Algoritma *Blowfish* untuk Penyandian Pesan

Sugi Guritman,* Meuthia Rachmaniah,* Dian Mardiana **

*) *Staf Jurusan Ilmu Komputer, Fakultas Matematika dan Ilmu Pengetahuan Alam*
**) *Mahasiswa Jurusan Ilmu Komputer, Fakultas Matematika dan Ilmu Pengetahuan Alam*

Abstrak

Meningkatnya penggunaan internet untuk melakukan pengiriman pesan menyebabkan metode pengamanan terhadap pesan menjadi amat penting. Metode pengamanan pesan yang digunakan saat ini adalah metode kriptografi. Secara umum, kriptografi digunakan untuk melakukan penyandian pesan dan autentikasi pesan. Pada penelitian ini teknik yang digunakan adalah teknik kriptografi penyandian pesan menggunakan algoritme Blowfish.

Berdasarkan kunci pengaman yang digunakannya, teknik kriptografi penyandian pesan dibedakan menjadi dua yaitu simetrik dan asimetrik. Blowfish merupakan algoritma kriptografi penyandian pesan yang menggunakan teknik simetrik; artinya kunci yang digunakan pada proses enkripsi sama dengan kunci yang digunakan pada proses dekripsi. Tulisan ini bertujuan untuk mempelajari, mengimplementasikan, dan menganalisis algoritma Blowfish, sehingga dapat diketahui kinerjanya dalam melakukan penyandian serta menjaga kerahasiaan pesan yang disandikannya. Implementasi algoritma ini menggunakan dua modus operasi yaitu Electronic Code-Book (ECB) dan Cipher-Block Chaining (CBC). Sedangkan analisis yang dilakukan meliputi analisis teori, analisis algoritma, analisis keamanan, dan analisis hasil implementasi.

Analisis teori menunjukkan bahwa Blowfish merupakan algoritma kriptografi yang menggunakan kunci simetrik dengan panjang bervariasi asalkan tidak lebih dari 448-bit. Blowfish juga mengkombinasikan fungsi f tak-membalik, key-dependent S-Box, dan jaringan Feistel. Proses enkripsi-dekripsi menggunakan ECB dan operasi CBC memiliki kasus terburuk yang sama yaitu O(n). Meskipun notasi-O pada keduanya sama, pengukuran kecepatan pada hasil implementasi menunjukkan bahwa kecepatan Blowfish dengan ECB lebih baik dibandingkan Blowfish dengan CBC. Namun ditinjau dari segi keamanan, Blowfish menggunakan operasi CBC lebih baik dibandingkan Blowfish menggunakan operasi ECB. Hingga saat ini belum ada attack yang mampu membongkar keamanan Blowfish 16-round. Dengan menggunakan exhaustive key search, kunci rahasia Blowfish dapat ditemukan melalui $7{,}27 \times 10^{134}$ *operasi dekripsi (kasus terburuk). Penelitian Vaudenay tahun 1995 berhasil menganalisis weak key pada algoritma ini, yaitu disebabkan oleh adanya dua entries identik pada suatu S-Box-nya, tetapi penelitian ini belum mampu menunjukkan nilai dari weak key tersebut.*

PENDAHULUAN

A. Latar Belakang

Kemajuan teknologi khususnya dalam bidang telekomunikasi dan komputer telah memungkinkan seseorang melakukan pengiriman pesan kepada orang lain melalui internet. Kegiatan mengirim pesan melalui internet ini sangat beresiko, apalagi untuk pesan yang sifatnya rahasia dan berharga, karena internet merupakan media umum yang rentan akan terjadinya penyusupan oleh pihak yang tidak berhak *(unauthorized person).*

Bentuk ancaman yang dilakukan oleh penyusup dapat berupa ancaman pasif (*passive attack*) atau pun ancaman aktif (*active attack*). Ancaman pasif terjadi jika seorang penyusup secara sengaja mengambil, membaca, dan menampilkan isi dari pesan. Sedangkan ancaman aktif terjadi jika seorang penyusup tidak hanya menampilkan isi pesan, tetapi juga memodifikasi atau bahkan memalsukan isi pesan dengan pesan yang baru.

Untuk mencegah terjadinya kedua bentuk ancaman tersebut telah dikembangkan suatu metode pengamanan pesan yang dapat menjaga kerahasiaan dan keutuhan isi pesan yang dikirim melalui internet. Metode ini dikenal dengan nama teknik kriptografi. Pada awal perkembangannya, teknik kriptografi hanya digunakan untuk tujuan militer dan diplomatik. Akan tetapi mulai tahun 1970-an teknik ini telah digunakan untuk keperluan bisnis dan perorangan *(Heriyanto, 1999).*

Secara umum, teknik kriptografi digunakan untuk melakukan penyandian pesan dan autentikasi pesan *(Prasetya, 2001).* Teknik kriptografi penyandian pesan menekankan pada pencegahan terhadap terjadinya ancaman pasif. Sedangkan teknik kriptografi autentikasi pesan menekankan pada pencegahan terjadinya ancaman aktif. Dalam penelitian ini teknik yang digunakan adalah teknik kriptografi penyandian pesan.

Teknik kriptografi penyandian pesan dibagi menjadi dua yaitu simetrik dan asimetrik. Keduanya menggunakan suatu kunci rahasia tertentu untuk menyandikan sekaligus mengamankan isi suatu pesan. *Blowfish* merupakan algoritma kriptografi penyandian pesan yang menggunakan teknik simetrik. Algoritma ini dikembangkan oleh Bruce Schneier pada tahun 1993. Pada penelitian ini dilakukan implementasi dan analisis terhadap algoritma *Blowfish* untuk mengetahui kinerjanya dalam melakukan penyandian dan mengamankan pesan yang disandikannya.

B. Tujuan

Tujuan dari penelitian ini adalah :

1. Mempelajari dan memahami teknik kriptografi penyandian pesan dengan menggunakan algoritma *Blowfish*.
2. Mengimplementasikan algoritma tersebut ke dalam suatu program komputer dengan menggunakan bahasa pemrograman *Visual Basic 6.0*.
3. Melakukan analisis terhadap algoritma tersebut.

