MAKALAH TUGAS AKHIR

STUDI DAN IMPLEMENTASI ALGORITMA BLOWFISH UNTUK ENKRIPSI EMAIL

OLEH :

CHUMAIDI RAHMAN **7408.040.503**

JURUSAN TEKNOLOGI INFORMASI
POLITEKNIK ELEKTRONIKA NEGERI SURABAYA
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2009

STUDI DAN IMPLEMENTASI ALGORITMA BLOWFISH UNTUK ENKRIPSI EMAIL

IMPLEMENTATION AND STUDY BLOWFISH ALGOTYTHM FOR EMAIL ENCRYPTION

Chumaidi Rahman [1], Isbat Uzzin Nadhori [2], Kholid Fathoni[2]
Mahasiswa Jurusan Teknologi Informasi [1] , Dosen Pembimbing [2]
Politeknik Elektronika Negeri Surabaya
Institut Teknologi Sepuluh Nopember
Kampus PENS-ITS Keputih Sukolilo Surabaya 60111
Telp (+62)31-5947280, 5946114, Fax. (+62)31-5946114
Email : x_newbies@yahoo.com

Makalah Proyek Akhir

Abstrak

Blowfish, merupakan sebuah kunci rahasia yang dibuat dengan menggunakan blok cipher. Di dalamnya terdapat jaringan Feistel, iterasinya hanya menggunakan fungsi enkripsi sederhana sebanyak 16 putaran. Ukuran blok adalah 64 bit, dan kunci dapat berupa panjang hingga 448 bit. Meskipun ada fase inisialisasi yang kompleks diperlukan sebelum enkripsi dilakukan, tetapi sebenarnya enkripsi data yang dilakukan sangat efisien jika digunakan pada mikroprosesor besar.

Kata kunci: Encryption Blowfish, Cipher Blok

Abstract

Blowfish, is a secret-key block cipher, is proposed. It is a Feistel network, iterating a simple encryption function 16 times. The block size is 64 bits, and the key can be any length up to 448 bits. Although there is a complex initialization phase required before any encryption can take place, the actual encryption of data is very efficient on large microprocessors..

Keyword: Blowfish Encryption, Block Cipher

1. PENDAHULUAN

Perkembangan dunia informatika yang sangat pesat saat ini membawa pertumbuhan dunia ke dalam masa teknologi informasi menjadi ujung tombak kemajuan. Karena itulah nilai informasi saat ini sangat tinggi dan penting. Teknologi informasi yang telah terjalin saat ini berdiri di atas media komunikasi sebagai media penyampaian informasi dari suatu tempat ke tempat lainnya. Informasi-informasi yang ingin disampaikan berjalan melalui media komunikasi tersebut.

Media komunikasi yang banyak digunakan tentu harus merupakan media yang mudah dijangkau dan digunakan oleh semua orang. Contoh media komunikasi yang saat ini sering digunakan adalah telepon, jaringan internet dan email. Kemudahan pengaksesan media komunikasi oleh semua orang membawa dampak bagi keamanan informasi atau pesan yang menggunakan media komunikasi tersebut. Informasi menjadi sangat rentan untuk diketahui, diambil dan dimanipulasi oleh pihak-pihak yang tidak berkepentingan.

Karena itulah dibutuhkan suatu metode yang dapat menjaga kerahasiaan informasi ini. Metode yang dimaksud adalah kriptografi yaitu sebuah seni dan bidang keilmuan dalam penyandian informasi atau pesan dengan tujuan menjaga keamanannya. Walaupun telah berkembang sejak zaman dahulu kala, teknik kriptografi yang dibutuhkan masa kini harus menyesuaikan dirinya terhadap meluasnya penggunaan komputer dijital pada masa kini.

Penggunaan komputer dijital mendorong berkembangnya kriptografi modern yang beroperasi dalam mode bit (satuan terkecil dalam dunia dijital) daripada dalam mode karakter yang biasa digunakan dalam kriptografi klasik. Akan tetapi, kriptografi modern tetap menggunakan prinsip substitusi dan transposisi yang sudah berkembang sejak kriptografi klasik. Tetapi

kriptografi moder memiliki tingkat kerumitan yang lebih dibanding kriptografi klasik. Dalam kriptografi modern dikenal 2 model kriptografi yaitu kriptografi Simetrik dan kriptografi Asimetrik..

Rumusan Permasalahan

Permasalahan pada proyek akhir ini adalah bagaimana membangun sistem aplikasi dengan menerapkan enkripsi Blowfish untuk mengamankan pengiriman surat secara elektronik sehingga orang yang tidak berkepentingan tidak bisa melihat, mengakses dan mengganti content yang ada dalam surat elektronik tersebut (Dalam kasus ini adalah pengiriman Email).

Untuk itu dari permasalahan tersebut akan dibahas hal-hal penting berikut :

- Bagaimana kita bisa melakukan sinkronisasi antara mail Server dengan mail Client ketika kita ingin melakukan pengambilan email dari server.
- Bagaimana kita bisa melakukan pengamanan terhadap data yang ada dalam email dengan menggunakan algoritma Blowfish sehingga email tersebut aman ketika dikirim melalui jaringan yang tidak aman.
- Bagaimana membangun kunci dari algoritma Blowfish karena dalam algoritma tersebut terdapat dua proses yaitu Pembentukan Kunci dan Enkripsi Data.

