PENYANDIAN DATA DENGAN KRIPTOGRAFI *PASSWORD BASED ENCRYPTION* MENGGUNAKAN *MESSAGE DIGEST 5* DAN *DATA ENCRYPTION STANDART*

Edy Winarno
Fakultas Teknologi Informasi, Universitas Stikubank Semarang

Abstraksi

Password Based Encryption (PBE) adalah sebuah metode kriptografi simetrik yang menggunakan kunci seperti password dalam melakukan proses enkripsinya dan menggunakan kunci yang sama untuk melakukan proses dekripsinya sehingga akan dihasilkan data yang sama dengan data plaintex aslinya.

Data plaintex yang telah dienkripsi akan menghasilkan sebuah chipertex yang tidak dapat dibaca oleh orang lain. Chipertex inilah yang akan dikirimkan ke pihak kedua sehingga akan memiliki kerahasiaan yang bisa diandalkan. Data chipertex yang dihasilkan akan berubah-ubah sesuai masukan data kunci password yang diberikan.

Password Based Encryption dengan Message Digest (MD5) dan Data Encryption Standart (DES) merupakan metode kriptografi menggunakan algoritma yang menggabungkan antara metode hashing dan enkripsi standar. MD5 adalah algoritma message digest yang dikembangkan oleh Ronald Rivest dimana MD5 ini mengambil pesan dengan panjang sembarang dan menghasilkan message digest 128 bit. Pada MD5 pesan diproses dalam blok 512 bit dengan empat round berbeda.

DES bekerja dalam model bit , atau angka biner 0 dan 1. Setiap kelompok dari 4 bit membentuk hexadesimal, atau bilangan berbasis 16. Angka biner 0001 membentuk angka heksa 1, dan seterusnya. DES bekerja dengan mengenkripsi setiap group yang terdiri dari 64 bit data.

Kata kunci : Kriptografi, enkripsi, dekripsi, Password

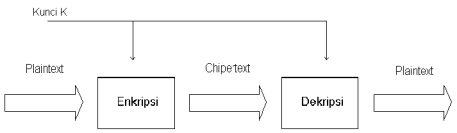
PENDAHULUAN

Penyandian data atau ilmu kriptografi adalah ilmu yang mempelajari tentang penyembunyian huruf atau tulisan sehingga membuat tulisan tersebut tidak dapat dibaca oleh orang yang tidak berkepentingan. Kriptografi sudah dipakai sejak jaman Julius Caesar dimana akan mengirimkan pesan kepada panglimanya tetapi tidak mempercayai kurir pembawa pesan tersebut. Kriptografi mempunyai 2 (dua) bagian yang penting, yaitu enkripsi dan dekripsi. Enkripsi adalah proses dari penyandian pesan asli menjadi pesan yang tidak dapat diartikan seperti aslinya.

Dekripsi sendiri berarti merubah pesan yang sudah disandikan menjadi pesan aslinya. Pesan asli biasanya disebut *plaintext*, sedangkan pesan yang sudah disandikan disebut *ciphertext*. Pada Gambar 1. dapat dilihat bahwa masukan berupa *plaintext* akan masuk ke dalam blok enkripsi dan keluarannya akan berupa *ciphertext*, kemudian *ciphertext* akan masuk ke dalam blok dekripsi dan keluarannya akan kembali menjadi *plaintext* semula.

Gambar 1. Proses Enkripsi dan Dekripsi

Ada 2 (dua) model algoritma enkripsi yang menggunakan kunci, yaitu kunci simetrik dan kunci asimetrik. Enkripsi kunci simetrik yang biasanya disebut enkripsi konvensional adalah enkripsi yang menggunakan kunci yang sama untuk enkripsi maupun dekripsi, dari Gambar 2. terlihat bahwa untuk mengenkripsi maupun mendekripsi pesan hanya menggunakan satu buah kunci (K) saja.



Gambar 2. Enkripsi-dekripsi Kunci Simetrik

Penggunaan metode ini membutuhkan persetujuan antara pengirim dan penerima tentang kunci sebelum mereka saling mengirim pesan. Keamanan dari kunci simetrik tergantung pada kerahasiaan kunci,

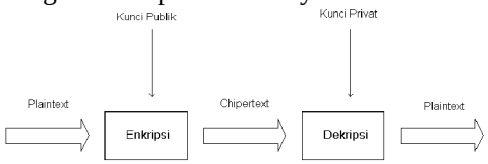
apabila seorang penyusup dapat menemukan kunci maka dengan mudah dapat membaca pesan yang sudah dienkripsi. Enkripsi kunci simetrik dapat dibagi kedalam 2 (dua) kelompok yaitu metode *stream cipher* dan metode *block cipher*.

