Seminar Nasional Teknologi Komputer & Sains (SAINTEKS)
SAINTEKS 2020
ISBN: 978-602-52720-7-3
Februari 2020
Hal 490-494

Implementasi Algoritma Tiny Encryption Algorithm Untuk Pengamanan File PDF

Aulia Husna, Juli Mawarni Hulu, Yuliani Susiawati Hondro

Program Studi Teknik Informatika, STMIK Budi Darma, Medan, Indonesia
Email:[1]aulia.bahri@gmail.com, [2]julimawarnihulu@gmail.com, [3] yulianisusiawatihondro@gmail.com

Abstrak−Kriptografi merupakan teknik untuk pengamanan data dengan cara menyandikan pesan agar maknanya tidak mudah dimengerti. Pengaman *file* akan mengurasi risiko terhadap penyadapan yang dilakukan oleh pihak yang tidak berkepentingan pada proses pengiriman. Dalam pengiriman data, *file* PDF *(Portable Document Format)* banyak gunakan dalam bertukar informasi, sehingga perlu adanya pengamanan terlebih dahulu. Implementasi kriptografi untuk pengamanan *file* banyak digunakan dalam penelitian dan salah satu contohnya Algoritma TEA *(Tiny Encryption Algorithm)*. Implementasi algoritma TEA memiliki keamanan yang sangat tinggi dalam pengamanan *file*.

Kata Kunci: Kriptografi, Pengamanan, Implementasi, Algoritma TEA, File PDF

1. PENDAHULUAN

Pengamanan dan menjaga kerahasiaan file sangat perlu diperhatikan terutama untuk file yang bersifat rahasia yang isinya hanya boleh diketahui oleh pihak tertentu. Dalam pengiriman data, file PDF (File Portable Document Format) banyak gunakan dalam bertukar informasi, sehingga perlu adanya pengamanan terlebih dahulu. Pengamanan file akan mengurangi risiko penyadapan yang dilakukan pihak yang tidak berkepentingan pada proses pengiriman. Pengaman file dapat dilakukan dengan teknik kriptografi yaitu dengan menyandikan data sehingga informasi dari data tersebut tidak mudah untuk diketahui.

Implementasi algoritma kriptografi telah banyak diterapkan dalam penelitian untuk pengamanan file, diantaranya penelitian yang dilakukan Meylisa Siska Bangun, tahun 2019, dengan judul Implementasi Algoritma Route Cipher Dalam Pengamanan File Pdf, dengan kesimpulan algoritma route cipher melakukan enkripsi terhadap file dengan menggunakan kunci yang dapat membentuk matriks [1]. Penelitian yang dilakukan Yemima Kristiana Br. Simbolon, tahun 2019, dengan judul Perancangan Aplikasi Pengamanan File PDF Menggunakan Algoritma Playfair Cipher, dengan kesimpulan metode kriptografi dengan algoritma playfair cipher dapat dijadikan sebagai media mengamankan file PDF agar isi file tersebut aman terhadap gangguan dari berbagai tindakan yang tidak di inginkan [2]. Penelitian yang dilakukan Mukti Qamal, tahun 2014, dengan judul Kriptografi File Citra Menggunakan Algoritma TEA (Tiny Encryption Algorithm), dengan kesimpulan algoritma TEA dengan 32 round sangat cocok untuk sistem keamanan file citra yang mengandalkan kecepatan proses yang optimal [3].

Dalam penelitian ini algoritma yang digunakan untuk pengamanan file adalah algoritma TEA (Tiny Encryption Algorithm). Tiny Encryption Algorithm merupakan algoritma kriptografi block cipher yang memiliki tingkat keamanan yang tinggi dibandingkan dengan algoritma kriptografi simetri lainnya dalam pengamanan data. Kelebihan algoritma TEA adalah meminimkan penggunaan memori dan memaksimalkan kecepatan prosesnya.

2. METODOLOGI PENELITIAN

2.1 Kriptografi

Kriptografi merupakan ilmu dan seni untuk keamanan pesan yang dikirim dari suatu tempat ke tempat lain. Dalam perkembangannya kriptografi juga digunakan untuk mengidentifikasi pengiriman pesan tanda tangan digital dan keaslian dengan sidik jari digital [4].