C. Ruang Lingkup

Implementasi algoritma *Blowfish* pada penelitian ini ditujukan untuk melakukan penyandian pada pesan dalam bentuk *file* teks *(txt)*. Implementasi ini menggunakan dua modus operasi yaitu *Electronic CodeBook (ECB)* dan *Cipher-Block Chaining (CBC)*. Untuk mempermudah penelitian, manajemen penentuan nilai kunci dan *Initialization Vector (IV)* diasumsikan telah terjamin keamanannya. Sedangkan analisis yang dilakukan meliputi beberapa faktor yaitu analisis teori, analisis algoritma, analisis keamanan dan analisis hasil implementasi *(kecepatan)*.

TINJAUAN PUSTAKA

A. Kriptografi

Kriptografi berasal dari bahasa Yunani yaitu *kryptós* yang artinya rahasia dan *gráphein* yang artinya menulis. Sedangkan Schneier *(1994a)* mendefinisikan kriptografi sebagai ilmu yang mempelajari teknik-teknik untuk menjaga keamanan pesan. Orang yang melakukannya disebut kriptografer.

Menurut Menezes *et al.* (1997) kriptografi dapat memenuhi kebutuhan umum suatu transaksi yang meliputi :

1. Kerahasiaan *(confidentiality)*
2. Keutuhan *(integrity)*
3. Transaksi tidak bisa disangkal *(non-repudiation)*.

Pada penelitian ini teknik yang digunakan adalah teknik kriptografi penyandian pesan yang hanya menekankan pada aspek kerahasiaan *(confidentiality)*.

Pesan asli yang belum disandikan disebut ***plaintext*** disimbolkan dengan **P**. *Plaintext* dapat berupa berkas teks, rangkaian bit, gambar video digital, atau pun gelombang suara digital. Dalam penelitian ini *plaintext* yang digunakan adalah berkas teks. Sedangkan pesan yang sudah disandikan disebut ***ciphertext***, disimbolkan dengan **C**.

B. Enkripsi dan Dekripsi

Schneier *(1994a)* mendefinisikan enkripsi, disimbolkan dengan E, sebagai proses untuk mengubah suatu *plaintext* menjadi *ciphertext*. Fungsi enkripsi *(E)* terhadap *plaintext (P)* akan menghasilkan *ciphertext (C)* yang secara matematis dapat dinyatakan sebagai berikut :

$$E(P) = C$$

Sedangkan dekripsi, disimbolkan dengan D, sebagai kebalikan dari enkripsi yaitu proses mengembalikan *ciphertext* menjadi *plaintext*. Fungsi dekripsi *(D)* terhadap *ciphertext (C)* akan menghasilkan *plaintext (P)* yang secara matematis dapat dinyatakan sebagai berikut :

$$D(C) = P$$

Proses enkripsi yang dikuti dengan proses dekripsi merupakan proses untuk mengembalikan pesan ke *plaintext* yang asli, sehingga berlaku identitas :

$$D(E(P)) = D(C) = P$$

C. Algoritma Kriptografi

Algoritma kriptografi atau *cipher* adalah rangkaian fungsi matematika yang digunakan dalam proses enkripsi dan dekripsi *(Schneier, 1994a)*. Proses enkripsi menggunakan algoritma enkripsi sedangkan proses dekripsi menggunakan algoritma dekripsi.

Untuk keamanannya, semua algoritma kriptografi tergantung pada kerahasiaan kunci *(disimbolkan dengan k)*. Nilai dari kunci ini akan mempengaruhi fungsi enkripsi dan dekripsi, sehingga menjadi :

$$E_k(P) = C$$
$$D_k(C) = P$$

D. Algoritma Simetrik

Algoritma kriptografi yang digunakan dalam penelitian ini adalah algoritma teknik simetrik. Algoritma merupakan algoritma kriptografi yang menggunakan kunci enkripsi dan kunci dekripsi yang sama.

Schneier *(1994a)* menyatakan proses enkripsi dan dekripsi pada algoritma simetrik dengan fungsi sebagai berikut :

$$E_k(P) = C$$
$$D_k(C) = P$$
$$D_k(E_k(P)) = D_k(C) = P$$

Algoritma simetrik dibedakan menjadi dua yaitu **algoritma *stream*** *(stream cipher)* dan **algoritma blok** *(block cipher)*. Algoritma *stream* adalah algoritma simetrik yang mengoperasikan *plaintext* satu bit per satuan waktu. Sedangkan algoritma blok mengoperasikan *plaintext* dalam sekumpulan bit *(block)* per satuan waktu. *Blowfish* merupakan algoritma simetrik yang masuk ke dalam kategori algoritma blok. Ukuran bloknya adalah 64-bit.

E. Round

Proses enkripsi dan dekripsi pada algoritma blok dilakukan dengan menjalankan suatu proses yang terdiri dari beberapa iterasi *(pengulangan)*. Iterasi ini dikenal dengan sebutan *round (Schneier, 1994a)*.

Jumlah *round* yang digunakan tergantung kepada tingkat keamanan yang diinginkan. Dalam banyak kasus, peningkatan jumlah *round* akan memperbaiki tingkat keamanan suatu algoritma blok.

F. Padding dan Unpadding

Input *plaintext* yang akan dienkripsi oleh algoritma blok akan dibagi menjadi blok-blok yang masing-masing panjangnya n-bit, dengan n sebagai ukuran blok. Jika blok yang digunakan berukuran 64-bit, maka pembagian ini akan mengakibatkan blok terakhir akan memiliki ukuran lebih kecil atau sama dengan 64-bit. Untuk mengatasi hal tersebut dilakukan proses *padding* yaitu penambahan bit-bit isian pada blok terakhir input *plaintext* yang akan dienkripsi.

Padding pada penelitian ini dilakukan dengan cara menambahkan karakter baru yang memiliki kode ASCII 1-8 (Ireland, 2002), dan aturannya adalah sebagai berikut :

$$n_{Pad} = \left(\left(\left\lfloor \frac{c}{8} \right\rfloor + 1 \right) \times 8 \right) - c$$

dengan c adalah jumlah karakter pada blok terakhir dan n_{Pad} adalah jumlah/kode untuk karakter *padding*.

Sebagai contoh, misalkan untuk blok *plaintext* yang berukuran 24-bit *(3-karakter), padding* dilakukan dengan cara menambah 5-karakter baru yang memiliki kode ASCII 5 pada blok tersebut. Dengan pola ini maka pada proses dekripsi, program dapat melakukan proses *unpadding* yaitu menghilangkan 5-karakter terakhir pada blok tersebut.

