = 01001001
P 80 = 01010000
Spc 32 = 00100000
B 66 = 01000010
U 85 = 01010101
D 68 = 01000100
I 73 = 01001001
D 68 = 01000100
A 65 = 01000001
R 82 = 01010010
M 77 = 01001101
A 65 = 01000001

Sehingga didapat :

k[0] = 01010110 01001001 01000001 01001001
k[1] = 01001001 01010000 00100000 01000010
k[2] = 01010101 01000100 01001001 01000100
k[3] = 01000001 01010010 01001101 01000001

5. Cipher $R_0(Z)$ akan mengalami pergeseran *bit* ke kiri sebanyak 4 *bit* dan pergeseran ke kanan sebayak 5 *bit*.
   Cipher R_0 : 01010100 01001001 01001110 01001001
   Menjadi :
   Zsl (Z *shift left*) : 01000100 10010100 11100100 10010101
   Zsr (Z *shift right*) : 01001010 10100010 01001010 01110010

6. Zsl ditambahkan dengan kunci k[0]:
   Zsl : 01000100 10010100 11100100 10010101
   K[0] : 01010110 01001001 01000001 01001001
   +
   10011010 11011110 00100101 11011110
7. Zsr ditambahkan dengan kunci k[1] :
   Zsr : 01001010 10100010 01001010 01110010
   K[1] : 01001001 01010000 00100000 01000010
   +
   10010011 11110010 01101010 10110100
8. Kemudian *cipher* $R_0(Z)$ yang tidak mengalami pergeseran *bit* ditambahkan dengan bilangan delta, dimana bilangan delta yang digunakan secara konstanta
   yaitu : 9E3779B9 atau dalam biner 10011110 00110111 01111001 10111001
   R(Z) : 01010100 01001001 01001110 01001001
   Delta : 10011110 00110111 01111001 10111001
   +
   11110010 10000000 11001000 00000010
9. Hasil R(Z) + Delta di *xor*kan dengan *cipher* Zsl yang ditambah K[0].
   R(Z)+Delta : 11110010 10000000 11001000 00000010
   Zsl + k[0] : 10011010 11011110 00100101 11011110
   xor
   01101000 01011110 11101101 11011100
10. Hasil sebelumnya di *xor*kan dengan *cipher* Zsr yang ditambah K[1] :
    R(Z)+Delta xor k[0] : 01101000 01011110 11101101 11011100
    Zsl + k[1] : 10010011 11110010 01101010 10110100
    xor
    11111011 10101100 10000111 01101100
11. *Cipher* $L_0(Y)$ proses yang terjadi pada dasarnya sama seperti pada *cipher* R(z), yaitu *cipher* L(y) juga yang mengalami pergeseran *bit* ke kiri sebanyak 4 *bit* dan ke kanan sebanyak 5 *bit*.
    Cipher $L_0(y)$: 01001111 01001011 01010100 01000001
    Menjadi
    Lsl (L Shift left) : 11110100 10110101 01000100 00010100
    Lsr (L Shift right) : 00001010 01111010 01011010 10100010
12. Lsl ditambahkan dengan kunci K[2] :
    Lsl : 11110100 10110101 01000100 00010100
    K[2] : 01010101 01000100 01001001 01000100
    +
    01001001 11111001 10001101 01011000
13. Lsr ditambahkan dengan k[3] :
    Lsr : 00001010 01111010 01011010 10100010
    K[3] : 01000001 01010010 01001101 01000001

+
01001011 11001100 10100111 11100011

14. Kemudian *cipher* L(Y) yang tidak mengalami pergeseran *bit* ditambahkan dengan bilangan delta yang digunakan secara konstanta yaitu : 9E3779B9 atau dalam bilangan biner 10011110 00110111 01111001 10111001
    L(y) : 01001111 01001011 01010100 01000001
    Delta : 10011110 00110111 01111001 10111001
    +
    11101101 10000010 11001101 11111010
15. Hasil L(y)+Delta di xorkan dengan cipher Lsl yang ditambah K[2] :
    L(y) + Delta : 11101101 10000010 11001101 11111010
    Lsl +K[2] : 01001001 11111001 10001101 01011000
    xor
    10100100 01111011 01000000 10100010
16. Hasil sebelum di xorkan dengan cipher Lsl yang ditambahkan K[3]:
    L(Y)+delta xor K[3] : 10100100 01111011 01000000 10100010
    Lsr + K[3] : 10001011 11001100 10100111 11100011
    xor
    11101111 10110111 11100111 01000001

17. Hasil akhir *cipher* R(Z) ditambahkan dengan *chiper* L(Y) yang tidak mengalami pergeseran, yang mana hasilnya akan dijadikan *cipher* L1(Y1) untuk *round* berikutnya. Demikian juga halnya hasil akhir pada *cipher* L(Y) akan ditambahkan dengan *cipher* R(Z) yang tidak mengalami pergeseran yang akan dijadikan *cipher* R1(Z1) pada *round* berikutnya.

