Analisis Algoritma *Blowfish* Pada Proses Enkripsi Dan Dekripsi File

Hendri Maradona[1], Basorudin[2]
Sistem Informasi, Fakultas Ilmu Komputer, Universitas Pasir Pengaraian
Jl. Tuanku Tambusai, Kumu Kec. Rambah Hilir Kabupaten Rokan Hulu
hendrimaradonapakpahan@gmail.com, basorudin09@gmail.com

Abstrak

Jurnal ini membahas bagaimana jalannya algoritma Blowfish khususnya dalam hal proses enkripsi dan dekripsi (kriptografi) file. Algoritma Blowfish merupakan salah satu alternatif untuk algoritma enkripsi. Algoritma Blowfish termasuk kedalam cipher blok yang masih dianggap aman karena belum ditemukan attack yang benar-benar dapat mematahkannya. Disamping analisis kinerja algoritma blowfish, pembahasan ini juga mencoba menguji dan mengimplementasi algoritma Blowfish pada sebuah aplikasi yang akan melakukan enkripsi dan dekripsi pada teks menggunakan bahasa pemrograman microsoft visual basic 6.0. Aplikasi tersebut diharapkan dapat menerapkan algoritma Blowfish secara optimal, oleh karena itu akan dibahas mengenai karakteristik algoritma Blowfish yang dapat membuat algoritma ini berjalan secara optimal. Dengan adanya implementasi itu pula, diharapkan dapat menjadi contoh penggunaan algoritma Blowfish untuk proses enkripsi yang sesungguhnya. Aplikasi yang akan dibuat merupakan aplikasi sederhana yang hanya dapat melakukan enkripsi pada file teks, namun hal tersebut dianggap cukup untuk memberi contoh penggunaan algoritma Blowfish dalam proses enkripsi.

Kata Kunci : Kriptografi, Algoritma *Blowfish*, Enkripsi dan Dekripsi

1. PENDAHULUAN

1.1 Latar Belakang

Kriptografi merupakan salah satu teknik dalam duni komputer dan telah menjadi bagian penting dalam dunia teknologi informasi saat ini. Sebagai contoh hampir semua penerapan teknologi informasi saat ini menggunakan kriptografi sebagai alat untuk menjamin keamanan dan kerahasiaan informasi baik bagi keperluan individu maupun bagi corporate atau perusahaan. Karena itu pulalah, kriptografi menjadi ilmu yang berkembang pesat. Dalam waktu singkat, amat banyak bermunculan algoritma-algoritma baru yang dianggap lebih unggul daripada pendahulunya. Namun, tetap saja cipher yang digunakan tidak lepas dari penemuan lama. Algoritma kunci simetri termasuk algoritma yang masih sering digunakan dalam pembuatan algoritma kriptografi. Pada saat ini, algoritma enkripsi kunci simetri yang banyak digunakan adalah algoritma blok, yang beroperasi pada suatu potongan pesan (blok) yang berukuran sama (biasanya 64 bit) pada suatu saat.

Selain algoritma blok, ada juga algoritma aliran yang beroperasi pada potongan data yang bervariasi. Dibandingkan dengan cipher aliran, baik dalam desain dan penerapan, cipher blok dianggap lebih rumit.

Pada saat Blowfish dirancang, diharapkan mempunyai kriteria perancangan sebagai berikut :

1. Cepat, Blowfish melakukan enkripsi data pada microprocessors 32-bit. dengan rate 26 clock cycles per byte.

2. Compact, Blowfish dapat dijalankan pada memori kurang dari 5K.
3. Sederhana, Blowfish hanya menggunakan operasi-operasi Sederhana, penambahan, XOR, dan lookup tabel pada operan 32-bit.
4. Memiliki tingkat keamanan yang bervariasi, panjang kunci yang digunakan oleh Blowfish dapat bervariasi dan bisa sampai sepanjang 448 bit.

Materi ini bertujuan dapat mempelajari algoritma Blowfish secara keseluruhan agar dapat memahami cara kerja dan struktur algoritmanya, kemudian menerapkan strategi perancangan Blowfish sehingga algoritma ini dapat berjalan secara optimal. Atau paling tidak dapat melakukan enkripsi dengan baik pada aplikasi yang akan dibuat.

Aplikasi yang akan dibuat diharapkan dapat menggambarkan penggunaan algoritma Blowfish pada proses enkripsi teks sehingga lebih mudah memahami cara kerja dan struktur algoritma Blowfish.