Komponen kriptografi terdiri dari [5]:

1. Enkripsi, merupakan cara mengubah pesan asli *(plaintext)* menjadi kode-kode yang tidak dimengerti *(ciphertext)*
2. Dekripsi, merupakan cara mengubah pesan berkode *(ciphertext)* menjadi pesan asli *(plaintext)*
3. *Key*, merupakan kunci yang dipakai untuk melakukan enkripsi dan dekripsi. *Key* terbagi 2 yaitu *private key* (kunci rahasia) dan *public key* (kunci umum)

Algoritma kriptografi dibagi menjadi tiga bagian berdasarkan kunci yang dipakainya [3]:

1. Algoritma Simetri (menggunakan satu kunci untuk enkripsi dan dekripsinya)
2. Algoritma Asimetri (menggunakan kunci yang berbeda untuk enkripsi dan dekripsi)
3. *Hash Function*

2.2 Algoritma TEA *(Tiny Encryption Algorithm)*

Algoritma TEA *(Tiny Encryption Algorithm)* adalah algoritma sandi yang diciptakan oleh David Wheeler dan Roger Needham dari Computer Laboratory, Cambridge University, England pada bulan November tahun 1994. Algoritma ini merupakan algoritma penyandian *block cipher* yang dirancang untuk penggunaan memori yang seminimal mungkin dengan kecepatan proses yang maksimal [6]. Berikut ini adalah langkah-langkah algoritma TEA *(Tiny Encryption Algorithm)* :

2.2.1 Enkripsi

Pada proses enkripsi *plainteks* dibagi menjadi dua blok di mana tiap-tiap blok terdiri dari 32-bit data. Misalkan kedua blok adalah blok Z dan Y. Lalu *key* dengan panjang 128 bit dibagi ke dalam 4 blok, yaitu blok K_0, K_1, K_2, dan K_3 di mana tiap-tiap blok terdiri dari 32 bit data. Algoritma TEA menggunakan bilangan konstanta, yaitu 9E3779B9 dalam bilangan

ISBN: 978-602-52720-7-3
Februari 2020
Hal 490-494

heksadesimal, konstanta ini sering juga disebut sebagai *golden number* dan bilangan ini disimbolkan dengan delta dan berikut ini proses enkripsi dengan algoritma TEA *(Tiny Encryption Algorithm)* [7] :

a. *First round*
   1. *Shift-left process*, blok Z digeser 4 bit ke kiri. Hasilnya disimpan dengan nama Z_1
   2. Penjumlahan bagian pertama. Setelah digeser ke kiri, hasilnya dijumlahkan dengan K_0, hasilnya disimpan dengan nama Z_2. Nilai Z awal dijumlahkan dengan delta, lalu hasilnya disimpan dengan nama Z_3.
   3. *Shift-right process*, nilai Z awal digeser ke kanan 5 bit dan hasilnya disimpan dengan nama Z_5.
   4. Proses peng-XOR-an. Proses Peng-XOR-an diawali dengan meng-XOR-kan nilai Z_2 dan Z_3. Lalu hasilnya di XOR-kan dengan Z_5. Hasilnya disimpan dengan nama Z_6.
   5. Penjumlahan bagian kedua adalah menambahkan nilai Z_6 dengan nilai awal Z, tujuannya adalah untuk menjadi penghubung antara proses *first round* dan *second round*, sehingga ketika *first round* selesai, dapat dilanjutkan ke proses *second round*. Hasil penjumlahan disimpan dengan nama x_1.