    $R_0(Z)$: 11111011 10101100 10000111 01101000
    $L_0(Y)$: 01001111 01001011 01010100 01000001
    +
    01001010 11110111 11011011 10101001 = $L_1(Y_1)$
    $L_0(Y)$: 11101111 10110111 11100111 01000001
    $R_0(Z)$: 01010100 01001001 01001110 01001001
    +
    01000011 00000001 00110101 10001010 = $R_1(Z_1)$

18. **Round 1**
    $Y_1 = y + (((z<<4)+k[0])^z+sum^((z>>5)+k[1]))$
    Y_1=01001010 11110111 11011011 10101001 +((01000100 10010100 11100100 10010101 + 01010110 01001001 01000001 01001001)^ 11110010 10000000 11001000 00000010 ^ ((01001010 10100010 01001010 01110010 + 01001001 01010000 00100000 01000010))
    Y_1= 01000101 10100100 01100011 00010001
    69 164 99 17

    Z_1= Z + (((y<<4)+k[2]^y+sum^((y>>5)+k[3]))
    Z_1= 01000011 00000001 00110101 10001010 + ((11110100 10110101 01000100 00010100 + 01010101 01000100 01001001 010001)^ 11101101 10000010 11001101 11111010 ^ ((00001010 01111010 01011010 10100010 + 01000001 01010010 01001101 01000001))
    Z_1= 00110011 10110111 00011000 01000011
    51 183 24 67

    Round 2 :
    $Y_2 = Y_1 + (((z<<4)+k[0])^z+sum^((z>>5)+k[1]))$
    Y_2 = 01000101 10100100 01100011 00010001 + ((01000100 10010100 11100100 10010101 + 01010110 01001001 01000001 11101101)^ 11110010 10000000 11001000 00000010 ^((01001010 10100010 01001010 01110010 + 01001001 01010000 00100000 01000010))
    Y_2 =01000000 01010000 11101010 01111001
    64 80 234 121
    $Z_2= Z_1 + (((y<<4)+k[2]^y+sum^((y>>5)+k[3]))$
    Z_2 = 00110011 10110111 00011000 01000011 + ((11110100 10110101 01000100 00010100 + 01010101 01000100 01001001 01000100)^ 11101101 10000010 11001101 11111010 ^ ((00001010 11111010 01011010 10100010 + 01000001 01010010 01001101 01000001))
    Z_2=00101110 01100011 10011111 10101011
    46 99 159 171

Perhitungan sampai round 32, akan diperoleh Cipherteks sebagai berikut :
Cipherteks = ¯ Ü A A Ø î ö s (175-220-65-65-157-238-246-115).

b. Proses Dekripsi TEA

Berikut ini dijelaskan proses dekripsi pada data text menggunakan algoritma *TEA*.

1. *Input* Cipherteks
   Cipherteks : ¯ Ü A A(157) î ö s
2. Partisi Cipherteks

Bagi cipherteks menjadi 2 blok ke dalam blok R dan blok L:
L_0= ¯ Ü A A
R_0= (157) î ö s

¯	175= 10101111
Ü	220= 11011100
A	65 = 01000001
A	65 = 01000001
157	157= 10011101
Î	238= 11101110
Ö	246= 11110110
s	115= 01110011

Sehingga didapat :
Cipher L_0(y)= 10101111 11011100 01000001 01000001
Cipher R_0(z)= 10011101 11101000 11110110 01110011

3. *Input* Kunci
Kunci = VIAPIP BUDIDARMA
4. Partisi Kunci
Bagi kunci menjadi 4 blok ke dalam blok k[0], k[1], k[2] dan k[4] :

k[0]	=	VIAP
k[1]	=	IPspcB
k[2]	=	UDID
k[3]	=	ARMA
V	86	01010110
I	73	01001001
A	65	01000001
P	80	01010000
I	73	01001001
P	80	01010000
Spc	32	00100000
B	66	01000010
U	85	01010101
D	68	01000100
I	73	01001001
D	68	01000100
A	65	01000001
R	82	01010010
M	77	01001101
A	65	01000001

Sehingga didapat :

k[0]	=	01010110 01001001 01000001 01001001
k[1]	=	01001001 01010000 00100000 01000010
k[2]	=	01010101 01000100 01001001 01000100
k[3]	=	01000001 01010010 01001101 01000001

5. *Cipher* R_0(z) akan mengalami pergeseran *bit* ke kiri sebanyak 4 *bit* dan pergeseran ke kanan sebayak 5 *bit*.
Cipher R_0 : 10101111 11011100 01000001 01000001
Menjadi :
Zsl (Z *shift left*) : 11111101 11000100 00010100 00011010
Zsr (Z *shift right*) : 00001101 01111110 11100010 00001010
6. Zsl ditambahkan dengan kunci k[1] :