Enkripsi kunci asimetrik (biasa disebut enkripsi kunci publik) dibuat sedemikian rupa sehingga kunci yang dipakai untuk enkripsi berbeda dengan kunci yang dipakai untuk dekripsi. Enkripsi kunci publik disebut demikian karena kunci untuk enkripsi boleh disebarluaskan kepada umum sedangkan kunci untuk mendekripsi hanya disimpan oleh orang yang bersangkutan. Enkripsi asimetrik dapat ditulis seperti berikut:

Ek (P) = C

Dk (C) = P

Contohnya seperti pada Gambar 3. bila seseorang ingin mengirim pesan kepada orang lain maka orang tersebut menggunakan kunci publik orang tersebut untuk mengenkripsi pesan yang kita kirim kepadanya lalu orang tersebut akan mendekripsi pesan tersebut dengan kunci privat miliknya.

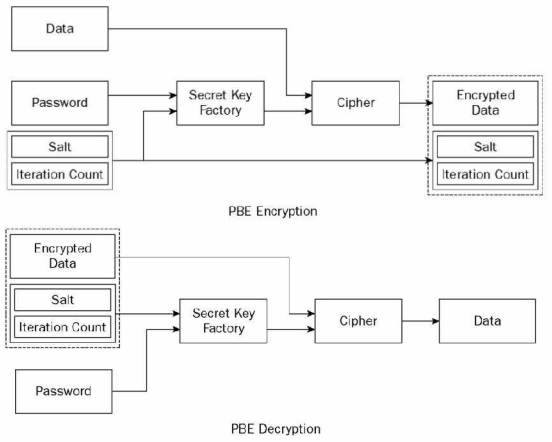


Gambar 3. Enkripsi Kunci Asimetrik

PASSWORD BASED ENCRYPTION

Password Based Encryption (PBE) adalah sebuah metode kriptografi simetrik yang menggunakan kunci seperti *password* dalam melakukan proses enkripsinya dan menggunakan kunci yang sama untuk melakukan proses dekripsinya sehingga akan dihasilkan data yang sama dengan data *plaintex* aslinya. Data *plaintex* yang telah dienkripsi akan menghasilkan sebuah *chipertex* yang tidak dapat dibaca oleh orang lain. *Chipertex* inilah yang akan dikirimkan ke pihak kedua sehingga akan memiliki kerahasiaan yang bisa diandalkan. Data *chipertex* yang dihasilkan akan berubah-ubah sesuai masukan data kunci *password* yang diberikan.

Kriptografi PBE dibuat berdasarkan mekanisme *hashing*. Sebuah *password* dan s*alt* akan dikombinasikan sehingga akan menghasilkan data yang random melalui proses fungsi aplikasi dan akan diolah oleh perhitungan iterasi (*iteration count*) sehingga ketika proses pencampuran telah selesai akan menghasilkan data berupa *chipertex*. Gambar 4. akan menunjukkan proses enkripsi menggunakan *Password Based Encryption* (PBE)



Gambar 4. Proses Enkripsi dan Dekripsi menggunakan PBE

Password, merupakan data yang selain harus dijaga kerahasiaannya juga merupakan data yang harus sulit ditebak oleh orang lain sehingga aplikasi yang dikerjakan akan menjadi sangat aman. Berapa lebar *bandwidth* yang didapatkan pada sebuah *password* dalam pemrograman java tergantung pada metode PBE yang digunakan. Pada umumnya penggunaan seperti PKC#5 hanya memperhitungkan jumlah karakter ASCII dan hanya didapatkan 8 bits pada masing-masing java karakter yang telah diproses ke dalam fungsi. Jika menggunakan metode PKC#12 bisa didapatkan sampai 12 bits *full* untuk masing- masing karakter java.