G. Modus Operasi Algoritma Blok

Modus operasi pada algoritma blok digunakan untuk meningkatkan keamanan algoritma tersebut. Terdapat beberapa modus operasi untuk algoritma blok, diantaranya adalah *Electronic CodeBook (ECB), Cipher-Block Chaining (CBC), Cipher FeedBack (CFB)* dan *Output FeedBack (OFB)*. Implementasi algoritma *Blowfish* dalam penelitian ini hanya menggunakan modus operasi ECB dan CBC.

G.1. Electronic Codebook (ECB)

Modus ECB selalu mengenkripsi setiap blok *plaintext* ke suatu blok *ciphertext* yang sama. Demikian pula sebaliknya untuk proses dekripsi. Secara matematis modus ECB ini dapat dituliskan sebagai berikut :

$$C = E_k(P) \text{ ; untuk enkripsi}$$
$$P = D_k(C) \text{ ; untuk dekripsi}$$

G.2. Cipher-Block Chaining (CBC)

Modus CBC menggunakan mekanisme umpan balik, yaitu hasil enkripsi pada blok sebelumnya digunakan untuk memodifikasi enkripsi blok berikutnya. Setiap blok *ciphertext* yang dihasilkan tidak hanya tergantung pada blok *plaintext* yang membangkitkannya, tetapi juga tergantung pada semua blok *plaintext* sebelumnya.

Dalam modus CBC, sebelum dienkripsi *plaintext* akan di-XOR terlebih dahulu dengan blok *ciphertext* hasil enkripsi dari blok *plaintext* sebelumnya. Secara matematis modus CBC ini dapat dituliskan sebagai berikut :

$$C_i = E_k(P_i \text{ XOR } C_{i-1}) \quad \text{; untuk enkripsi, } i = 1, 2, \ldots\ldots, t$$
$$P_i = C_{i-1} \text{ XOR } D_k(C_i) \quad \text{; untuk dekripsi, } i = 1, 2, \ldots\ldots, t$$

Setelah satu blok *plaintext* dienkripsi, blok *ciphertext* yang dihasilkan disalin ke sebuah *register feedback*. Sebelum blok *plaintext* berikutnya dienkripsi, blok *plaintext* ini di-XOR terlebih dahulu dengan blok *ciphertext* pada *register feedback* dan menjadi input bagi proses enkripsi berikutnya. Blok *ciphertext* selanjutnya disalin ke *register feedback,* untuk kemudian di-XOR dengan blok *plaintext* berikutnya. Begitu. seterusnya hingga akhir pesan.

Dekripsi merupakan proses kebalikannya. Blok *ciphertext* didekripsi secara normal. Setelah blok berikutnya didekripsi, blok *plaintext* hasil dari dekripsi sebelumnya di-XOR dengan blok *ciphertext* pada *register forward.* Lalu blok *ciphertext* selanjutnya disalin ke *register forward,* begitu seterusnya hingga akhir pesan.

Untuk menginisialisasi CBC, *register feedback* dan *register forward* perlu diberi suatu nilai awal yang disebut ***Initialization Vector (IV)***. IV merupakan suatu nilai yang ukurannya sama dengan ukuran blok yang digunakan. Nilai IV ini sebaiknya merupakan nilai yang diambil secara acak, dan untuk keamanannya nilai IV ini harus selalu berbeda untuk setiap pesan.

Karena *Blowfish* merupakan algoritma blok 64-bit, maka nilai IV untuk algoritma ini adalah nilai yang berukuran 64-bit, yang jika direpresentasikan dalam karakter ASCII terdiri atas 8-karakter. Untuk mendapatkan *plaintext* yang benar maka nilai IV pada proses dekripsi harus sama dengan nilai IV pada proses enkripsi.

H. Jaringan Feistel

Hampir semua algoritma blok menggunakan konstruksi jaringan Feistel *(Feistel Network)* yang dikembangkan pada tahun 1970-an oleh Horst Feistel. Jaringan Feistel bekerja pada blok yang panjangnya n-bit. Lalu membaginya menjadi dua bagian yaitu sisi kiri *(L)* dan sisi kanan *(R)* yang masing-masing panjangnya n/2 bit.

Dalam jaringan Feistel, output pada *round* ke-i ditentukan oleh ouput *round* sebelumnya. Secara matematis dapat ditulis sebagai berikut :

$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \text{ XOR } f(R_{i-1}, K_i)$$

Dengan K_i merupakan subkunci yang digunakan pada *round* ke-i dan *f* adalah fungsi yang digunakan berulang dalam setiap *round.*

Hal utama dari konstruksi ini adalah sifatnya yang dapat dikembalikan *(reversible),* sehingga semua algoritma blok jaringan Feistel tidak perlu mengimplementasikan dua buah algoritma berbeda untuk melakukan proses enkripsi dan proses dekripsi. Dengan konstruksi ini satu buah algoritma sudah mampu menangani kedua proses tersebut.

I. Data Encryption Standard (DES)

Data Encryption Standard (DES) merupakan standar enkripsi data yang ditetapkan oleh *National Bureau of Standards (NBS)* Amerika Serikat pada tahun 1977. Pada perkembangan selanjutnya DES semakin banyak digunakan dan telah menjadi standar enkripsi data dunia *(Schneier, 1994a).*

Algoritma DES merupakan algoritma blok 16-*round* yang mengenkripsi data yang panjang bloknya 64-bit dengan menggunakan kunci rahasia 56-bit. DES memiliki delapan buah 6*4-bit *S-Box* statis serta merupakan algoritma kriptografi modern yang menerapkan konstruksi jaringan Feistel.