1.2 RUMUSAN MASALAH

Dari uraian pada latar belakang tersebut, maka dapat dirumuskan masalah, yaitu :

1. Bagaimana membangun aplikasi kriptografi dengan menggunakan algoritma *blowfish*?
2. Bagaimana langkah-langkah dalam proses enkripsi dan dekripsi pada algoritma *blowfish*?
3. Bagaimana mengananalisis kinerja dari algoritma *blowfish* dalam proses enkripsi dan dekripsi ?

1.3 BATASAN MASALAH

Adapun batasan masalah dalam artikel ini, yaitu :

1. Membahas hanya mengenai algoritma *blowfish* pada proses enkripsi dan dekripsi.
2. Pembangunan aplikasi digunakan untuk menganalisis kinerja dalam hal estimasi waktu proses enkripsi dan dekripsi.
3. File yang akan diuji untuk keperluan analisis perbandingan adalah file dokumen *extention *.doc, *.xls, *.psd, *.ppt*.
4. Sistem ini dibangun menggunakan Visual Basic 6.0 dan berbasis *windows.*

1.4 TUJUAN

Adapun tujuan dari artikel ini, yaitu untuk :

1. Mengetahui kecepatan proses enkripsi dan dekripsi suatu file pada algoritma *blowfish*.
2. Mengetahui langkah-langkah dalam proses enkripsi dan dekripsi pada algoritma *blowfish*
3. Mengetahui hasil kinerja dari algoritma *blowfish*, sehingga dapat dijadikan rekomendasi untuk proses enkripsi dan dekripsi.
4. Menganalisa proses enkripsi dan dekripsi pada algoritma *blowfish*.
5. Merancang program yang dapat digunakan untuk keperluan analisis perbandingan kinerja dari algoritma *blowfish*.
6. Membangun program kriptografi yang mengadopsi algoritma *blowfish*.
7. Menguji algoritma *blowfish* dalam proses enkripsi dan dekripsi dengan file uji yang telah ditetapkan menggunakan program kriptografi

2. TINJAUAN PUSTAKA

2.1 Algoritma *Kunci Simetris*

Algoritma kunci simetrik adalah algoritma kriptografi yang memiliki kunci yang sama untuk proses enkripsi dan dekripsinya. Kunci tersebut merupakan satu-satunya jalan untuk proses dekripsi (kecuali mencoba membobol algoritma tersebut), sehingga kerahasiaan kunci menjadi nomor satu. Untuk mengirimkan kunci tersebut ke suatu pihak tanpa diketahui pihak yang lain merupakan masalah awal dari algoritma kunci simetrik. Algoritma kunci simetrik terbagi

menjadi dua buah bergantung pada datanya. Keduanya adalah: cipher aliran (stream cipher) dan cipher blok (block cipher). Cipher aliran memproses satu bit pesan sekali dalam satu waktu, sedangkan cipher blok memproses sekumpulan bit sekaligus sebagai satu unit. Ukuran blok yang umu dipakai adalah 64 bit. Algoritma kunci simetrik terbagi menjadi dua buah bergantung pada datanya. Keduanya adalah: cipher aliran (stream cipher) dan cipher blok (block cipher)

2.2 Algoritma *Blowfish*

Blowfish atau yang disebut juga "OpenPGP.Cipher.4″ adalah algoritma kunci simetrik cipher blok yang dirancang pada tahun 1993 oleh Bruce Schneider untuk menggantikan DES (Data Encryption Standard). Algoritma Blowfish dibuat untuk digunakan pada komputer yang mempunyai microposesor besar (32-bit keatas dengan cache data yang besar).

Pada saat itu banyak sekali rancangan algoritma yang ditawarkan, namun hampir semua terhalang oleh paten atau kerahasiaan pemerintah Amerika. Schneier menyatakan bahwa blowfish bebas paten dan akan berada pada domain publik. Dengan pernyataan Schneier tersebut blowfish telah mendapatkan tempat di dunia kriptografi, khususnya bagi masyarakat yang membutuhkan algoritma kriptografi yang cepat, kuat, dan tidak terhalang oleh lisensi.