Tujuan

Tujuan dari pembuatan proyek akhir ini adalah membuat suatu aplikasi yang bisa melakukan penyandian (*Enkrip*si dan *Dekripsi*) terhadap sebuah dokumen elektronik (pada kasus ini adalah Email) sehingga ketika dikirimkan melalui jaringan yang tidak aman isi dari email tersebut tidak bisa dimengerti oleh orang lain kecuali orang yang mempunyai aplikasi ini dan mengetahui kunci yang digunakan untuk mengenkripsi email tersebut

2. TINJAUAN PUSTAKA

2.1 Dasar Teori Cipher Blok

Cipher blok merupakan algoritma kriptografi yang beroperasi dalam bentuk blok bit. Proses enkripsi dilakukan terhadap blok bit plainteks dengan mengguanakan kunci yang berukuran sama dengan ukuran blok plainteks. Algoritma ini akan menghasilkan cipherteks yang sama panjang dengan blok plainteks. Proses dekripsi terhadap cipherteks berlangsung dengan cara serupa seperti enkripsi. Hanya saja pada proses dekripsi, operasi berjalan kebalikan dari proses enkripsi. Proses enkripsi dengan kunci *K* dinyatakan secara formal dengan persamaan

$$E_K(P) = C$$

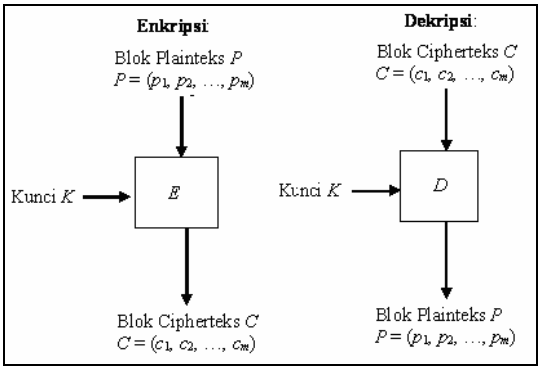
Sedangkan persamaan yang menyatakan proses dekripsi dengan kunci *K* adalah

$$D_K(C) = P$$

Fungsi *E* yang digunakan dalam proses enkripsi harus merupakan fungsi yang berkoresponden satu-ke-satu, sehingga

$$E^{-1} = D$$

Skema enkripsi dan dekripsi cipher blok dapat dilihat pada Gambar.



Gambar 1. Ilustrasi Enkripsi dan Dekripsi

Pada cipher blok plainteks dibagi menjadi beberapa blok dengan panjang tetap yaitu sepanjang kunci yang dimasukkan. Karena itulah terdapat kemungkinan panjang plainteks tidak habis dibagi dengan panjang ukuran blok yang ditetapkan atau panjang kunci. Hal ini mengakibatkan blok terakhir berukuran lebih pendek daripada ukuran blok-blok lainnya. Solusi permasalahan ini adalah dengan padding, yaitu dengan menambahkan blok terakhir dengan pola bit yang teratur hingga panjang blok sama dengan ukuran blok yang ditetapkan. Pola bit teratur ini misalnya ditambahkan bit 0 semua, atau bit 1 semua, atau bit 0 dan 1 secara bergantian. Setelah proses dekripsi hapus kembali padding, agar hasil dekripsi kembali menjadi plainteks.

Pada algoritma kriptografi yang beroperasi pada mode blok ini dikenal beberapa mode operasi. Empat mode operasi yang lazim digunakan pada cipher blok adalah:

1. *Electronic Code Book* (ECB)
2. *Cipher Block Chaining* (CBC)
3. *Cipher Feedback* (CFB)
4. *Output Feedback* (OFB)

Penggunaan mode-mode operasi tersebut tidak merubah fungsi enkripsi dan dekripsi yang telah didefinisikan.

2.1.1 Electronic Code Book (ECB)

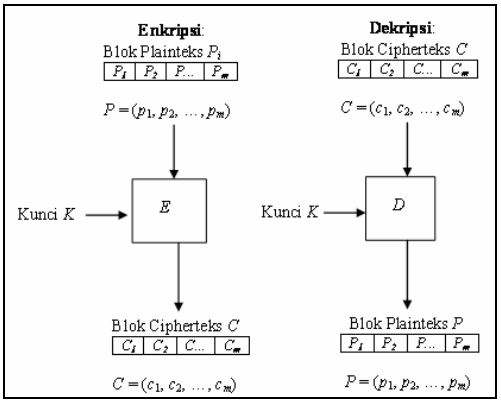
Pada mode ECB, setiap mode plainteks Pi dienkripsi secara individual dan independen menjadi blok cipherteks Ci. Secara matematis proses enkripsi dengan mode ECB dapat dinyatakan sebagai berikut:

$$Ci = E_K(Pi)$$

dan proses dekripsi sebagai berikut:

$$Pi = D_K(Ci)$$

Dalam hal ini, Pi dan Ci merupakan blok plainteks dan cipherteks ke-i. Skema enkripsi dan dekripsi dengan mode ECB dapat dilihat pada Gambar.



Gambar 2. Ilustrasi Enkripsi dan Dekripsi ECB

Mode ini merupakan mode termudah dari keempat mode yang telah disebutkan di atas. Setiap blok plainteks dienkripsi secara independen, sehingga proses enkripsi tidak harus berlangsung secara linear atau proses enkripsi dapat dilakukan pada blok-blok secara tidak berurutan. Keuntungan mode ECB lainnya adalah kesalahan satu bit pada satu blok hanya akan mempengaruhi blok cipherteks yang berkoresponden pada proses dekripsi.

2.1.2 Cipher Block Chaining (CBC)

Pada mode CBC terdapat mekanisme umpan balik pada sebuah blok, yaitu blok plainteks current di-XOR-kan terlebih dahulu dengan dengan blok cipherteks hasil enkripsi sebelumnya. Selanjutnya hasil operasi XOR ini dimasukkan ke dalam fungsi enkripsi. Dengan demikian pada mode CBC, setiap blok cipherteks bergantung tidak hanya pada blok plainteksnya, tetapi juga pada seluruh blok plainteks sebelumnya. Dekripsi dilakukan dengan cara memasukkan blok cipherteks current ke dalam fungsi dekripsi, kemudian meng-XOR-kan hasilnya dengan blok cipherteks sebelumnya. Secara matematis proses enkripsi dapat dinyatakan sebagai berikut:

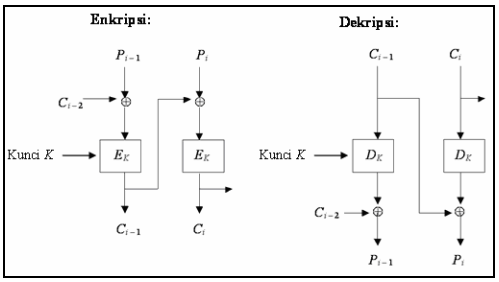
$$C_i = E_K(P_i \text{ Xor } C_{i-1})$$

sedangkan proses dekripsi dapat dinyatakan sebagai berikut:

$$P_i = D_K(C_i) \text{ Xor } C_{i-1}$$

Dalam hal ini C0 merupakan IV (Initialization Vactor). IV dapat diberikan oleh pengguna atau dibangkitkan secara acak oleh aplikasi. IV ini merupakan rangkaian bit yang tidak bermakna dan hanya digunakan sebagai inisialisasi untuk membuat setiap blok cipherteks menjadi unik. Gambar 3 memperlihatkan skema enkripsi dan dekripsi dengan mode CBC.