J. Kriptoanalisis

Kriptoanalisis *(cryptanalysis)* adalah ilmu untuk mendapatkan pesan asli dari pesan yang sudah disandikan tanpa memiliki kunci untuk membuka pesan tersebut. Orang yang melakukannya disebut kriptoanalis. Sedangkan usaha untuk melakukan kriptoanalisis disebut serangan *(attack).*

Jenis *attack* yang paling sering menyerang algoritma blok adalah :

1. *Ciphertext-only attack.* Dalam penyerangan ini kriptoanalis memiliki *ciphertext* dari sejumlah pesan yang seluruhnya telah dienkripsi menggunakan algoritma yang sama.
2. *Known-plaintext attack.* Dalam penyerangan ini, kriptoanalis memiliki akses tidak hanya ke *ciphertext* sejumlah pesan, namun ia juga memiliki *plaintext* pesan-pesan tersebut
3. *Chosen-plaintext attack.* Pada penyerangan ini, kriptoanalis tidak hanya memiliki akses atas *ciphertext* dan *plaintext* untuk beberapa pesan, tetapi ia juga dapat memilih *plaintext* yang akan dienkripsi

K. *Exhaustive Key Search*

Exhaustive key search atau Brute-force search adalah suatu teknik dasar yang digunakan kriptoanalis untuk mencoba setiap kunci yang mungkin sampai ditemukan kunci yang sebenarnya. Untuk suatu algoritma blok n-bit dengan kunci k-bit dibutuhkan $\lceil (k+4)/n \rceil$ pasangan *plaintext-ciphertext* yang dienkripsi menggunakan kunci rahasia k. Pada kasus terburuk, *exhaustive key search* dapat menemukan kunci K tersebut dengan melakukan operasi dekripsi sebanyak 2^{k-1}. Sebagai contoh, misalkan ukuran kunci yang digunakan adalah 56-bit maka untuk menemukan nilai dari kunci tersebut dibutuhkan satu pasang *plaintext-ciphertext* dan 2^{55} proses dekripsi.

L. Kriptoanalisis Differensial

Kriptoanalisis differensial (*differential cryptanalysis)* merupakan metode serangan yang diperkenalkan oleh Eli Biham dan Adi Shamir *(Biham & Shamir, 1990)*. Ide dasar dari *attack* ini adalah menganalisis efek yang ditimbulkan oleh perbedaan pada suatu pasangan *plaintext* terhadap perbedaan yang terjadi pada resultan pasangan *ciphertext* yang dihasilkan.

Perbedaan-perbedaan ini digunakan untuk menentukan peluang kunci-kunci yang mungkin, dan selanjutnya digunakan untuk menentukan kunci enkripsi yang sebenarnya. Pada algoritma blok, struktur perbedaan ini dipilih sebagai hasil XOR yang tetap dari dua buah *plaintext.*

M. Desain S-Box

S-Box merupakan suatu bentuk substitusi sederhana, yaitu pemetaan dari m-bit input menjadi n-bit output. Suatu *S-Box* dengan m-bit input dan n-bit output dinamakan m*n-bit *S-Box*. Secara umum, *S-Box* merupakan bagian yang menjadi pengaman bagi algoritma blok. Semakin besar ukuran yang digunakan maka desain *S-Box* tersebut akan semakin tahan terhadap serangan kriptoanalisis differensial.

Selain ukuran yang besar, untuk membuat desain *S-Box* semakin tahan terhadap kriptoanalisis differensial maka sebaiknya nilai *S-Box* ini merupakan nilai yang digabungkan dengan nilai kunci yang digunakan *(key-dependent S-Box)*. Penggabungan ini menyebabkan nilai *S-Box* tidak diketahui *(tersembunyi)*, sehingga menyulitkan kriptoanalis yang ingin mencoba mengetahui strukturnya.

Algoritma *Blowfish* menggunakan empat buah 8*32-bit *key-dependent S-Box*. Masing-masing *S-Box* tersebut terdiri atas 256 *entries*.

N. Weak Key

Weak key adalah suatu kunci rahasia pada algoritma blok dengan suatu nilai tertentu yang dapat memperlihatkan suatu keteraturan yang terjadi pada proses enkripsi. Keteraturan ini akan mempermudah kerja seorang kriptoanalis yang mencoba untuk melakukan *attack* terhadap pesan yang dienkripsi dengan menggunakan kunci tersebut.

O. Analisis Algoritma

Dalam penelitian ini, algoritma *Blowfish* dievaluasi berdasarkan keadaan kompleksitas waktu untuk waktu terburuk, dinotasikan dengan *O (big O)*. Kasus terburuk untuk algoritma ini didefinisikan sebagai waktu maksimum yang diperlukan algoritma tersebut untuk menyelesaikan suatu pekerjaan, yaitu waktu yang berlangsung sejak dimulainya algoritma hingga selesai.

DESKRIPSI ALGORITMA *BLOWFISH*

Blowfish merupakan algoritma kriptografi yang didesain untuk beroperasi pada blok pesan 64-bit dan menggunakan kunci yang panjangnya dapat mencapai 448-bit *(56-byte)*. Algoritma ini terdiri dari dua bagian utama yaitu ekspansi kunci *(key expansion)* dan proses enkripsi. Sedangkan proses dekripsi menggunakan proses yang sama persis dengan proses enkripsi, hanya berbeda dalam urutan subkunci yang digunakannya.

Ekspansi kunci mengubah kunci yang dapat mencapai 448-bit menjadi beberapa *array* subkunci dengan total 4168-byte. Sedangkan proses enkripsi dan dekripsi terdiri dari iterasi fungsi sederhana sebanyak *16-round*. Operasi yang digunakan adalah operasi penambahan dan operasi XOR pada variabel 32-bit. Tambahan operasi lainnya adalah empat penelusuran tabel *(tabel lookup) array* berindeks untuk setiap *round.*

Blowfish menggunakan subkunci yang besar. Subkunci ini harus dihitung sebelum proses enkripsi atau dekripsi dilakukan. *Blowfish* memiliki subkunci *P-array* yang terdiri dari delapan belas 32-bit subkunci :

$$P_1, P_2, P_3, \ldots, P_{18}$$

dan empat buah **8*32-bit *S-Box*** yang masing-masing terdiri atas 256 *entries* :

$$\begin{array}{l} S_{1,0}, S_{1,1}, S_{1,2}, \ldots, S_{1,255} \\ S_{2,0}, S_{2,1}, S_{2,2}, \ldots, S_{2,255} \\ S_{3,0}, S_{3,1}, S_{3,2}, \ldots, S_{3,255} \\ S_{4,0}, S_{4,1}, S_{4,2}, \ldots, S_{4,255} \end{array}$$

A. Ekspansi Kunci

Ekspansi kunci adalah proses untuk membangkitkan subkunci yang langkahnya adalah sebagai berikut :

1. Inisialisasi delapan belas *P-array* dan kemudian empat buah *S-Box* secara berurutan dengan string yang tetap. String ini terdiri atas digit heksadesimal bilangan pi (π), contoh :

   $$P_1 = 0x243f6a88$$
   $$P_2 = 0x85a308d3$$
   $$P_3 = 0x13198a2e$$
   $$P_4 = 0x3707344$$