Blowfish dirancang dan diharapkan mempunyai kriteria perancangan yang diiginkan sebagai berikut :

1. Cepat, Blowfish melakukan enkripsi data pada microprocessor 32-bit dengan rate 26 clock cycles per byte.
2. Compact,Blowfish dapat dijalankan pada memory kurang dari 5K.
3. Sederhana, Blowfish hanya menggunakan operasi – operasi operasi sederhana, seperti : penambahan, XOR, dan lookup tabel pada operan32-bit.
4. Memiliki tingkat keamanan yang bervariasi, panjang kunci yang digunakan oleh Blowfish dapat bervariasi dan bisa sampai sepanjang minimal 32-bit, maksimal 448 -bit, Multiple 8 bit, default 128 bit

3. METODOLOGI PENELITIAN

3.1. Pendahuluan

Pada Bab ini akan diuraikan kerangka kerja dari penelitian yang akan dilakukan berkaitan dengan analisa perbandingan kinerja algoritma *blowfish*. Kerangka kerja ini merupakan tahapan-tahapan yang akan dilakukan dalam menyelesaikan masalah yang akan dibahas agar penelitian dapat berjalan dengan baik, mulai dari identifikasi masalah, studi pustaka, analisis sistem, perancangan, implementasi, pengujian, hingga dokumentasi.

3.2. Kerangka Kerja

Untuk memperoleh hasil kinerja yang efektif dari algoritma *blowfish*, maka penulis merancang kerangka kerja. Adapun tahapan nya sebagai berikut:

1. Identifikasi Masalah
   Identifikasi masalah merupakan tahap awal dari penelitian ini.
2. Studi Pustaka
   Studi pustaka dilakukan untuk melengkapi pengetahuan dasar yang dimiliki peneliti, sehingga peneliti dapat menyelesaikan penelitian ini.
3. Analisis Sistem
   Pada tahapan ini dilakukan analisis terhadap kebutuhan sistem, serta menganalisis elemen-elemen yang dibutuhkan oleh sistem. Pada tahap ini dilakukan studi terhadap sistem kerja

algoritma *blowfish* yang didapatkan. Studi ini dilakukan untuk memperoleh gambaran dari sistem algoritma *blowfish* sehingga dapat merancang program enkripsi dan dekripsi file algoritma *blowfish.*

4. Perancangan
   Pada tahapan ini dilakukan perancangan tampilan program yang akan dibuat penulis. Tampilan program bersifat *user-friendly* atau mudah digunakan oleh pengguna.
5. Implementasi
   Tahapan ini dilakukan untuk mengimplementasikan hasil rancangan dan analisis di atas. Pada tahapan ini dilakukan pembuatan program, pembuatan antarmuka masukan dan keluaran, dan antarmuka proses enkripsi dan dekripsi algoritma *blowfish.* Dalam proses pembuatan program enkripsi dan dekripsi algoritma *blowfish*, penulis menggunakan spesifikasi sebagai berikut :
   1) Perangkat Keras
      a.Processor Intel Dual Core
      b. Memory 2 GB
      c.Hardisk 120 GB
      d. Mouse, Keyboard, Monitor, dan lain-lain
   2) Perangkat Lunak
      a.Sistem Operasi Microsoft Windows XP Professional SP3.
      b. Microsoft Visual Basic 6.0.
6. Pengujian
   Pada tahap ini penulis melakukan pengujian terhadap algoritma *blowfish* dengan menggunakan aplikasi kriptografi. Adapun mekanisme pengujiannya yaitu :
   a. Mempersiapkan file-file yang akan diuji, yaitu file dokumen dengan *extention *.doc, *.xls, *.psd, *.ppt*.
   b. Menetapkan kunci untuk proses enkripsi dan dekripsi. Karena algoritma *blowfish* merupakan kunci simetris, maka kunci yang digunakan adalah kunci yang sama dalam proses enkripsi dan dekripsi
   c. Melakukan proses enkripsi setiap file asli yang akan diuji dengan kunci yang telah ditetapkan. Kemudian mencatat waktu proses enkripsi, dan kapasitas file sebelum dan sesudah proses enkripsi dilakukan.
   d. Melakukan proses dekripsi setiap file hasil enkripsi dengan kunci yang telah ditetapkan (kunci yang sama dengan kunci yang digunakan dalam proses enkripsi). Kemudian mencatat waktu proses dekripsi, dan kapasitas file sebelum dan sesudah proses dekripsi dilakukan, menghitung kecepatan proses enkripsi dan dekripsi.
7. Dokumentasi

Tahapan ini penulis akan melaporkan hasil penelitian yang sudah dilakukan. Dokumen berisi laporan mulai dari identifikasi masalah hingga implementasi dan pengujian.