Dengan mode CBC, kesalahan pada satu bit plainteks akan mempengaruhi blok cipherteks yang berkoresponden dan blok-blok cipherteks selanjutnya. Sedangkan kesalahan satu bit pada cipherteks hanya akan mempengaruhi satu blok plainteks yang berkoresponden dan satu bit pada blok berikutnya dengan posisi bit yang berkoresponden pula.



Gambar 3. Ilustrasi Enkripsi dan DekripsiCBC

2.1.3 Cipher Feedback (CFB)

Mode CBC memiliki kelemahan yaitu proses enkripsi hanya dapat dilakukan pada ukuran blok yang utuh sehingga mode CBC tidak efisien jika diterapkan pada aplikasi komunikasi data. Permasalahan ini dapat diatasi pada mode CFB. Mode CFB mengenkripsikan data dalam unit yang lebih kecil daripada ukuran blok. Proses enkripsi pada unit yang lebih kecil daripada ukuran blok ini membuat mode CFB berlaku seperti cipher aliran. Karena hal inilah, mode CFB dapat diterapkan pada aplikasi komunikasi data.Unit yang dienkripsi dapat berupa bit per bit. Bila unit yang dienkripsi berupa satu karakter setiap kalinya, maka mode CFB ini disebut CFB 8-bit. Mode ini membutuhkan sebuah antrian yang berukuran sama dengan ukuran blok asukan. Secara formal, proses enkripsi mode CFB n-bit dapat dinyatakan sebagai berikut:

$$C_i = P_i \text{ Xor } MSB_m(E_K(X_i))$$
$$X_{i+1} = LSB_{m-n}(X_i) \,||\, C_i$$

sedangkan proses dekripsi dapat dinyatakan sebagai berikut:

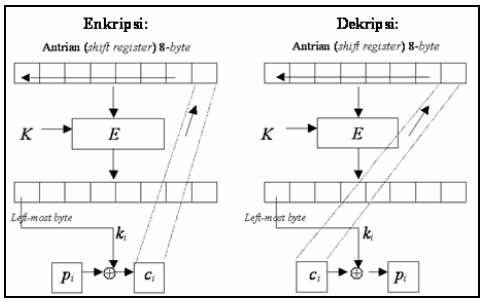
$$P_i = C_i \text{ Xor } MSB_m(D_K(X_i))$$
$$X_{i+1} = LSB_{m-n}(X_i) \,||\, C_i$$

Keterangan:

X_i = isi antrian dengan X_1 adalah IV
E = fungsi enkripsi
K = kunci

M = panjang blok enkripsi
N = panjang unit enkripsi
|| = operator penyambungan (*concatenation*)
MSB = *Most Significant Byte*
LSB = *Least Significant Byte*

Mode CFB mempunyai keunikan tersendiri, yaitu untuk proses enkripsi dan dekripsi digunakan fungsi yang sama. Skema enkripsi dan dekripsi dengan mode CFB 8-bit dapat dilihat pada Gambar



Gambar 4. Ilustrasi Enkripsi dan Dekripsi CFB

Pada mode CFB pun terdapat perambatan kesalahan baik pada proses enkripsi maupun proses dekripsi. Pada proses enkripsi, kesalahan satu bit pada plainteks mempengaruhi blok cipherteks yang berkoresponden dan blok-blok cipherteks berikutnya. Sedangkan kesalahan satu bit pada blok cipherteks akan bit yang berkoresponden dan bit-bit lainnya selama bit error tersebut terdapat di dalam shift register. Pada mode CFB 8-bit dan ukuran blok 64 bit, maka kesalahan satu byte pada blok cipherteks akan mempengaruhi satu byte plainteks yang berkoresponden dan delapan byte berikutnya (lama byte error terdapat dalam shift register adalah delapan putaran).

2.1.4 Output Feedback (OFB)

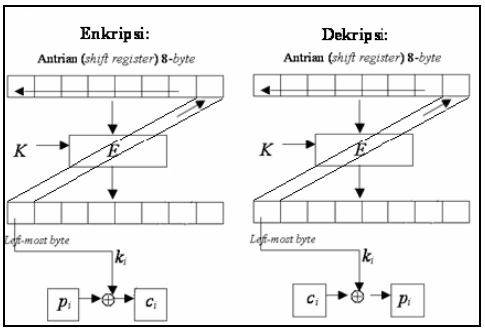
Mode OFB berkerja dengan cara yang mirip dengan mode CFB, kecuali n-bit dari hasil fungsi enkripsi terhadap antrian disalin menjadi elemen paling kanan antrian. Gambar 5 menunjukkan skema enkripsi dan dekripsi pada mode OFB 8-bit. Secara formal, proses enkripsi mode OFB n-bit dapat dinyatakan sebagai berikut:

$$C_i = P_i \; Xor MSB_m(E_K(X_i))$$
$$X_{i+1} = LSB_{m-n}(X_i) \;||\; MSB_m(E_K(X_i))$$

sedangkan proses dekripsi dapat dinyatakan sebagai berikut:

$$P_i = C_i \; Xor \; MSB_m(D_K(X_i))$$
$$X_{i+1} = LSB_{m-n}(X_i) \;||\; MSB_m(E_K(X_i))$$

Pada mode OFB tidak terdapat perambatan kesalahan. Kesalahan satu bit pada plainteks hanya mengakibatkan kesalahan satu bit yang berkoresponden pada cipherteks. Sebaliknya kesalahan satu bit pada cipherteks hanya mengakibatkan kesalahan satu bit yang berkoresponden pada plainteks.