b. *Second round*
   1. *Shift-left process*, nilai x_1 digeser sebanyak 4 bit ke kiri. Hasilnya disimpan dengan nama Y_1.
   2. Penjumlahan bagian pertama, nilai Y_1 ditambahkan dengan K_2 dan hasilnya disimpan dengan nama Y_2, lalu nilai x_1 ditambah dengan delta, lalu disimpan dengan nama Y_3.
   3. *Shift-right process*, nilai x_1 digeser ke kanan sebanyak 5 bit, lalu hasilnya disimpan dengan nama Y_4.
   4. Penjumlahan bagian kedua, nilai Y_4 ditambah dengan K_3, lalu hasilnya disimpan dengan nama Y_5
   5. Proses peng-XOR-an. Proses Peng-XOR-an diawali dengan meng-XOR-kan nilai Y_2 dan Y_3. Lalu, hasilnya di XOR-kan dengan Y_5. Hasilnya disimpan dengan nama Y_6. Dan hasil Y_6 harus ditambahkan dengan nilai awal Z, lalu hasilnya disimpan dengan nama x_2. Apabila proses *first round* dan proses *second round* telah selesai dilakukan, maka proses *one cycle* telah berhasil dilakukan, sehingga proses *first round* dan *second round* dapat dilakukan di blok berikutnya. Untuk mendapatkan ciphertext yang sesuai dengan algoritma TEA, maka harus dilakukan proses *one cycle* sebanyak 32 kali atau 64 *round*. *Ciphertext* dari enkripsi dengan algoritma TEA berada pada blok Z_{32} dan pada blok Y_{32}.

2.2.2 Dekripsi

Proses dekripsi diawali pada blok Z_{32} dan blok Y_{32}, *plaintext* yang diinginkan tersedia pada blok Z_0 dan blok Y_0 dan berikut ini adalah tahap-tahap yang digunakan untuk dekripsi dengan algoritma TEA *(Tiny Encryption Algorithm)* [7] :

a. *First round*
   1. *Shift-left process*, blok final_z digeser sebanyak 4 bit ke kiri. Hasilnya disimpan dengan nama final_z1.
   2. Penjumlahan bagian pertama. Setelah digeser ke kiri maka final_z1 dijumlahkan dengan K_2 dan hasilnya disimpan dengan nama final_z2. Sementara nilai final_z yang belum digeser dijumlahkan dengan delta, lalu hasilnya disimpan dengan nama final_z3.
   3. *Shift-right process*, nilai final_z digeser ke kanan 5 bit, lalu hasilnya disimpan dengan nama final_z4.
   4. Penjumlahan bagian kedua, final_z4 ditambahkan dengan K_3, lalu hasilnya disimpan dengan nama final_z5.
   5. Proses peng-XOR-an. Nilai final_z2 di-XOR-kan dengan final_z3 lalu hasilnya di XOR-kan lagi dengan final_z5. Hasil akhirnya disimpan dengan nama final_z6.
   6. Proses pengurangan. Pada proses ini, nilai final_z6 dikurangkan dengan final_y agar proses *first round* dari algoritma TEA selesai. Hasil pengurangannya disimpan dengan nama final_x1.

b. *Second round*
   1. *Shift-left process*, nilai final_x1 digeser 4 bit ke kiri. Hasilnya disimpan dengan nama final_y1.
   2. Penjumlahan bagian pertama, Pada proses ini, nilai final_y1 ditambahkan dengan K_0 dan hasilnya disimpan dengan nama final_y2. Lalu nilai final_x1 yang belum digeser ditambahkan dengan delta, hasilnya disimpan dengan nama final_y3.
   3. *Shift-right process*, nilai final_x1 digeser ke kanan 5 bit lalu hasilnya disimpan dengan nama final_y4.
   4. Penjumlahan bagian kedua, nilai final_y4 ditambah dengan K_1 dan hasilnya disimpan dengan nama final_y5.
   5. Proses peng-XOR-an. Proses Peng-XOR-an diawali dengan meng-XOR-kan nilai final_y2 dan final_y3. Lalu hasilnya di XOR-kan dengan final_y5. Hasilnya disimpan dengan nama final_y6.
   6. Proses pengurangan. Pada proses ini, nilai final_y6 dikurangkan dengan final_z agar proses *second round* dari algoritma TEA selesai. Hasil pengurangannya disimpan dengan nama final_x2. Proses *first round* dilakukan sebanyak 32 kali dan proses *second round* dilakukan sebanyak 32 kali juga untuk memenuhi persyaratan dari dekripsi dengan menggunakan algoritma TEA. langkah terakhir adalah *plaintext* yang telah di dekripsi dibandingkan dengan *plaintext* awal untuk mengecek apakah pesan tersebut sama dengan *plaintext* awal atau tidak.