4. PEMBAHASAN

4.1. Analisa Sistem

Pada tahap ini, penulis menganalisa proses enkripsi dan dekripsi yang diterapkan pada algoritma *blowfish* menggunakan *flowchart*, sehingga didapati gambaran bagaimana proses enkripsi dan dekripsi pada kedua algoritma tersebut.

4.1.1. Algoritma *Blowfish*

Untuk lebih memahami proses enkripsi pada algoritma blowfish, maka penulis membuat contoh, perhitungan manual yang terjadi pada proses enkripsi seperti gambar 4.2. Dalam hal ini, penulis menggunakan parameter sebagai berikut :

Plaintext = UPI YPTK

Password = 2905

Langkah perhitungan manual yang penulis lakukan, yaitu sebagai berikut :

1. Inisialisasi *P-Array* (P_0, P_1, ..., P_{17}) masing-masing 32 bit, seperti tabel 4.1.

Tabel 4.1. P-Array Konversi ke Biner

P-array	Hexa	Konversi Biner (32 bit)
P0	243F6A88	00100100 00111111 01101010 10001000
P1	85A308D3	10000101 10100011 00001000 11010011
P2	13198A2E	00010011 00011001 10001010 00101110
P3	3707344	00000011 01110000 01110011 01000100
P4	A4093822	10100100 00001001 00111000 00100010
P5	299F31D0	00101001 10011111 00110001 11010000
P6	82EFA98	00001000 00101110 11111010 10011000
P7	EC4E6C89	11101100 01001110 01101100 10001001
P8	452821E6	01000101 00101000 00100001 11100110
P9	38D01377	00111000 11010000 00010011 01110111
P10	BE5466CF	10111110 01010100 01100110 11001111
P11	34E90C6C	00110100 11101001 00001100 01101100
P12	C0AC29B	11000000 10101100

	7	00101001 10110111
P13	C97C50DD	11001001 01111100 01010000 11011101
P14	3F84D5B5	00111111 10000100 11010101 10110101
P15	B5470917	10110101 01000111 00001001 00010111
P16	9216D5D9	10010010 00010110 11010101 11011001
P17	8979FB1B	10001001 01111001 11111011 00011011

2. Inisialisasi S-Array yang berjumlah masing-masing 255 dalam bentuk hexadecimal yang kemudian dikonversi ke biner, seperti tabel 4.2.

Tabel 4.2. Konversi S-Array ke Biner

S-Array	Hexa	Konversi Biner
S1,0 ... S1,255	D1310BA6 6E85076A	11010001 00110001 00001011 10100110 01101110 10000101 00000111 01101010
S2,0 ... S2,255	4B7A70E9 DB83ADF7	01001011 01111010 01110000 11101001 11011011 10000011 10101101 11110111
S3,0 ... S3,255	E93D5A68 406000E0	11101001 00111101 01011010 01101000 01000000 01100000 00000000 11100000
S4,0 ... S4,255	3A39CE37 3AC372E6	00111010 00111001 11001110 00110111 00111010 11000011 01110010 11100110

3. Plaintext = UPI YPTK

Tabel 4.3. Konversi Plaintext Ke Biner

Karakter	ASCII (Hexa)	Biner
U	55	01010101
P	50	01010000
I	49	01001001
<space>	20	00100000
Y	59	01011001
P	50	01010000
T	54	01010100
K	4B	01001011

4. Kemudian Plaintext dibagi menjadi 2 bagian *XL* dan *XR* menjadi :
   XL = 01010101 01010000 01001001 00100000
   XR = 01011001 01010000 01010100 01001011
5. Pembangkitan Sub Kunci :
   Kunci : 2905
   Tabel 4.4. Konversi Kunci Ke Biner

Karakter	ASCII (Hexa)	Biner
2	32	00110010
9	39	00111001
0	30	00110000
5	35	00110101

Biner : 00110010 00111001 00110000 00110101

a. SubKunci Untuk Iterasi Pertama :
   P_0 = P_0 XOR Kunci
   P_0 = 00100100 00111111 01101010 10001000 XOR
   00110010 00111001 00110000 00110101
   P_0 = 00010110 00000110 01011010 10111111
b. SubKunci untuk iterasi kedua :
   P_1 = P_1 XOR P_0
   P_1 = 10000101 10100011 00001000 11010011 XOR
   00010110 00000110 01011010 10111111
   P_1 = 10010011 10100101 01010010 01101100