Gambar 5. Ilustrasi Enkripsi dan Dekripsi OFB

2.2 Prinsip Perancangan *Cipher* Blok

Perancangan cipher blok harus memperhatikan beberapa prinsip perancangan yang telah biasa digunakan. Prinsip-prinsip yang dibahas pada upa-bab ini adalah prinsip cipher berulang (iterated cipher), jaringan Feistel (Feistel network), dan kotak-S (S-box).

2.2.1 Cipher Berulang (Iterated Cipher)

Cipher berulang merupakan fungsi transformasi sederhana yang mengubah plainteks menjadi cipherteks dengan proses perulangan sejumlah kali. Pada setiap putaran digunakan upa-kunci atau kunci putaran yang dikombinasikan dengan plainteks. Secara formal, cipher berulang dinyatakan sebagai berikut:

$$C_i = f(C_{i-1}, K_i)$$

Keterangan:
i = 1, 2, ..., *r* (*r* adalah jumlah putaran)
Ki = upa-kunci (subkey) pada putaran ke-i
f = fungsi transformasi (di dalamnya terdapat fungsi substitusi, permutasi, dan/atau ekspansi, kompresi)
Plainteks dinyatakan dengan C_0 dan cipherteks dinyatakan dengan C_r.

2.2.2 Jaringan Feistel (Feistel Network)

Hampir semua algoritma cipher blok bekerja dalam model jaringan Feistel. Jaringan ini ditemukan oleh Horst Feistel pada tahun 1970. Secara formal, operasi transformasi pada jaringan feistel dapat dinyatakan sebagai:

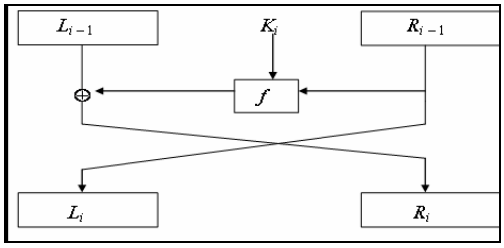
$$Li = Ri - 1$$
$$Ri = Li - 1 \text{ Xor } f(Ri - 1, Ki)$$

Proses enkripsi dan dekripsi dapat menggunakan model jaringan Feistel yang sama.

Model jaringan feistel bersifat reversible untuk proses enkripsi dan dekripsi. Sifat reversible ini memungkinkan mendekripsi cipherteks menjadi plainteks tanpa membuat algoritma baru. Contoh:

$$Li - 1 \text{ Xor } f(Ri - 1, Ki) \text{ Xor } f(Ri - 1, Ki) = Li - 1$$

Selain itu, sifat reversible tidak bergantung pada fungsi f sehingga fungsi f dapat dibuat serumit mungkin. Skema jaringan feistel dapat dilihat pada Gambar 6.



Gambar 6. Contoh Jaringan Feistel

2.2.3 Kotak-S (S-Box)

Kotak-S adalah suatu matriks sederhana yang berisi substitusi sederhana yang memetakan satu atau lebih bit dengan satu atau lebih bit yang lain. Pada kebanyakan algoritma cipher blok, kotak-S memetakan m bit masukan menjadi n bit keluaran, sehingga kotak-S tersebut dinamakan kotak m x n S-box.

Kotak-S merupakan satu-satunya langkah nirlanjad dalam algoritma, karena operasinya adalah look-up table. Masukan dari operasi look-up table dijadikan sebagai indeks kotak-S, dan keluarannya adalah entry di dalam kotak-S.

Contoh: Kotak-s di dalam algoritma DES adalah 6 x 4 S-box yang berarti memetakan 6 bit masukan menjadi 4 bit keluaran. Salah satu kotak-S yang terdapat di dalam algoritma DES dapat dilihat pada Tabel 1.

Tabel 1. Salah Satu Kotak S-Box pada DES

12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

Baris diberi nomor dari 0 sampai 3.
Kolom diberi nomor dari 0 sampai 15.
Masukan untuk proses substitusi adalah 6 bit, yaitu $b_1b_2b_3b_4b_5b_6$

Nomor baris dari tabel ditunjukkan oleh string bit b_1b_6 (menyatakan 0 sampai 3 desimal). Sedangkan nomor kolom ditunjukkan oleh string bit $b_2b_3b_4b_5$ (menyatakan 0 sampai 15 desimal).

Misalnya masukan adalah 110100
Nomor baris tabel = 10 (artinya baris 2 desimal)
Nomor kolom tabel = 1010 (artinya kolom 10 desimal)
Jadi, substitusi untuk 110100 adalah *entry* pada baris 2 dan kolom 10, yaitu 0100 (atau 4 desimal).

Perancangan kotak-s menjadi isu penting karena kotak-S harus dirancang sedemikian sehingga kekuatan kriptografinya bagus dan mudah diimplementasikan. Ada empat cara atau pendekatan yang dapat digunakan dalam pembuatan kotak-S. Keempat cara itu adalah:

1. Dipilih secara acak.
   Untuk kotak-S yang kecil, cara pengisian secara acak tidak aman, namun untuk kotak-s yang besar cara pemilihan acak ini cukup bagus untuk diterapkan.
2. Dipilih secara acak lalu diuji.
   Sama seperti cara nomor 1, namun nilai acak yang dibangkitkan diuji apakah memenuhi sifat tertentu.
3. Dibuat oleh orang (man made).
   Entry di dalam kotak-S dibangkitkan dengan teknik yang lebih intuitif. Perancang kotak-S mengisi kotak-S secara intuitif.
4. Dihitung secara matematis (math made).
   Entry di dalam kotak-S dibangkitkan berdasarkan prinsip matematika yang terbukti aman dari serangan kriptanalisis.