2.3 PDF *(Portable Document Format)*

PDF (*Portable Document Format*) sebuah *file Adobe Acrobat* yang bisa berupa vektor dan raster, tergantung dari karya asli. Jika karya seni asli adalah *file* vektor, PDF akan menjadi seni vektor, Jika karya seni asli adalah *file* raster, PDF akan menjadi *file* raster.

File pdf sering merupakan kombinasi keduanya. *File* PDF dikompres seperti *file* JPG, kompresi dapat dibuat berbagai tingkat kompresi, yang berdampak pada kualitas reproduksi akhir. Untuk tujuan cetak, *file* PDF harus disimpan dengan resolusi tertinggi, yang mungkin terjadi jika *file* PDF dibuat menggunakan *Adobe Acrobat Distiller* atau program pembuatan PDF [2].

ISBN: 978-602-52720-7-3
Februari 2020
Hal 490-494

3. ANALISA DAN PEMBAHASAN

Proses pengamanan *file* PDF dengan menggunakan algoritma TEA *(Tiny Encryption Algorithm)* menggunakan sampel *plaintext* yang ditandai dengan kotak berwarna merah.

Implementasi Algoritma Tiny Encryption Algorithm Untuk Pengamanan File PDF

Aulia Husna[1], **Juli Mewarni Halu**[2], **Yuliani Susiawati Hondro**[3]
Program Studi Teknik Informatika, STMIK Budi Darma, Medan, Indonesia
[1]aulia.hsbri@gmail.com, [2] juliimewarnihalu@gmail.com, [3] yulianisusiawatihondro@gmail.com

Abstrak

Kriptografi merupakan teknik untuk pengamanan data dengan cara menyembunyikan pesan agar maknanya tidak mudah dimengerti. Pengaman *file* akan mengurasi risiko terhadap penyadapan yang dilakukan oleh pihak yang tidak berkepentingan pada proses pengiriman. Dalam pengamanan data, *file* PDF *(Portable Document Format)* banyak gunakan dalam bertukar informasi, sehingga perlu adanya pengamanan terhadap dalamnya. Implementasi kriptografi untuk pengamanan *file* banyak digunakan dalam penelitian dan salah satu contohnya Algoritma TEA *(Tiny Encryption Algorithm)*. Implementasi algoritma TEA memiliki keamanan yang sangat tinggi dalam pengamanan *file*

Kata Kunci : Kriptografi, Pengamanan, Implementasi, Algoritma TEA, File PDF

1. PENDAHULUAN

Gambar 1. Teks Yang Akan Dienkripsi

Plaintext : File(spasi)PDF
Key : algoritmaTEA1994
Delta : 9E3779B9 (dalam bilangan hexadecimal)

Plaintext dibagi menjadi 2 blok, di mana blok Y = File dan Z = (spasi)PDF. Untuk *key* 128 bit dibagi menjadi 4 blok yang masing-masing mendapat 32 bit di mana K_0 = algo, K_1 = ritm, K_2 = aTEA dan K_3 = 1994. Selanjutnya *plaintext* diubah menjadi ASCII, kemudian diubah ke biner dan hasilnya seperti berikut :

$Y_{(File)}$ = 01000110011010010110110001100101
$Z_{((spasi)PDF)}$ = 00100000010100000100010001000110
Delta = 10011110001101110111100110111001

Sedangkan hasil untuk *key* yang diubah menjadi ASCII dan kemudian diubah ke binar sebagai berikut :

K_0 (algo) = 01100001011011000110011101101111
K_1 (ritm) = 01110010011010010111010001101101
K_2 (aTEA) = 01100001010101000100010101000001
K_3 (1994) = 00110001001110010011100100110100

3.1 Enkripsi

a. *First round*

1. *Shift-left process*
   $Z_{((spasi)PDF)}$ = 00100000010100000100010001000110
   Z_1 = 00000101000001000100010001100010
2. Penjumlahan bagian pertama
   Z_1 = 00000101000001000100010001100010
   K_0 (algo) = 01100001011011000110011101101111 +
   Z_2 = 01100110011100001010101111010001