Salt, merupakan sebuah nilai publik dan dapat dengan mudah untuk ditemukan oleh orang lain. Salt digunakan untuk menambah sebuah *string* dari *byte-byte* yang random pada password, *password* yang sama dapat digunakan sebagai sebuah sumber untuk nomor yang besar dari kunci-kunci yang berbeda. *Salt* yang bagus adalah *salt* yang besarnya bisa menyamai ukuran blok dari fungsi yang digunakan pada untuk memproses *password*.

Iteration Count, adalah juga merupakan sebuah nilai publik. Fungsi dari *iteratin count* adalah untuk menambah perhitungan waktu yang dibutuhkan untuk mengkonversi sebuah *password* menjadi sebuah kunci.

Beberapa algoritma yang digunakan dalam *Password Based Encryption adalah:*

1. PBEWithMD5AndDES
2. PBEWithSHA1AndDES
3. PBEWithSHA1AndRC2
4. PBEWithMD5AndRC2
5. PBEWithSHA1AndIDEA
6. PBEWithSHA1And3-KeyTripleDES
7. PBEWithSHA1And2-KeyTripleDES
8. PBEWithSHA1And4OBitRC2
9. PBEWithSHA1And4OBitRC4
10. PBEWithSHA1And128BitRC2
11. PBEWithSHA1And128BitRC4
12. PBEWithSHA1AndTwofish

PBE dengan MD5 dan DES

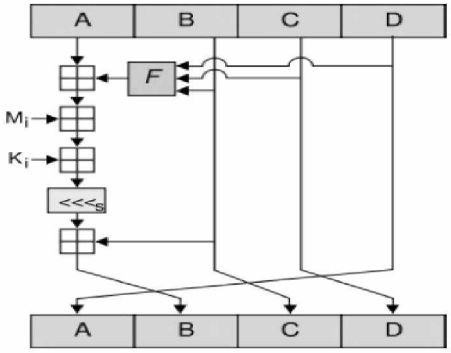
PBE dengan MD5 dan DES merupakan metode kriptografi menggunakan algoritma *Message Digest* 5 (MD5) dan *Data Encryption Standard* (DES). MD5 adalah algoritma *message digest* yang dikembangkan oleh Ronald Rivest pada tahun 1991. MD5 mengambil pesan dengan panjang sembarang dan menghasilkan *message digest* 128 bit. Pada MD5 pesan diproses dalam blok 512 bit dengan empat *round* berbeda.

DES, akronim dari *Data Encryption Standard*, adalah nama dari *Federal Information Processing* Standard (FIPS) 46-3, yang menggambarkan *data encryption algorithm* (DEA). DEA juga didefinisikan dalam ANSI *standard* X3.92. DEA merupakan perbaikan dari algoritma Lucifer yang dikembangkan oleh IBM pada awal tahun 7Oan. Meskipun algoritmanya pada intinya dirancang oleh IBM, NSA dan NBS (sekarang NIST (*National Institute of Standards and Technology*)) memainkan peranan penting pada tahap akhir pengembangan. DEA, sering disebut DES, telah dipelajari secara ekstensif sejak publikasinya dan merupakan algoritma simetris yang paling dikenal dan paling banyak digunakan.

DES memiliki ukuran blok 64-bit dan menggunakan kunci 56-bit kunci selama eksekusi (8 bit paritas dihilangkan dari kunci 64 bit). Saat digunakan untuk komunikasi, baik pengirim maupun penerima harus mengetahui kunci rahasia yang sama, yang dapat digunakan untuk mengenkrip dan mendekrip pesan, atau untuk proses *generate* dan verifikasi *message authentication code* (MAC).

MD5 (Message Digest 5)

Message Digest 5 (MD-5) adalah salah satu penggunaan fungsi *hash* satu arah yang paling banyak digunakan. MD-5 merupakan fungsi *hash* kelima yang dirancang oleh Ron Rivest. MD-5 merupakan pengembangan dari MD-4 dimana terjadi penambahan satu ronde. MD-5 memproses teks masukan ke dalam blok- blok bit sebanyak 512 bit, kemudian dibagi ke dalam 32 bit sub blok sebanyak 16 buah. Keluaran dari MD-5 berupa 4 buah blok yang masing-masing 32 bit yang mana akan menjadi 128 bit yang biasa disebut nilai *hash*. Pada Gambar 5. terlihat simpul utama dari MD-5. Simpul utama MD5 mempunyai blok pesan dengan panjang 512 bit yang masuk ke dalam 4 buah ronde. Hasil keluaran dari MD-5 adalah berupa 128 bit dari *byte* terendah A dan tertinggi *byte* D.