3. METODOLOGI

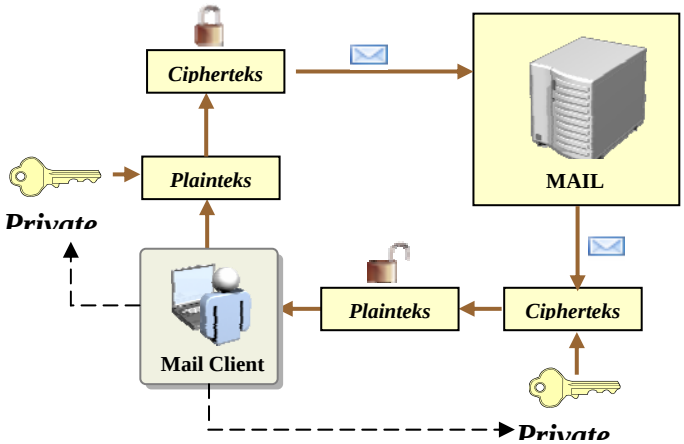
Dalam melakukan penyusunan program ini dilakukan dengan penelitian dan pecobaan secara nyata dengan mail client sehingga diharapkan nanti bisa diaplikasikan dengan kondisi lingkungan yang sesungguhnya.

4. PERANCANGAN SISTEM

4.1 Perancangan Data

Dalam proyek akhir ini dibuat sebuah aplikasi yang digunakan untuk melakukan enkripsi terhadap email yang nantinya email tersebut akan mengirimkan sebuah *ciphertext* sehingga akan aman jika di kirimkan melalui internet

4.2 Perancangan Proses



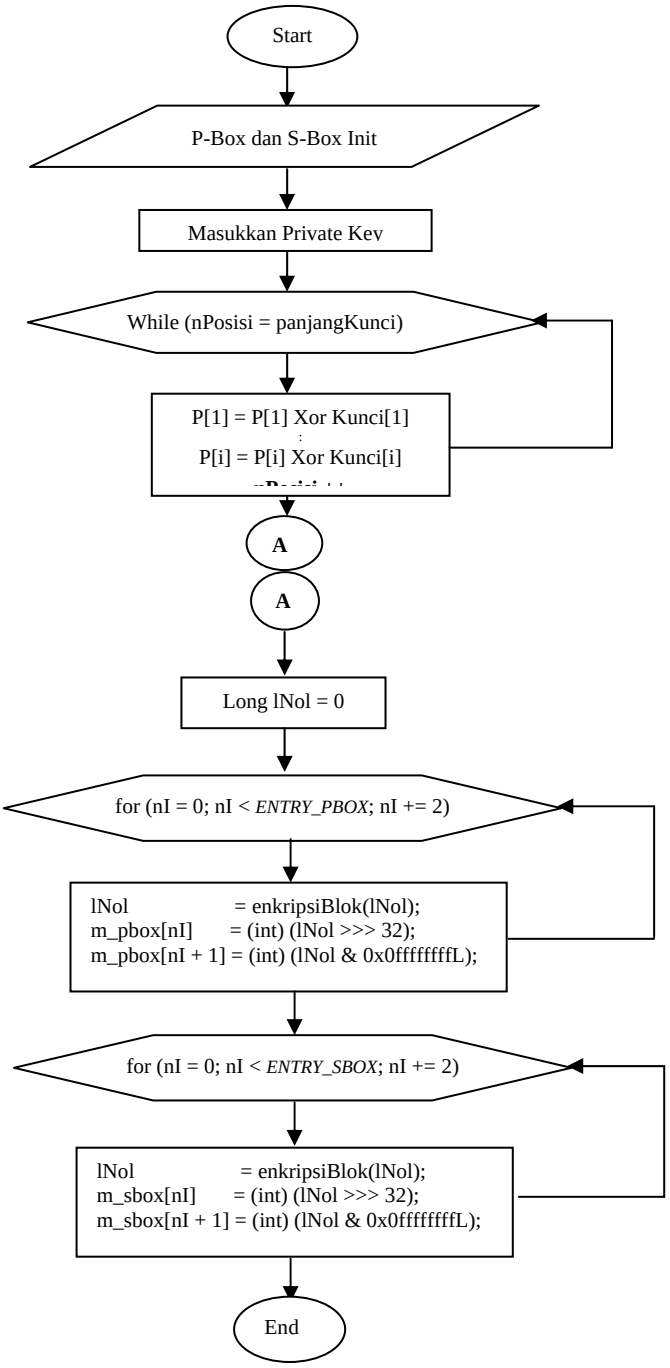
Gamba 7. Bagan Pengiriman dan Penerimaan Email

4.3 Proses Penerapan Algoritma Blowfish

Blowfish adalah algoritma kunci simetri, yang berarti menggunakan kunci yang sama untuk melakukan enkripsi dan dekripsi file. Blowfish juga merupa kan *cipher* blok, yang berarti selama proses enkripsi dan dekripsi, Blowfish akan membagi pesan menjadi blok-blok dengan ukuran yang sama panjang. Panjang blok untuk algoritma Blowfish adalah 64-bit. Pesan yang bukan merupakan kelipatan delapan *byte* akan ditambahkan bit-bit tambahan (*padding*) sehingga ukuran untuk tiap blok sama. Algoritma dalam Blowfish terbagi menjadi dua bagian, yaitu *key expansion* dan *data encryption*.

Proses *Key Expansion*

Pada tahap ini akan dijelaskan tentang bagaimana melakukan *key expansion* yang akan digunakan pada proses enkripsi dan dekripsi.



Gambar 8. Flowchart Pembangkitan Kunci Blowfish

1. Pertama-tama inisialisasi P-array dan kemudian keempat kotak-S, secara berurutan dengan sebuah string yang sama. String yang digunakan ini terdiri dari digit heksadesimal dari p.

```
m_pbox = new int[ENTRY_PBOX];
// Lakukan looping untuk memasukkan nilai yang ada dalam array inisialisasi
for (nI = 0; nI < ENTRY_PBOX; nI++) {
    m_pbox[nI] = pbox_init[nI];
}

m_sbox1 = new int[ENTRY_SBOX];
m_sbox2 = new int[ENTRY_SBOX];
m_sbox3 = new int[ENTRY_SBOX];
m_sbox4 = new int[ENTRY_SBOX];

// Lakukan looping untuk memasukkan nilai yang ada dalam array inisialisasi
for (nI = 0; nI < ENTRY_SBOX; nI++) {
      m_sbox1[nI] = inisialisasi_sbox_1[nI];
      m_sbox2[nI] = inisialisasi_sbox_2[nI];
      m_sbox3[nI] = inisialisasi_sbox_3[nI];
      m_sbox4[nI] = inisialisasi_sbox_4[nI];
}
```

2. Lakukan operasi XOR antara P1 dengan 32 bit pertama dari kunci, lakukan operasi XOR antara P2 dengan 32 bit kunci berikutnya, dan begitu seterusnya untuk semua bit pada kunci (hingga P18). Ulangi langkah ini melalui perputaran bit-bit kunci hingga seluruh elemen pada *P-array* telah dilakukan operasi XOR dengan bit-bit kunci.