Zsl	:	11111101	11000100	00010100	00011010
K[1]	:	01001001	01010000	00100000	01000010 +
		01000110	00010100	00110100	01011100

7. Zsr ditambahkan dengan kunci k[0] :

Zsr	:	00001101	01111110	11100010	00001010
K[0]	:	01010110	01001001	01000001	01001001 +
		11110010	00111000	10111001	01010011

8. Kemudian cipher R(Z) yang tidak mengalami pergeseran bit ditambahkan dengan bilangan delta, dimana bilangan delta yang digunakan secara konstanta yaitu : 9E3779B9 atau dalam biner 10011110 00110111 01111001 10111001

R(Z):		10101111	11011100	01000001	01000001
Delta	:	10011110	00110111	01111001	10111001 +
		00111011	00100110	01110000	11101010

9. Hasil R(Z) + Delta di XOR kan dengan cipher Zsl yang ditambah K[1]

R(Z) + Delta		01001101	00011110	10111010	11111010
Zsl + K[1]		01000110	00010100	00110100	01011100 +
		01111101	00110010	01000100	01110000

10. Hasil sebelumnya di XOR kan dengan cipher Zsr yang ditambah K[0] :

R(Z)+Delta Xor K[0] :01111101 00110010 01000100 01110000
Zsr+K[0] :11110010 00111000 10111001 00000011 xor
10001111 00001010 11111101 01110011

11. Cipher L(Y) proses yang terjadi pada dasarnya sama seperti pada cipher R(Z), yaitu cipher L(Y) juga mengalami pergeseran bit ke kiri sebanyak 4 bit dan ke kanan sebanyak 5 bit. Cipher L : 10011101 11101110 11110110 01110011
Menjadi :
Lsl (L Shift left) : 11011110 11101111 01100111 00111001
Lsr (L Shift right) : 10011100 11101111 01110111 10110011
12. Lsl ditambahkan dengan kunci k[3] :

Lsl	:	11011110	11101111	01100111	00111001
K[3]	:	01000001	01010010	01001101	01000001 +
		00011111	01000001	10110100	01111010

13. Lsr ditambahkan dengan kunci k[2] :

Lsr	:	10011100	11101111	01110111	10110011
K[2]	:	01010101	01000100	01001001	01000100 +
		11110001	00110011	11000000	11110111

14. Kemudian cipher L(Y) yang tidak mengalami pergeseran *bit* ditambahkan dengan bilangan delta, dimana bilangan delta yang digunakan secara konstanta yaitu : 9E3779B9 atau dalam biner 10011110 00110111 01111001 10111001

L(Y):	10011101	11101110	11110110	01110011	
Delta	:	10011110	00110111	01111001	10111001 +
		01001101	00010011	10111010	11111010

15. Hasil L(Y) + Delta di XOR kan dengan cipher Lsl yang ditambah K[3]

L(Y) + Delta	01001101	00010011	10111010	11111010
Lsl + K[3]	00011111	01000001	10110100	01111010 xor
	01010010	01010010	00001110	10000000

16. Hasil sebelumnya di XOR kan dengan cipher Lsr yang ditambah K[2] :
L(Y) + Delta Xor K [3] : 01010010 01010010 00001110 10000000
Lsr + K[2] :11110001 00110011 11000000 11110111 xor
11110001 00110011 11000000 11110111
17. Hasil akhir *cipher R(Z)* ditambahkan dengan *cipher* L(Y) yang tidak mengalami pergeseran, yang mana hasilnya akan dijadikan cipher L1(Y1) untuk *round* berikutnya. Demikian juga halnya hasil akhir pada cipher L(Y) akan ditambahkan dengan cipher R(Z) yang tidak mengalami pergeseran yang akan dijadikan cipher R1 (Z1) pada *round* berikutnya.

R(Z):	10001111	00001010	11111101	01110011
L0(Y):	10011101	11101110	11110110	01110011 +
	00101100 11111001	11110011	11100110	L1(Y1)
L(Y):	11110001	00110011	11000000	11110111
R0(Z):	10101111	11011100	01000001	01000001 +
	10100000	00010000	00000010	0011100R1 (Y1)

18. Round 1
Y1= y + (((z<<44)+k[0])^z+sum^((z>>5)+k[1]))
Y1 =00101100 11111001 11110011 11100110+(((11111101 11000100 00010100 00011010+ 01010110 01001001 01000001 01010000)^ 00111011 00100110 01110000 00101100 ^((10011100 11101111 01110111 10110011 + 01001001 01010000 00100000 01000010))
Y1 = 00101001 10111110 01110000 01001001
41 190 112 73
Z1 = Z + (((y<<4)+k[2])^y+sum^((y>>5)+k[3]))
Z1 = 10100000 00010000 00000010 00111000+ (((11011110 11101111 01100111 00111001 + 01010101 01000100 01001001 01000100)^ 01001101 00010011 10111010 11111010^ ((1001110011101111 01110111 10110011+ 01000001 01010010 01001101 01000001))
Z1 = 01101010 10110111 01001101 00111001
106 183 77 57