   $Z_{((spasi)PDF)}$ = 00100000010100000100010001000110
   Delta = 10011110001101110111100110111001 +
   Z_3 = 10111110100001111011110111111111
3. *Shift-right process*
   $Z_{((spasi)PDF)}$ = 00100000010100000100010001000110
   Z_5 = 00110001000000101000001000100010
4. Proses peng-XOR-an
   Z_2 = 01100110011100001010101111010001
   Z_3 = 10111110100001111011110111111111
   = 11011000111101110001011000101110
   Z_5 = 00110001000000101000001000100010
   Z_6 = 11101001111101011001010000001100
5. Penjumlahan bagian kedua

ISBN: 978-602-52720-7-3
Februari 2020
Hal 490-494

Z_6 = 11101001111101011001010000001100
$Z_{((spasi)PDF)}$ = 00100000010100000100010001000110 +
x_1 = 10000101001000101110110000101001 0

b. *Second round*

1. *Shift-left process*
   x_1 = 10000101001000101110110000101001 0
   Y_1 = 01010010001011101100001010010 1000
2. Penjumlahan bagian pertama
   Y_1 = 01010010001011101100001010010 1000
   K_2 (aTEA) = 01100001010101000100010101000001 +
   Y_2 = 10000010110110001110010100110100 1

   x_1 = 10000101001000101110110000101001 0
   Delta = 10011110001101110111100110111001 +
   Y_3 = 11010100001111101010100100000101 1
3. *Shift-right process*
   x_1 = 10000101001000101110110000101001 0
   Y_4 = 10010100001010010001011101100001 0
4. Penjumlahan bagian kedua
   Y_4 = 10010100001010010001011101100001 0
   K_3 (1994) = 00110001001110010011100100110100 +
   Y_5 = 10101100110001011011001111111 10110
5. Proses peng-XOR-an
   Y_2 = 10000010110110001110010100110100 1
   Y_3 = 11010100001111101010100100000101 1
   = 01010110111001100100110000110001 0
   Y_5 = 10101100110001011011001111111 10110
   Y_6 = 11111010001000111111111111001 0100
   $Z_{((spasi)PDF)}$ = 00100000010100000100010001000110 +
   x_2 = 11101010000010111101110111101 0010

Proses *first round* dan *second round* sudah selesai dilakukan, maka proses *one cycle* telah berhasil dilakukan, maka proses *first round* dan *second round* dapat dilakukan di blok berikutnya. Untuk mendapatkan *ciphertext* dengan algoritma TEA, maka dilakukan proses *one cycle* sebanyak 32 kali atau 64 *round*. *Ciphertext* dari *plaintext* "File(spasi)PDF" dengan *key* "algoritma TEA1994" adalah **_46?k&??**

3.2 Dekripsi

Ciphertext yang di dapat dari enkripsi yaitu **_46?k&??** akan dibagi menjadi 2 blok, di mana blok Y_{32} = _46? dan Z_{32} = k&?? dan selanjutnya *ciphertext* diubah menjadi ASCII, kemudian diubah ke biner dan hasilnya seperti berikut :

Y_{32} (_46?) = 01011111001101000011011010011111
Z_{32} (k&??) = 01101011001001101110000111010100
Delta = 10011110001101110111100110111001

a. *First round*

1. *Shift-left process*
   Z_{32} (k&??) = 01101011001001101110000111010100
   final_z1 = 10110010011011100001110101000110
2. Penjumlahan bagian pertama
   final_z1 = 10110010011011100001110101000110
   K_2 (aTEA) = 01100001010101000100010101000001 +
   final_z2 = 10001001111000010011000101000011 1

   Z_{32} (k&??) = 01101011001001101110000111010100
   Delta = 10011110001101110111100110111001 +
   final_z3 = 10000100101011110010110111000110 1
3. *Shift-right process*
   Z_{32} (k&??) = 01101011001001101110000111010100
   final_z4 = 10100011010110010011011100001110
4. Penjumlahan bagian kedua
   final_z4 = 10100011010110010011011100001110
   K_3 (1994) = 00110001001110010011100100110100 +
   final_z5 = 11010100100100100111000001000010
5. Proses peng-XOR-an
   final_z2 = 10001001111000010011000101000011 1

ISBN: 978-602-52720-7-3
Februari 2020
Hal 490-494

final_z3 = 10000100101011110010110111000110 1
= 00001101010011100001110010000101 0
final_z5 = 10110011011110010011111100111110
final_z6 = 01010100111100101000001100011010 0