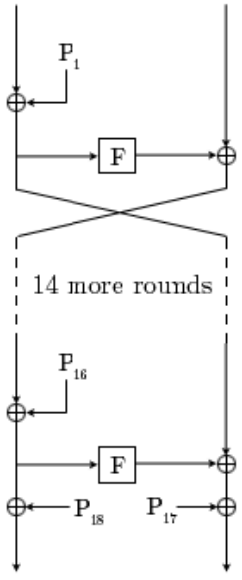
```
for (nI = 0; nI < ENTRY_PBOX; nI++) {
      for (nJ = 0; nJ < 4; nJ++) {

            // Keterangan : 0x0ff nilai desimalnya 255 dan binernya 32 bit
             nJalan = (nJalan << 8) | (((int) kunci[nPosisiKunci]) & 0x0ff);

             if (++nPosisiKunci == panjangKunci) {
                  nPosisiKunci = 0;
             }
      }
      m_pbox[nI] ^= nJalan;
}
```

3. Proses enkripsi terhadap string dengan keseluruhan elemen nol dengan algoritma Blowfish, menggunakan upa-kunci yang dijelaskan pada langkah di atas.
4. Ubah nilai P1 dan P2 dengan hasil keluaran pada langkah 3.
5. Lakukan proses enkripsi terhadap hasil keluaran dari langkah 3 menggunakan algoritma Blowfish dengan upa-kunci yang telah dimodifikasi.
6. Ubah nilai P3 dan P4 dengan hasil keluaran pada langkah 5.
7. Lanjutkan proses mengubah semua elemen yang terdapat pada P-array, dan kemudian keempat kotak-S secara berurutan, dengan hasil keluaran algoritma Blowfish yang terus menerus berubah.

```
// Inisialisasi nilai lNol dengan nol;
  long lNol = 0;

// (sama seperti contoh, dibutuhkan 512 iterasi untuk membangkitkan seluruh sub-key yang dibutuhkan)

for (nI = 0; nI < ENTRY_PBOX; nI += 2) {
      lNol = enkripsiBlok(lNol);
      m_pbox[nI] = (int) (lNol >>> 32);
      m_pbox[nI + 1] = (int) (lNol & 0x0ffffffffL);
}

for (nI = 0; nI < ENTRY_SBOX; nI += 2) {
      lNol = enkripsiBlok(lNol);
       m_sbox1[nI] = (int) (lNol >>> 32);
       m_sbox1[nI + 1] = (int) (lNol & 0x0ffffffffL);
}
                .
                .
                .
                .
                .
for (nI = 0; nI < ENTRY_SBOX; nI += 2) {
      lNol = enkripsiBlok(lNol);
      m_sbox4[nI] = (int) (lNol >>> 32);
      m_sbox4[nI + 1] = (int) (lNol & 0x0ffffffffL);
}
```

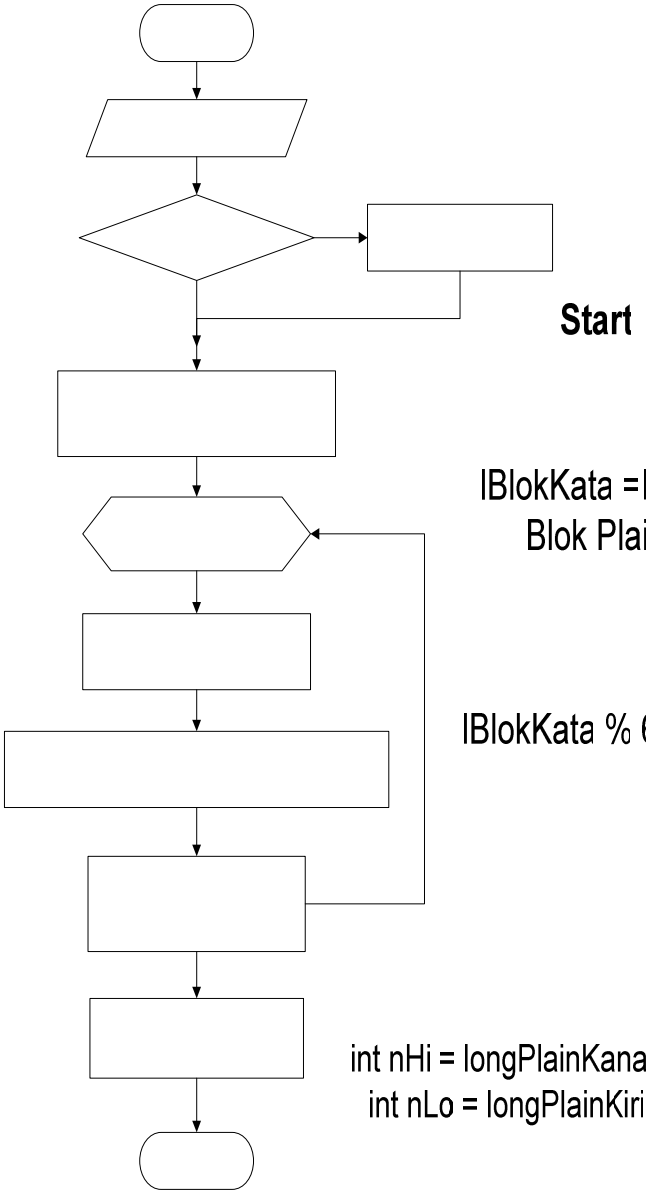
Proses *Data Encryption*

- **Proses Enkripsi :**
  Proses enkripsi dilakukan dengan langkah-langkah sebagai berikut:



Gambar 9. Skema Jaringan Feistel Blowfish

Ketika kita melihat bagan di atas bisa diketahui bahwa masukan dalam bentuk Long (64 Bit) dipecah menjadi 2 bagian Left dan Right masing – masing 32 bit.