   Digit heksadesimal bilangan pi merupakan deret bilangan pi dalam bentuk heksadesimal (π = 3,243f6a8885a308d313198a2e03707344 A4093822 ...) yang dapat dibangkitkan oleh formula Bailey-Borwein-Plouffe *(Finch, 1995)*.
2. XOR P_1 dengan 32-bit pertama dari kunci, XOR P_2 dengan 32-bit kedua dari kunci dan seterusnya untuk setiap bit dari kunci *(mungkin sampai P_{14})*. Ulangi terhadap bit kunci sampai seluruh *P-array* di-XOR dengan bit kunci.
3. Enkripsi semua string nol dengan algoritma *Blowfish* dengan menggunakan subkunci seperti dijelaskan pada langkah *(1)* dan *(2)*.
4. Ganti P_1 dan P_2 dengan keluaran dari langkah *(3)*
5. Enkripsi keluaran dari langkah *(3)* dengan algoritma *Blowfish* dengan subkunci yang sudah dimodifikasi.
6. Ganti P_3 dan P_4 dengan keluaran dari langkah *(5)*.
7. Lanjutkan proses tersebut, ganti seluruh elemen dari *P-array*, dan kemudian keempat *S-Box* secara berurutan, dengan keluaran yang berubah secara kontinu dari algoritma *Blowfish*.

Total iterasi yang diperlukan untuk menghasilkan subkunci *Blowfish* ini adalah :

$$\frac{((4 \times 256) + 18)}{2} = 521 \text{ iterasi}$$

B. Proses Enkripsi

Input proses enkripsi merupakan *plaintext* 64-bit, X. Untuk mengenkripsi X dilakukan langkah-langkah sebagai berikut :

Bagi X menjadi dua 32-bit: X_L, X_R
Untuk i = 1 sampai 16
  $X_L = X_L$ XOR P_i
  $X_R = f(X_L)$ XOR X_R
  Tukar X_L dan X_R
Tukar X_L dan X_R *(batalkan penukaran terakhir)*
  $X_R = X_R$ XOR P_{17}
  $X_L = X_L$ XOR P_{18}
  Kombinasikan kembali X_L dan X_R

Sedangkan fungsi f ditentukan sebagai berikut:

Bagi X_L, menjadi empat bagian 8-bit :
a, b, c dan d

$$f(X_L) = ((S_{1,a} + S_{2,b} \bmod 2^{32}) \text{ XOR } S_{3,c}) + S_{4,d} \bmod 2^{32}$$

C. Proses Dekripsi

Proses dekripsi sama persis dengan proses yang dilakukan pada proses enkripsi, tetapi P_1, P_2, P_3,. . . ., P_{18} digunakan pada urutan yang berbalik.

HASIL DAN PEMBAHASAN

A. Analisis Teori

Analisis teori dilakukan untuk mengetahui teori kriptografi apa saja yang mendukung *Blowfish*, serta menganalisis efek dari teori tersebut terhadap algoritma ini. Dari analisis ini diketahui bahwa *Blowfish* merupakan algoritma blok 64-bit yang mengkombinasikan fungsi f tak-membalik, *key-dependent S-Box (S-Box yang digabungkan dengan kunci)*, dan jaringan Feistel *(KremlinEncrypt, 2000)*.

Penggunaan fungsi f tak-membalik merupakan bagian yang paling menarik. Fungsi ini menggunakan aritmetika modular (operasi sisa-bagi) untuk membangkitkan *entries-entries* ke dalam *S-Box*. Sifat tak-membalik ini dapat dijelaskan pada ***Tabel 1.*** dengan menggunakan fungsi $f(x) = x^2 \bmod 7$ *(KremlinEncrypt, 2000)*.

Tabel 1. Fungsi $f(x) = x^2 \bmod 7$

x	1	2	3	4	5	6	7
x^2	1	4	9	16	25	36	49
$X^2 \bmod 7$	1	4	2	2	4	1	0

Berdasarkan output yang dihasilkan pada ***Tabel 1.***, tidak ada fungsi yang dapat membangkitkan input spesifik bagi $f(x)$. Sebagai contoh, jika diketahui bahwa fungsi tersebut memiliki output bernilai 4 di beberapa nilai x, maka tidak ada cara untuk mengetahui apakah nilai x tersebut adalah 2, 5, atau nilai lain yang memiliki fungsi $f(x) = 4$.

Blowfish menggunakan mod 2^{32} *(atau sekitar 4 milyar)* pada fungsi f-nya. Penggunaan fungsi f ini akan mempersulit kerja seorang kriptoanalis untuk mendapatkan pesan asli saat mencoba melakukan penyerangan jenis *ciphertext-only attack*.

Tidak seperti DES yang menggunakan *S-Box* yang statis, *Blowfish* menggunakan *S-Box* dinamis yang dibangkitkan melalui pengulangan aplikasi algoritma *Blowfish* itu sendiri terhadap kunci *(key-dependent S-Box)* saat proses ekspansi kunci dilakukan. Penggabungan *S-Box* dengan kunci ini menyebabkan nilai dari *S-Box Blowfish* tidak diketahui sehingga akan mempersulit kerja kriptoanalis yang mencoba melakukan *attack* jenis kriptoanalisis differensial.

Elemen penting lainnya dari *Blowfish* adalah konstruksi jaringan Feistel. Dengan konstruksi ini, algoritma *Blowfish* memiliki dua sifat unik yaitu proses enkripsi dan dekripsi menggunakan fungsi *f* yang sama serta memiliki kemampuan untuk mengunakan fungsi *f* berulang kali *(multiple-time)*. Pada *Blowfish* fungsi *f* ini diulang sebanyak 16 kali *(16-round)*. Pengulangan sebanyak 16 kali dipilih karena 16 merupakan ukuran yang sangat sesuai dengan ukuran *P-array* yang digunakan pada proses ekspansi kunci, serta mendukung penggunaan kunci *Blowfish* yang panjangnya mencapai 448-bit *(Schneier, 1994)*.

B. Analisis Algoritma

Analisis algoritma dilakukan dengan asumsi bahwa mesin yang digunakan adalah model *Random-Access Machine (RAM)*, berprosesor tunggal. Pada mesin jenis ini, instruksi-instruksi program dieksekusi baris perbaris secara berturut-turut *(Cormen et al., 1990)*.

B.1. Analisis Algoritma Enkripsi

Algoritma enkripsi *Blowfish* diimplementasikan dalam dua modus operasi yaitu ECB dan CBC.