6. Proses pengurangan
final_z6 = 01010100111100101000001100011010 0
Y_{32} (_46?) = 01011111001101000011011010011111 -
final_x1 = 00100101010110000110011111001010 1

b. *Second round*

1. *Shift-left process*
final_x1 = 00100101010110000110011111001010 1
final_y1 = 01010101100001100111110010101001 0
2. Penjumlahan bagian pertama
final_y1 = 01010101100001100111110010101001 0
K_0 (algo) = 01100001011011000110011011011111 +
final_y2 = 10000110001111001011000001100000 1

final_x1 = 00100101010110000110011111001010 1
Delta = 10011110001101110111100110111001 +
final_y3 = 11101000111010000100100101001110

3. *Shift-right process*
final_x1 = 00100101010110000110011111001010 1
final_y4 = 10101001001010101100001100111110 0
4. Penjumlahan bagian kedua
final_y4 = 10101001001010101100001100111110 0
K_1 (ritm) = 01110010011010010111010001101101 +
final_y5 = 11100010010111110111110101110100 1
5. Proses peng-XOR-an
final_y2 = 10000110001111001011000001100000 1
final_y3 = 11101000111010000100100101001110
= 11110010010010001001010011000111 1
final_y5 = 11100010010111110111110101110100 1
final_y6 = 00010000000101111110100110110011 0
6. Proses pengurangan
final_y6 = 00100000001011111101001101100110
Z_{32} (k&??) = 01101011001001101110000111010100 –
final_x2 = 11111111111111111111111111111111 1101101010000100011110001100100 10

Pada proses ini *second round* telah selesai dilakukan. Proses *first round* dan *second round* dilakukan sebanyak 32 kali untuk dekripsi dengan menggunakan algoritma TEA. Langkah terakhir adalah membandingkan *plaintext* dekripsi dengan *plaintext* awal untuk mengecek apakah pesan tersebut sama atau tidak.

4. KESIMPULAN

Kesimpulan dari implementasi algoritma Tiny Encryption Algorithm untuk pengaman file PDF adalah :

a. Algoritma TEA dapat diimplementasikan dalam pengamanan file PDF
b. Prosedur pengamanan file PDF dengan algoritma TEA adalah plaintext dibagi menjadi 2 blok yang masing-masing memiliki bit sebanyak 32 bit dan key 128 bit dibagi menjadi 4 blok di mana tiap blok memiliki 32 bit.
c. Proses enkripsi dan dekripsi dilakukan one cycle sebanyak 32 kali dimana one cyclei memiliki 2 round. Dengan kata lain proses enkripsi dan dekripsi dilakukan sebanyak 64 round.

REFERENCES

[1] M. S. Bangun, “Implementasi Algoritma Route Cipher Dalam Pengamanan File Pdf,” vol. 1, no. 1, pp. 1–6, 2019.
[2] Yemima Kristiana Br. Simbolon, “PERANCANGAN APLIKASI PENGAMANAN FILE PDF MENGGUNAKAN ALGORITMA PLAYFAIR CIPHER,” *Maj. Ilm. INTI*, vol. 14, pp. 187–190, 2019.
[3] M. Qamal, “KRIPTOGRAFI FILE CITRA MENGGUNAKAN ALGORITMA TEA (TINY ENCRYPTION ALGORITHM).”
[4] H. Mukhtar, *Kriptografi Untuk Data Keamanan*. Yogyakarta: Deepublish, 2018.
[5] D. Ariyus, *Pengantar Ilmu Kriptografi (Teori Analisis dan Implementasi*. Yogyakarta: Andi Offset, 2008.
[6] U. A. Siswantoa, M.Anifb, “Pengamanan Data User Login dengan Algoritma Kriptografi Tea dan Notifikasi SMS,” *Pros. Semin. Nas. SISFOTEK*, no. September, pp. 4–5, 2018.
[7] S. ; N. Hunn, Stephanie Ang Yee; Zarina, “The development of tiny encryption algorithm (TEA) crypto-core for mobile systems,” *IEEE Int. Conf. Electron. Des. Syst. Appl.*, 2012.