6. Dalam hal ini penulis, hanya melakukan satu iterasi, dikarenakan total iterasi proses enkripsi adalah 16 putaran.
   a. Untuk iterasi pertama $i = 0$ yaitu :
      XL = *XL* XOR P_0
      XL = 01010101 01010000 01001001 00100000 XOR
      00010110 00000110 01011010 10111111
      XL = 01000011 01010110 00010011 10011111
      Fungsi *F* didapat dari :

      XL dibagi menjadi 4 (*a*, *b*, *c*, *d*) masing-masing 8 bit =

      a = 01000011

      b = 01010110

      c = 00010011

      d = 10011111

      Fungsi *F* : $F(XL) = (((S_0.a + S_1.b \bmod 2^{32}) \text{ XOR } S_2, c) + S_3,d \bmod 2^{32})$

      $S_0.a + S_1.b \bmod 2^{32}$

      = (11010001 00110001 00001011 10100110 . 01000011) + (01001011 01111010 01110000 11101001. 01010110) mod 2^{32}

      =(11011010111111110101100000110001110010 +
      11001010110110010000111101110010000110) mod 2^{32}

=00011010 11110111 11111010 10111000

XOR S_2.c = 00011010 11110111 11111010 10111000 XOR (11101001 00111101 01011010 01101000. 11100100)

=00011010 11110111 11111010 10111000 XOR 11001111101110101010010010000010010100000

=11001111 10100000 01010011 01111110 00011000

+ S3.d mod 2^{32}

=(11001111 10100000 01010011 01111110 00011000 + (00111010 00111001 11001110 00110111. 10011111)) mod 2^{32}

=(11001111 10100000 01010011 01111110 00011000 + 10010000101001111001110001010000101001) mod 2^{32}

=11001010 00111010 10010010 01000001

F(XL) = 11001010 00111010 10010010 01000001

XR = *F(XL)* XOR *XR*

XR = 11001010 00111010 10010010 01000001 XOR

10010001 01111100 00111001 00111100

XR = 01011011 01000110 10101011 01111101

Menukar Nilai *XL* dan *XR* :

XL = *XR* ; *XR* = *XL*

XL = 01011011 01000110 10101011 01111101;

XR = 01000011 01010110 00010011 10011111

7. Setelah melakukan 16 iterasi, maka akan menghasilkan nilai baru *XL* dan *XR* masing-masing 32 bit. Tukar kembali *XL* dan *XR*. Setelah itu XOR-kan nilai *XL* dan *XR* : *XR* = *XR* XOR P_{16} dan *XL* = *XL* XOR P_{17}
8. Kemudian *XL* dan *XR* digabungkan sehingga menjadi 64 bit.
9. Nilai biner tersebut di konversikan ke dalam kode ASCII sehingga menghasilkan ciphertext yaitu : Ü-/*œ|9‹

Untuk lebih memahami proses dekripsi pada algoritma blowfish, maka penulis membuat contoh, perhitungan manual yang terjadi pada proses enkripsi seperti gambar 4.3. Dalam hal ini, penulis menggunakan parameter sebagai berikut :

Ciphertext = Ü-/*œ|9‹

Password = 2905

Langkah perhitungan manual yang penulis lakukan, yaitu sebagai berikut :

1. Inisialisasi *P-Array* (P_0, P_1, ..., P_{17}) masing-masing 32 bit, seperti tabel 4.5.

Tabel 4.5. Konversi P-Array Ke Biner

P-array	Hexa	Konversi Biner (32 bit)
P0	243F 6A88	00100100 00111111 01101010 10001000
P1	85A3 08D3	10000101 10100011 00001000 11010011
P2	1319 8A2 E	00010011 00011001 10001010 00101110
P3	3707 344	00000011 01110000 01110011 01000100
P4	A409 3822	10100100 00001001 00111000 00100010
P5	299F 31D0	00101001 10011111 00110001 11010000
P6	82EF A98	00001000 00101110 11111010 10011000
P7	EC4 E6C8 9	11101100 01001110 01101100 10001001
P8	4528 21E6	01000101 00101000 00100001 11100110
P9	38D0 1377	00111000 11010000 00010011 01110111
P10	BE54 66CF	10111110 01010100 01100110 11001111
P11	34E9 0C6 C	00110100 11101001 00001100 01101100
P12	C0A C29 B7	11000000 10101100 00101001 10110111
P13	C97 C50 DD	11001001 01111100 01010000 11011101