1. Langkah untuk melakukan enkripsi modus ECB adalah :
   a. ekspansi kunci
   b. *padding*
   c. enkripsi blok *plaintext* 64-bit
   d. output

   Waktu eksekusi pada langkah a, b, dan d adalah konstan karena tidak dipengaruhi oleh ukuran inputnya, misalkan α_1. Sedangkan langkah c dipengaruhi oleh jumlah blok input *plaintext* 64-bit, jadi untuk input berukuran n blok diperlukan waktu eksekusi sebesar $\varepsilon_1 n$.

   Secara keseluruhan waktu eksekusi enkripsi dengan modus operasi ECB adalah $\varepsilon_1 n + \alpha_1$, dengan ε_1 dan α_1 adalah suatu konstanta dan n adalah jumlah blok input. Jadi notasi-*O* untuk kasus terburuk proses enkripsi modus ECB adalah :

   $$E_{(ECB)} = \varepsilon_1 n + \alpha_1 \in O(n)$$

2. Langkah untuk melakukan enkripsi modus CBC adalah :
   a. Inisialisasi nilai untuk *initialization vector (IV)*, dan simpan di *register feedback*
   b. ekspansi kunci
   c. *padding*
   d. proses enkripsi
      d.1 XOR input blok *plaintext* 64-bit dengan blok *ciphertext* 64-bit pada *register feedback*
      d.2 enkripsi blok *plaintext* 64-bit hasil langkah [d.1]
      d.3 salin blok *ciphertext* 64-bit hasil langkah [d.2] ke *register feedback*
   e. output

   Waktu eksekusi a, b, c dan e adalah konstan karena tidak dipengaruhi oleh ukuran inputnya, misalkan α_2. Sedangkan langkah d dipengaruhi oleh jumlah blok input *plaintext* 64-bit, jadi untuk input berukuran n blok diperlukan waktu eksekusi sebesar $\varepsilon_2 n$.
   Secara keseluruhan waktu eksekusi enkripsi dengan modus operasi ECB adalah $\varepsilon_2 n + \alpha_2$, dengan ε_2 dan α_2 adalah suatu konstanta dan n adalah jumlah blok input. Jadi notasi-*O* untuk kasus terburuk proses enkripsi modus CBC adalah :

   $$E_{(CBC)} = \varepsilon_2 n + \alpha_2 \in O(n)$$

B.2. Analisis Algoritma Dekripsi

Sama halnya seperti pada algoritma enkripsi, algoritma dekripsi *Blowfish* juga diimplementasikan dalam modus operasi ECB dan CBC.

1. Langkah untuk melakukan dekripsi modus ECB adalah :
   a. ekspansi kunci
   b. dekripsi blok *ciphertext* 64-bit
   c. *unpadding*
   d. output

   Waktu eksekusi pada langkah a, c, dan d adalah konstan karena tidak dipengaruhi oleh ukuran inputnya, misalkan β_1. Sedangkan langkah b dipengaruhi oleh jumlah blok input

ciphertext 64-bit, jadi untuk input berukuran n blok diperlukan waktu eksekusi sebesar $\delta_1 n$.
Secara keseluruhan waktu eksekusi dekripsi dengan modus operasi ECB adalah $\delta_1 n + \beta_1$, dengan δ_1 dan β_1 suatu konstanta dan n adalah jumlah blok input. Jadi notasi-*O* untuk kasus terburuk proses dekripsi modus ECB adalah :

$$D_{(ECB)} = \delta_1 n + \beta_1 \in O(n)$$

2. Langkah untuk melakukan dekripsi modus CBC adalah :
   a. Inisialisasi nilai untuk *initialization vector (IV)*, dan simpan di *register forward*
   b. ekspansi kunci
   c. proses dekripsi
      c.1 salin blok *ciphertext* 64-bit ke *register value*
      c.2 dekripsi blok *ciphertext* 64-bit
      c.3 XOR blok *plaintext* 64-bit hasil langkah [c.2] dengan blok *ciphertext* 64-bit pada *register forward*
      c.4 salin blok *ciphertext* 64-bit pada *register value (langkah* [c.1]) ke *register forward*
   d. *unpadding*
   e. output

   Waktu eksekusi a, b, d dan e adalah konstan karena tidak dipengaruhi oleh ukuran inputnya, misalkan β_2. Sedangkan langkah c dipengaruhi oleh jumlah blok input *ciphertext* 64-bit, jadi untuk input berukuran *n* blok diperlukan waktu eksekusi sebesar $\delta_2 n$.
   Secara keseluruhan waktu eksekusi dekripsi dengan modus operasi ECB adalah $\delta_2 n + \beta_2$, dengan δ_2 dan β_2 adalah suatu konstanta dan n adalah jumlah blok input. Jadi notasi-*O* untuk kasus terburuk proses dekripsi modus CBC adalah :

$$D_{(CBC)} = \delta_2 n + \beta_2 \in O(n)$$

C. Analisis Keamanan

Dalam melakukan proses enkripsi-dekripsi, *Blowfish* menggunakan suatu kunci rahasia yang panjangnya bervariasi asalkan tidak lebih dari 448-bit. Variasi panjang bit kunci ini akan memperluas ruang kunci algoritma ini, sehingga mempersulit kerja seorang kriptoanalis yang mencoba melakukan penyerangan jenis *exhaustive key search*. Usaha yang dilakukan untuk menemukan kunci rahasia *Blowfish* dengan *exhaustive key search* adalah sebanyak $\sum_{k=1}^{448} 2^{k-1}$ atau sekitar $7{,}27\times10^{134}$ proses dekripsi (untuk kasus terburuk)

Tahun 1995 Serge Vaudenay melakukan penelitian untuk menganalisis *weak key* pada algoritma ini dengan menggunakan jenis *attack* kriptoanalisis differensial *(Vaudenay, 1996)*. Dari penelitian ini diketahui bahwa penyebab terjadinya *weak key* pada *Blowfish* adalah adanya *collision*, yaitu munculnya dua *entries* identik pada suatu *S-Box*. Secara acak, peluang munculnya *collision* pada empat *S-Box Blowfish* adalah 2^{-15}.