Gambar 10. Flowchart Proses enkripsi

```
protected long enkripsiBlok(long lBlokKata) {
    //  Pecah 64 bit menjadi masing-masing 32 bit

    int nHi = conv.longPlainKanan32(lBlokKata);
    int nLo = conv.longPlainKiri32(lBlokKata);
    // System.out.println("nHi : "+nHi);
    // System.out.println("nLo : "+nLo);
```

2. Lakukan langkah-langkah berikut dalam 16 putaran:

   $xL = xL \text{ Xor } Pi$

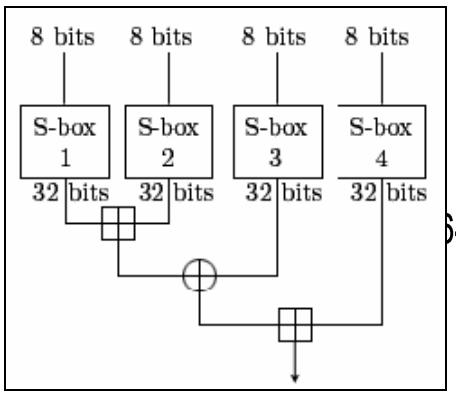
   $xR = F(xL) \text{ Xor } xR$

   Tukar xL dan xR

   Keterangan:

   i = 1, 2, ..., 16 (menunjukkan nomor putaran)

$$F(xL) = ((S_{1,a} + S_{2,b} \bmod 2^{32}) \text{ Xor } S_{3,c}) + S_{4,d} \bmod 2^{32}$$



Gambar 11. Fungsi F Blowfish

3. Tukar xL dan xR (membatalkan pertukaran terakhir).
4. xR = xR Xor P17
5. xL = xL Xor P18
6. Kembalikan kembali xL dan xR

```
// Fungsi untuk melakukan proses Enkripsi Blowfish  64 bit

public void encrypt(byte[] bufferMasuk, byte[] bufferKeluar) {
    int panjangKunci = bufferMasuk.length;
    long lTemp;
    for (int nI = 0; nI < panjangKunci; nI += 8) {
        // Enkrip buffer sementara menggunakan 64 bit blok
        lTemp = conv.byteArrayToLong(bufferMasuk, nI);
        lTemp = enkripsiBlok(lTemp);
        conv.longToByteArray(lTemp, bufferKeluar, nI);
    }
}
```

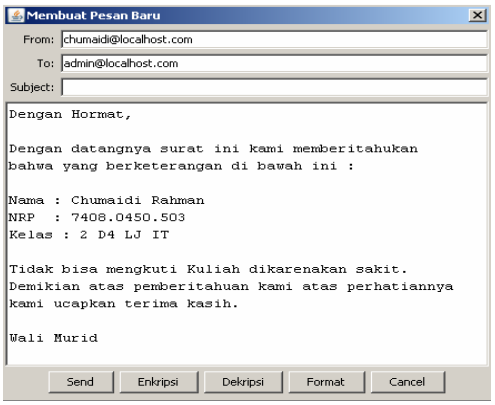
```
    int[] sbox1 = m_sbox1;
    int[] sbox2 = m_sbox2;
    int[] sbox3 = m_sbox3;
    int[] sbox4 = m_sbox4;

    int[] pbox = m_pbox;

    nHi ^= pbox[0];
    nLo ^= (((sbox1[nHi >>> 24] + sbox2[(nHi >>> 16) &
    0x0ff]) ^ sbox3[(nHi >>> 8) & 0x0ff]) + sbox4[nHi &
    0x0ff]) ^ pbox[1];
    nHi ^= (((sbox1[nLo >>> 24] + sbox2[(nLo >>> 16) &
    0x0ff]) ^ sbox3[(nLo >>> 8) & 0x0ff]) + sbox4[nLo &
    0x0ff]) ^ pbox[2];
        :
        :
        :
    nLo ^= (((sbox1[nHi >>> 24] + sbox2[(nHi >>> 16) &
    0x0ff]) ^ sbox3[(nHi >>> 8) & 0x0ff]) + sbox4[nHi &
    0x0ff]) ^ pbox[15];
    nHi ^= (((sbox1[nLo >>> 24] + sbox2[(nLo >>> 16) &
    0x0ff]) ^ sbox3[(nLo >>> 8) & 0x0ff]) + sbox4[nLo &
    0x0ff]) ^ pbox[16];
    // Akhirnya, silangkan dan kembalikan sesuai awal
    return conv.makeLong(nHi, nLo ^ pbox[17]);
}
```

1. Bagi lTemp menjadi setengah bagian, yaitu dengan ukuran 32 bit. Hasil pembagian ini adalah : xL dan xR.

$$F(XL) = ((S_{1,a} + S_{2,b} \bmod 2^{32}) \text{ Xor } S_{3,c}) + S_{4,d} \bmod 2$$

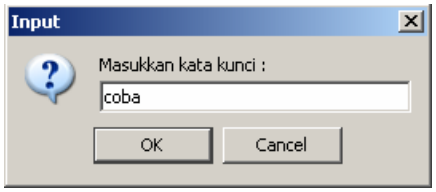
4.4 Proses Testing Mail Client

Tulisan dalam Form di bawah masih dalam bentuk plainteks, untuk mengirimkan email tersebut dalam betuk enkripsi maka klik tombol enkripsi.