P14	3F84 D5B 5	00111111 11010101	10000100 10110101
P15	B547 0917	10110101 00001001	01000111 00010111
P16	9216 D5D 9	10010010 11010101	00010110 11011001
P17	8979 FB1 B	10001001 11111011	01111001 00011011

2. Inisialisasi S-Array yang berjumlah masing-masing 255 dalam bentuk hexadecimal yang kemudian dikonversi ke biner, seperti tabel 4.6.

Tabel 4.6. Konversi S-Array ke Biner

S-Array	Hexa	Konversi Biner
S1,0 ... S1,255	D1310BA 6 6E85076A	11010001 00110001 00001011 10100110 01101110 10000101 00000111 01101010
S2,0 ... S2,255	4B7A70E9 DB83ADF 7	01001011 01111010 01110000 11101001 11011011 10000011 10101101 11110111
S3,0 ... S3,255	E93D5A68 406000E0	11101001 00111101 01011010 01101000 01000000 01100000 00000000 11100000
S4,0 ... S4,255	3A39CE37 3AC372E6	00111010 00111001 11001110 00110111 00111010 11000011 01110010 11100110

3. Ciphertext = Ü-/*œ|9‹

Tabel 4.7. Konversi Ciphertext ke Biner

Karakter	ASCII (Decimal)	Biner
Ü	154	10011010
-	45	00101101
/	47	00101111
*	42	00101010
œ	145	10010001
|	124	01111100
9	57	00111001
‹	60	00111100

4. Kemudian dibagi menjadi 2 bagian *XL* dan *XR* menjadi :
   XL = 10011010 00101101 00101111 00101010
   XR = 10010001 01111100 00111001 00111100
5. Pembangkitan Sub Kunci :
   Kunci : 2905
   Tabel 4.8. Konversi Kunci Ke Biner

Karakter	ASCII (Hexa)	Biner
2	32	00110010
9	39	00111001
0	30	00110000
5	35	00110101

   Biner : 00110010 00111001 00110000 00110101
   a. SubKunci Untuk Iterasi Pertama :
      P_{17} = P_{17} XOR Kunci
      P_{17} = 10001001 01111001 11111011 00011011 XOR
      00110010 00111001 00110000 00110101
      P_{17} = 10111011 01000000 11001011 00101110
   b. SubKunci untuk iterasi kedua :
      P_{16} = P_{16} XOR P_{17}
      P_{16} = 10010010 00010110 11010101 11011001 XOR
      10111011 01000000 11001011 00101110
      P_{16} = 00101001 01010110 00011110 11110111
6. Dalam hal ini penulis, hanya melakukan dua iterasi, dikarenakan total iterasi proses dekripsi adalah 16 putaran.
   Untuk iterasi pertama $i = 0$ yaitu :

   XL = *XL* XOR P_{17}
   XL = 10011010 00101101 00101111 00101010 XOR
   10111011 01000000 11001011 00101110
   XL = 00100001 01101101 11100100 00000100

   Fungsi *F* didapat dari :
   XL dibagi menjadi 4 (*a*, *b*, *c*, *d*) masing-masing 8 bit =
   a = 00100001
   b = 01101101
   c = 11100100
   d = 00000100
   Fungsi *F* : $F(XL) = (((S_0.a + S_1.b \bmod 2^{32}) \text{ XOR } S_{2,}\, c) + S_{3,}d \bmod 2^{32})$
   $S_0.a + S_1.b \bmod 2^{32}$ = (11010001 00110001 00001011 10100110 . 00100001) + (01001011 01111010 01110000 11101001. 01101101) mod 2^{32}
   = (00011010 11110111 01010010 10000000 01100110 +00100000 00100011 00100010 00010011 00110101) mod 2^{32}
   = 00011010 01110100 10010011 10011011
   XOR S_2.c = 00011010 01110100 10010011 10011011 XOR (11101001 00111101 01011010 01101000. 11100100)
   = 00011010 01110100 10010011 10011011 XOR 11001111 10111010 10100100 10000100 10100000
   = 11001111 10100000 11010000 00010111 00111011
   + S3.d mod 2^{32} = (11001111 10100000 11010000 00010111 00111011 + 00111010 00111001 11001110 00110111 . 00000100) mod 2^{32}
   = (11001111 10100000 11010000 00010111 00111011 + 11101000111001110011100011011100) mod 2^{32}