Kesimpulan yang didapat dari penelitian Vaudenay *(1996)* adalah sebagai berikut :

a. Dengan menggunakan *S-Box* yang diketahui dan bukan merupakan *key-dependent*, kriptoanalisis differensial membutuhkan 2^{48} *chosen-plaintext* untuk menemukan semua nilai subkunci *P-array* pada varian *Blowfish 8-round*.
b. Dengan suatu *weak key* tertentu yang menyebabkan munculnya *collision* pada *S-Box*, *attack* ini membutuhkan 2^{23} *chosen-plaintext* pada *Blowfish 8-round* dan 3×2^{51} *chosen-plaintext* pada *Blowfish 16-round* (*S-Box-nya juga diasumsikan telah diketahui dan bukan key-dependent*).
c. Dengan suatu *S-Box* yang tidak diketahui, *attack* ini membutuhkan 2^{22} *chosen-plaintext* untuk mendeteksi bahwa kunci yang sedang digunakan merupakan *weak key*. Tetapi *attack* ini belum mampu menentukan nilai dari *S-Box*, *P-array*, ataupun nilai dari *weak key* tersebut, juga hanya dapat dilakukan pada varian *Blowfish 8-round*.

Keamanan suatu algoritma kriptografi tergantung pada kunci rahasia yang digunakannya. Algoritma kriptografi dikatakan aman apabila usaha untuk membongkar kunci rahasia tersebut memerlukan waktu yang sangat lama, sehingga usaha pembongkaran tersebut baru akan berhasil setelah pesan sudah tidak berlaku lagi *(Prasetya, 2001)*.

Tabel 2. menyajikan perbandingan usaha yang dilakukan untuk membongkar kunci rahasia algoritma DES 56-bit dan *Blowfish* dengan menggunakan kriptoanalisis differensial dan *exhaustive key search*. Kriptoanalisis differensial pada DES 56-bit merupakan hasil penelitian Biham dan Shamir *(1992)* dan pada *Blowfish* merupakan hasil penelitian Vaudenay *(1996)*.

Tabel 2. ***Perbandingan usaha untuk membongkar kunci rahasia algoritma DES 56-bit dan Blowfish***

Algoritma		Kriptoanalisis Differensial	*Exhaustive Key Search*
DES 56-bit	*8-round*	2^{14} *chosen-plaintext* (0,016 det)*	2^{55} operasi (± 1142 thn)**
	16-round	2^{47} *chosen-plaintext* (4,46 thn)*	
Blowfish	*8-round*	2^{48} *chosen-plaintext* dengan *known S-Box* (8,9 thn)*	$7{,}27 \times 10^{134}$ operasi (± $2{,}3 \times 10^{121}$ thn)**
	16-round	Belum ada	

* *Asumsi : menggunakan mesin yang mampu menganalisis 10^6 chosen-plaintext per detik.*

** *Asumsi : menggunakan mesin yang mampu mencoba 10^6 kunci per detik.*

Dari ***Tabel 2.*** ini terlihat bahwa waktu untuk membongkar *Blowfish* lebih lama dibandingkan waktu untuk membongkar DES 56-bit, sehingga dapat disimpulkan bahwa tingkat keamanan *Blowfish* lebih baik dibandingkan dengan DES 56-bit. Hingga saat ini selain *exhaustive key search*, belum ada *attack* lain yang berhasil membongkar keamanan kunci pada *Blowfish 16-round*.

D. Analisis Hasil Implementasi

Implementasi algoritma *Blowfish* ini menggunakan spesifikasi perangkat keras dan perangkat lunak sebagai berikut :

<u>Perangkat Keras</u>
1. Prosesor Pentium II 300 MHz
2. RAM 64 MB
3. Harddisk 4,3 GB

<u>Perangkat Lunak</u>
1. Sistem operasi *Win 98 SE*
2. Bahasa pemrograman *Visual Basic 6.0*

Dari hasil implementasi ini dilakukan pengukuran kecepatan proses enkripsi dan dekripsi. Pengukuran ini dilakukan sebanyak 10 kali percobaan dengan menggunakan ukuran input bervariasi. Hasil pengukuran ini disajikan pada ***Tabel 3.*** dan ***Tabel 4.***

Tabel 3. ***Hasil pengukuran kecepatan proses enkripsi***

File	Ukuran Input * (Byte)	Ukuran Output ** (Byte)	Waktu (ms)		Kecepatan Rata2 (Byte/ms)	
			ECB	CBC	ECB	CBC
1	4	8	18	18	0,44	0,44
2	94	96	20	20	4,80	4,80
3	795	800	25	26	32,00	30,77
4	4.084	4.088	61	96	67,02	42,58
5	10.545	10.552	129	132	81,80	79,94
6	16.606	16.608	178	186	93,30	89,29
7	24.587	24.592	278	280	88,46	87,83
8	50.902	50.904	555	562	91,72	90,58
9	106.081	106.088	1.042	1.140	101,81	93,06
10	135.600	135.608	1.330	1.454	101,96	93,27
Kecepatan Rata-rata (Byte/ms)					**66,34**	**61,26**

* *Ukuran file sebelum padding*

** *Ukuran file setelah padding*

Tabel 3. menunjukkan bahwa kecepatan enkripsi *Blowfish* dengan modus ECB lebih baik dibandingkan dengan kecepatan enkripsi *Blowfish* dengan modus CBC. Hal ini disebabkan oleh adanya operasi XOR antara input blok *plaintext* dengan blok *ciphertext* pada *register feedback*, sehingga waktu penyelesaian CBC lebih lama dari pada ECB.

Namun demikian, operasi XOR ini membuat pengamanan CBC terhadap pesan lebih baik dibandingkan dengan ECB, karena operasi XOR tersebut menimbulkan ketergantungan setiap blok *ciphertext* dengan blok *plaintext*-nya dan semua blok *plaintext* sebelumnya. Blok *ciphertext* yang dihasilkan tidak mengimplikasikan secara identik ke blok *plaintext*-nya, sehingga dapat mempersulit kerja seorang kriptoanalis yang mencoba memetakan suatu *ciphertext* ke *plaintext*-nya. Sedangkan pada ECB, setiap blok *ciphertext* yang dihasilkan mengimplikasikan secara identik ke blok *plaintext*-nya. Dengan alasan ini, ECB tidak baik untuk mengenkripsi pesan yang jumlah bloknya lebih dari satu atau pun untuk suatu aplikasi yang kuncinya digunakan berulang-ulang untuk mengenkripsi lebih dari satu pesan berblok-tunggal.