Round 2

Y2= y1 + (((z<<44)+k[0])^z+sum^((z>>5)+k[1]))
Y2 = 00101001 10111110 01110000 01001001 + (((11111101 11000100 00010100 00011010 + 01010110 01001001 01000001 01010000)^ 00111011 00100110 01110000 00101100 ^ ((10011100 11101111 01110111 10110011 + 01001001 01010000 00100000 01000010))
Y2 = 00100100 11010010 00110101 00110111
36 210 53 55
Z2 = Z1 + (((y<<4)+k[2])^y+sum^((y>>5)+k[3]))
Z2 = 1101010 10110111 01001101 00111001 +(((11011110 11101111 01100111 00111001 + 01010101 01000100 01001001 01000100)^ 01001101 00010011 10111010 11111010^ ((10011100 11101111 01110111 10110011 + 01000001 01010010 01001101 01000001))
Z2= 11001000 10010110 01001110 01100010
200 150 78 98

Pada akhir perhitungan dekripsi TEA sebanyak 32 round ini diperoleh plainteks sebagai berikut : “TINIOKTA”.

5. KESIMPULAN

Dari pembahasan yang penulis lakukan pada penulisan skripsi ini dapat ditarik beberapa kesimpulan antara lain:

1. Proses memecahkan permasalahan keamanan data teks menggunakan algoritma *Tiny Encryption Algorithm* (TEA) yaitu dengan menyandikan teks dengan proses enkripsi yang memiliki panjang plainteks 64 *bit* dan panjang kunci 128 *bit.*
2. Penerapan algoritma *Tiny Encryption Algorithm* (TEA) dalam mengamankan data teks menggunakan proses menambahkan fungsi matematik untuk menciptakan sifat *non-linearitas* kemudian pergesaran dua arah (ke kiri dan ke kanan) menyebabkan semua *bit* kunci dan data bercampur secara berulang-ulang sebaganyak 32 putaran sehingga dihasilkan *ciphertext*.
3. Perancangan aplikasi pengamanan data teks diawali dengan pemodelan sistem. Alat bantu yang digunakan untuk memodelkan sistem adalah *Unified Modelling Language* (UML) dan activity diagram kemudian dilakukan perancangan *interface* dengan *Visual Basic .Net 2008*.

DAFTAR PUSTAKA

[1] R. Munir, "Kriptografi," *Inform. Bandung*, 2006.

[2] A. M. Hasibuan, "Rancang Bangun Aplikasi Keamanan Data Menggunakan Metode AES Pada Smartphone," *MEANS (Media Inf. Anal. dan Sist.*, vol. 2, no. 1, pp. 29–35, Jun. 2017.

[3] A. W. Simatupang, "Aplikasi Pengamanan Data Gambar Dengan Menerapkan Algoritma Vigenere Chiper," *MEANS (Media Informasi Analisa dan Sistem)*, 2017. [Online]. Available: http://ejournal.ust.ac.id/index.php/Jurnal_Means/article/view/26/tr44. [Accessed: 02-Feb-2020].

[4] T. Limbong, "Pengujian Kriptografi Klasik Caesar Chipper Menggunakan Matlab," *no. Sept.*, vol. 2017, 2015.

[5] M. Qamal, "Kriptografi File Citra Menggunakan Algoritma Tea (Tiny Encryption Algorithm)," *TECHSI - J. Penelit. Tek. Inform.*, 2014.

[6] R. N. Ibrahim, "PERANGKAT LUNAK KEAMANAN DATA MENGGUNAKAN ALGORITMA KRIPTOGRAFI SIMETRI TINY ENCRIPTION ALGORITHM (TEA)," *JURNAL COMPUTECH & BISNIS*, 2019. [Online]. Available: http://jurnal.stmik-mi.ac.id/index.php/jcb/article/view/191/215. [Accessed: 02-Feb-2020].

[7] R. Munir, "Algoritma Knapsack," pp. 0–18, 2004.

[8] M. T. Suryadi *et al.*, "SMS Security System on Mobile Devices Using Tiny Encryption Algorithm," *IOP Conf. Ser. J. Phys. Conf. Ser.*, vol. 1007, p. 12037, 2018.

[9] N. Nurdin, "IMPLEMENTASI ALGORITMA TEA DAN FUNGSI HASH MD4 UNTU ENKRIPSI DAN DEKRIPSI DATA," *TECHSI - J. Tek. Inform.*, vol. 5, no. 1, 2013.