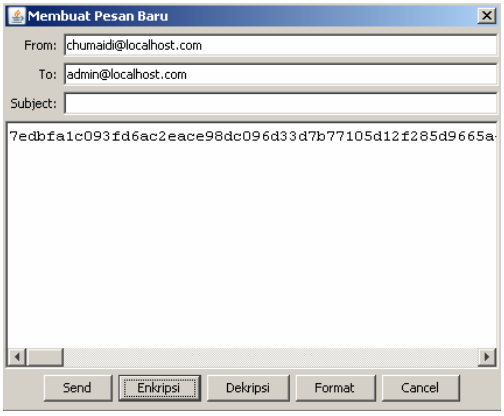
Gambar 12. Form Pengiriman Email

Saat tombol enkripsi ditekan maka akan muncul sebuah input box yang berisi kata ”Masukkan Kata Kunci”. Kata kunci ini digunakan untuk melakukan enkripsi dengan plainteks yang kita masukkan dan kata kunci ini juga yang akan digunakan untuk melakukan dekripsi cipherteks yang dihasilkan oleh kombinasi kata kunci dan plainteks. Kata kunci ini tidak boleh sampai lupa karena jika kata kunci yang kita masukkan lupa maka kita tidak akan bisa membuka email ini.



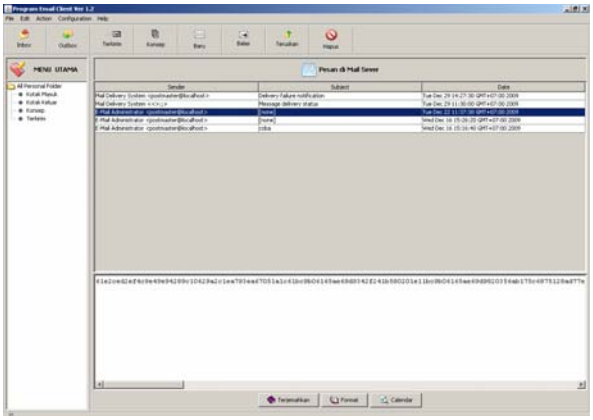
Gambar 13. Form Input Kata Kunci

Setelah kita masukkan kata kunci maka plainteks tadi akan berubah menjadi sebuah cipherteks (Dalam bentuk Hexadecimal). Cipherteks inilah yang nantinya akan kita kirimkan sehingga meskipun ada orang yang melihat isi dari email kita maka dia tidak akan bisa mengetahui apa maksud dan isi dari email kita, kecuali orang yang sudah mengetahui kata kuncinya dan memiliki aplikasi ini.



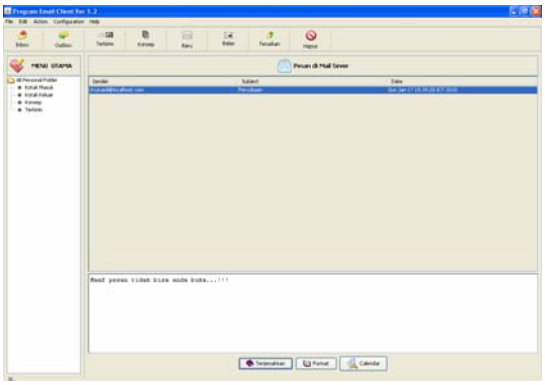
Gambar 14. Form Hasil Enkripsi Plainteks

Untuk membuka email tinggal klik tabel dan detailnya dari email yang masuk akan muncul di kotak detail email, di bawah juga terdapat beberapa tombol Kalender, Dekripsi, Format email.



Gambar 15. Form Menu Utama

Ketika kata kunci yang kita masukkan salah maka akan muncul pesan yang berisi “Maaf pesan tidak bisa anda buka…!!!”. Sebagai konsekuensi jika kita lupa dengan kata kuncinya maka pesan tidak akan bisa kita baca. Di bawah ini menunjukkan pesan yang tidak bisa dibuka karena kata kuncinya salah.



Gambar 15. Form Menu Utama

5. KESIMPULAN DAN SARAN

5.1 KESIMPULAN

Pada bagian ini akan diulas tentang kesimpulan dari seluruh percobaan dan pengujian dari software yang penulis buat :

1. Cepat. Blowfish dirancang agar dapat mengenkripsikan data pada mikroprosesor 32 bit dengan kecepatan 26 clock cycles per byte.
2. Kompak. Blowfish dirancang agar dapat berjalan dengan penggunaan memori kurang dari 5kB.
3. Flexibel. Untuk lebih memperkuat enkripsi dari Algoritma Blowfish lebih baik kita menggunakan kunci yang agak panjang, tetapi konsekuensinya waktu yang dibutuhkan untuk melakukan *Key Expansion* atau mengubah kunci menjadi sub-kunci menjadi lebih lama.
4. Sederhana. Blowfish dirancang hanya menggunakan operasi-operasi sederhana. Operasi-operasi yang digunakan dalam Blowfish adalah penambahan, XOR, dan table lookup dalam operand 32 bit. Rancangan yang sederhana ini mempermudah proses analisa yang membuat Blowfish terhindar dari kesalahan implementasi.

Aplikasi ini jauh dari kata sempurna maka jika ada kritik dan saran yang membangun kiranya penulis akan menerima dengan ihlas dan terima kasih.

5.2 SARAN – SARAN

Untuk pengembangan lebih lanjut, diharapkan dapat menggunakan metode lain yang lebih baik sehingga hasil pengenalan buah bisa lebih bagus lagi dan nilai kemiripan lebih tinggi.

DAFTAR PUSTAKA

1. [MUN06] Munir, Rinaldi. 2006. Diktat Kuliah IF5054 Kriptografi. Program Studi Teknik Informatika - Institut Teknologi Bandung.
2. [SCH93] Schneier, Bruce. 1993. Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish). Springer-Verlag.
3. [SCH95] Schneier, Bruce. 1995. The Blowfish Encryption Algorithm -- One Year Later. Dr. Dobb's Journal.
4. [SCH96] Schneier, Bruce. 1996. Applied Cryptography, Second Edition: Protocols, Algorithms, and Source Code in C. John Wiley and Sons.
5. [SCH99] Schneier, Bruce. 1999. Performance Comparison of the AES Submission.
6. [SCH01] Schneier, Bruce. 2001. The Twofish Encryption Algorithm - Block encryption for the 21st century. Dr. Dobb's Journal
7. Stallings Wiliam, Cryptography and Network Security, Prentice Hall, 2000