Tabel 4. *Hasil pengukuran kecepatan proses dekripsi*

File	Ukuran Input * (Byte)	Ukuran Output ** (Byte)	Waktu (ms)		Kecepatan Rata2 (Byte/ms)	
			ECB	CBC	ECB	CBC
1	8	4	18	18	0,44	0,44
2	96	94	19	21	5,05	4,57
3	800	795	25	34	32,00	23,53
4	4.088	4084	57	59	71,72	69,29
5	10.552	10.545	120	126	87,93	83,75
6	16.608	16.606	179	188	92,78	88,34
7	24.592	24.587	257	272	95,69	90,41
8	50.904	50.902	513	547	99,23	93,06
9	106.088	106.081	1.141	1.123	92,98	94,47
10	135.608	135.600	1.335	1.433	101,58	94,63
Kecepatan Rata-rata (Byte/ms)					67,94	64,25

* *Ukuran file* sebelum *unpadding*

** *Ukuran file* setelah *unpadding*

Tabel 4. menunjukkan bahwa kecepatan dekripsi *Blowfish* dengan modus ECB lebih baik dibandingkan dengan kecepatan dekripsi *Blowfish* dengan modus CBC. Hal ini terjadi karena untuk mendapatkan *plaintext* yang benar, setiap blok *plaintext* yang dihasilkan harus di-XOR terlebih dahulu dengan blok *ciphertext* pada *register forward*. Dengan demikian untuk melakukan proses dekripsi yang benar, selain dibutuhkan kunci yang benar, CBC juga membutuhkan blok *ciphertext* awal *(Initialization Vector, IV)* yang benar juga.

KESIMPULAN DAN SARAN

A. Kesimpulan

Kesimpulan yang didapatkan dari penelitian ini adalah sebagai berikut :

i. *Blowfish* merupakan algoritma blok 64-bit yang menggunakan kunci dengan panjang bervariasi asalkan tidak lebih dari 448-bit *(56-byte)*.
ii. Algoritma *Blowfish* mengkombinasikan fungsi *f* tak-membalik, *key-dependent S-Box*, dan jaringan Feistel.
iii. Proses enkripsi-dekripsi dengan modus ECB dan dengan modus CBC memiliki kasus terburuk yang sama yaitu $O(n)$.
iv. Kecepatan *Blowfish* dengan modus ECB lebih baik dibandingkan kecepatan *Blowfish* dengan modus CBC, namun jika ditinjau dari segi keamanannya, *Blowfish* dengan modus CBC lebih aman dibandingkan *Blowfish* dengan modus ECB.
v. *Blowfish* memiliki tingkat keamanan yang lebih baik dibandingkan DES 56-bit. Hingga saat ini belum ada *attack* yang mampu membongkar keamanan *Blowfish* yang lengkap menggunakan 16-*round*.
vi. Pada kasus terburuk, dengan menggunakan *exhaustive key search* kunci *Blowfish* dapat ditemukan dengan melakukan $7,27 \times 10^{134}$ proses dekripsi.
vii. *Weak key* pada *Blowfish* disebabkan oleh adanya *collision (Vaudenay, 1996)*, yaitu munculnya dua *entries* identik pada suatu *S-Box*-nya.

B. Saran

Untuk penelitian selanjutnya, disarankan hal-hal sebagi berikut :

1. Menggunakan modus operasi lainnya seperti *Cipher FeedBack (CFB)* dan *Output FeedBack (OFB)*, serta dengan memperhitungkan masalah perambatan yang terjadi pada modus-modus tersebut.
2. Dikombinasikan dengan teknik kriptografi autentikasi pesan.
3. Pemahaman mengenai *key space*, fungsi *f* tak-membalik, *key-dependent S-Box*, jaringan Feistel, dan jenis-jenis *attack* yang dapat membongkar keamanan algoritma blok sangat diperlukan, sehingga dapat menemukan kelemahan-kelemahan *Blowfish* dalam mengamankan pesan yang disandikannya.

DAFTAR PUSTAKA

Biham, E. & A. Shamir. 1990. *Differential Cryptanalysis of DES-like Cryptosystem*, hlm. 2-21. Di dalam *Advances in Cryptology – CRYPTO'90 Proceedings*. Springer-Verlag.

Biham, E. & A. Shamir. 1992. *Differential Cryptanalysis of the Full 16-round DES*, hlm.487-496. Di dalam *Advances in Cryptology – CRYPTO'92 Proceedings*. Springer-Verlag.

Cormen, T.H., C.E. Leiserson & R.L. Rivest. 1990. *Introduction to Algorithms*. The MIT Press, Massachusetts-London

Finch, S. 1995. *The Miraculous Bailey-Borwein-Plouffe Pi Algorithm*. MathSoft Engeering & Education, Inc. http://pauillac.inria.fr/algo/bsolve/constant/pi/plouffe/plouffe.html. *[25 September 2002]*.

Heriyanto, T. 1999. *Pengenalan Kriptografi*. http://www.tedi-h.com/papers/p_kripto.html. *[1 April 2002]*.

Ireland, D. 2002. *Blowfish : a Visual Basic Version*. DI Management Services. Sydney-Australia. http://www.di.mgt.com.au/crypto.html. *[24 Juli 2002]*

KremlinEncrypt. 2000. *Concepts of Cryptography*. http://www.kremlinencrypt.com/crypto.html. *[24 Juli 2002]*.

Menezes, A. P. van Oorschot. & S. Vanstone. 1997. *Handbook of Applied Cryptography*. CRC Press, New York.

Prasetya, M. 2001. *Message Digest 5 (MD5)* dan *Secure Hash Algorithm 1 (SHA1)* untuk Autentikasi Pesan. Skripsi. Jurusan Ilmu Komputer FMIPA IPB, Bogor.

Schneier, B. 1994a. *Applied Cryptography - Protocol, Algorithm, and Source Code in C*. John Willey & Sons, Inc., New York.

Schneier, B. 1994b. *Description of a New Variable-Length Key, 64-Bits Block Cipher*, hlm. 191-204. Di dalam *Fast Software Encryption. Cambridge Security Workshop Proceedings*, Springer-verlag.

Schneier, B. 1996. *Applied Cryptography - Protocol, Algorithm, and Source Code in C*. second edition. John Willey & Sons. Inc., New York.

Vaudenay, S. 1996. *On The Weak Key of Blowfish*, hlm. 27-35. Di dalam *Fast Software Encryption, Third International Workshop (LNCS 1039) Proceedings*. Springer-Verlag. http://www.lasec.epfl.ch/~vaudenay/pub.html. *[22 Juli 2002]*.