= 10001001 10110111 01010000 00010111
F(XL) = 10001001 10110111 01010000 00010111
XR = *F(XL)* XOR *XR*
XR = 10001001 10110111 01010000 00010111 XOR
10010001 01111100 00111001 00111100
XR = 00011000 11001011 01101001 00101011
Menukar Nilai *XL* dan *XR* :
XL = *XR* ; *XR* = *XL*
XL = 00011000 11001011 01101001 00101011;
XR = 00100001 01101101 11100100 00000100

7. Setelah melakukan 16 iterasi, maka akan menghasilkan nilai baru *XL* dan *XR* masing-masing 32 bit. Tukar kembali *XL* dan *XR*. Setelah itu XOR-kan nilai *XL* dan *XR* : *XR* = *XR* XOR P_1dan *XL* = *XL* XOR P_0
8. Kemudian *XL* dan *XR* digabungkan sehingga menjadi 64 bit
9. Nilai biner tersebut di konversikan ke dalam kode ASCII sehingga menghasilkan plaintext yaitu : UPI YPTK

KESIMPULAN

Berdasarkan keseluruhan proses yang dilakukan untuk membangun Aplikasi Kriptografi File menggunakan Algoritma Blowfish ini dapat disimpulkan bahwa aplikasi ini telah berhasil dibangun dan dapat berfungsi sesuai tujuan, yaitu mengamankan data atau pun informasi yang berupa file (plainteks) dengan mengacak file tersebut sehingga tidak dapat dibaca atau dimengerti. Aplikasi ini juga telah berhasil mengembalikan file yang telah diacak tersebut (cipherteks) seperti semula dengan menggunakan kunci yang sama sewaktu enkripsi. Selain itu, aplikasi ini dapat digunakan untuk melihat kinerja algoritma Blowfish dalam pengimplementasiannya, yaitu bagaimana kecepatan proses enkripsi/dekripsi jika dikaitkan dengan ukuran dari sebuah file. Kecepatan proses enkripsi/dekripsi bergantung pada besarnya ukuran file , semakin besar ukuran file semakin banyak waktu yang dibutuhkan untuk enkripsi/dekripsi. Terjadi penambahan byte pada file hasil enkripsi yang mengakibatkan ukuran file enkripsi dan file plainteks sedikit berbeda, tetapi ketika file enkripsi dikembalikan (didekripsi) ukuran file akan kembali seperti ukuran file plainteksnya.

DAFTAR PUSTAKA

[1] C. A. Sutanto, "Penggunaan Algoritma Blowfish dalam Kriptografi," Bandung, Indonesia, 2010.
[2] Defni, Indri Rahmayun. (2014). Enkripsi SMS (Short Message Service) Pada Telepon Selular Berbasis Android Dengan Metode RC6. Jurnal Momentum Vol.16 No.1. Februari 2014 Jurusan Teknologi Informasi Politeknik Negeri Padang.
[3] E Pratiwi, Apriyanti. (2011). Implementasi Enkripsi Data Dengan Algoritma Blowfish Menggunakan Java Pada Aplikasi Email. Jurnal Jurusan Teknik Komputer Politeknik Telkom Bandung.
[4] Munir, Rinaldi. (2006). Kriptografi. Bandung: Penerbit Informatika.
[5] Rahman, Abdul. (2012). Implementasi Algoritma Serpent Untuk Enkripsi Dan Dekripsi Data File Pada Ponsel Berbasis Android. Jurnal Jurusan Teknik Informatika STMIK GI MDP.
[6] Schneier, Bruce, 1994. Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish), Springer-Verlag.

[7] Schneier, Bruce, 1996, Applied Cryptography, Second Edition, John Wiley & Son, New York.
[8] Tjiharjadi, Semuil., Chandra Wijaya, Marvin. "Pengamanan Data Menggunakan Metoda Enkripsi Simetri Dengan Algoritma Feal". **SNATI** 20, 1-2:1907-5022. 2009
[9] Wahana, 2003, Memahami Model Enkripsi dan Security Data, Andi Offset, Yogyakarta.