Gambar 5. Operasi MD5

Gambar 5. merupakan operasi dari sebuah MD5. MD5 terdiri atas 64 operasi, dikelompokkan dalam empat putaran dari 16 operasi. *F* adalah fungsi nonlinear, satu fungsi digunakan pada tiap-tiap putaran. *Mi* menunjukkan blok 32- bit dari masukan pesan, dan *Ki* menunjukkan konstanta 32-bit, berbeda untuk tiap- tiap operasi. <<<s menunjukkan perputaran bit kiri oleh s, s bervariasi untuk tiap- tiap operasi menunjukkan tambahan modulo 2^{32}.

Setiap pesan yang akan dienkripsi, terlebih dahulu dicari berapa banyak bit yang terdapat pada pesan. Kita anggap sebanyak b bit. Di sini b adalah bit *non negatif integer*, b bisa saja nol dan tidak harus selalu kelipatan delapan. Pesan dengan panjang b bit dapat digambarkan seperti berikut : **m_0 m_1 …..m_(b-1)**

Terdapat 5 langkah yang dibutuhkan untuk untuk menghitung intisari pesan. Adapun langkah–langkah tersebut adalah :

1. Menambahkan bit

Pesan akan ditambahkan bit-bit tambahan sehingga panjang bit akan kongruen dengan 448, mod 512. Hal ini berarti pesan akan mempunyai panjang yang hanya kurang 64 bit dari kelipatan 512 bit. Penambahan bit selalu dilakukan walaupun panjang dari pesan sudah kongruen dengan 448, mod 512 bit. Penambahan bit dilakukan dengan menambahkan "1"di awal dan diikuti "0" sebanyak yang diperlukan sehingga panjang pesan akan kongruen dengan 448, mod 512.

2. Penambahan Panjang Pesan

Setelah penambahan bit, pesan masih membutuhkan 64 bit agar kongruen dengan kelipatan 512 bit. 64 bit tersebut merupakan perwakilan dari b (panjang pesan sebelum penambahan bit dilakukan). Bit-bit ini ditambahkan ke dalam dua word (32 bit) dan ditambahkan dengan *low-order* terlebih dahulu. Penambahan pesan ini biasa disebut juga ***MD Strengthening*** atau **Penguatan MD.**

3. Inisialisasi MD-5

Pada MD-5 terdapat empat buah *word* 32 *bit register* yang berguna untuk menginisialisasi *message digest* pertama kali. Register-register ini diinisialisasikan dengan bilangan *hexadesimal*.

word A: 01 23 45 67
word B: 89 AB CD EF
word C: FE DC BA 98
word D: 76 54 32 10

Register-register ini biasa disebut dengan nama *Chain variabel* atau variabel rantai.

4. Proses Pesan di dalam Blok 16 Word

Pada MD-5 juga terdapat 4 (empat) buah fungsi *nonlinear* yang masing masing digunakan pada tiap operasinya (satu fungsi untuk satu blok), yaitu:

F(x,Y,z) = (x ρ∀Y) ρ∀((ρ∀x) ρ∀z)

G(x,Y,z) = (x ρ∀z) ρ∀(Y ρ∀(ρ∀z))

H(x,Y,z) = x ρ∀Y ρ∀z

I (x,Y,z) = Y ρ∀(x ρ∀(ρ∀z))

(ρ∀untuk xOR, ρ∀untuk AND, ρ∀untuk OR dan ρ∀untuk NOT).

Berikut dapat dilihat satu buah operasi dari MD-5 dengan operasi yang dipakai sebagai contoh adalah FF(a,b,c,d,Mj,s,ti) menunjukan a = b + ((a + F(b,c,d) + Mj + ti) <<<s).

Bila Mj menggambarkan pesan ke-j dari sub blok (dari 0 sampai 15) dan <<<s menggambarkan bit akan digeser ke kiri sebanyak s bit, maka keempat operasi dari masing-masing ronde adalah:

FF(a,b,c,d,Mj,s,ti) menunjukan a = b + ((a + F(b,c,d) + Mj + ti) <<< s)

GG(a,b,c,d,Mj,s,ti) menunjukan a = b + ((a + G(b,c,d) + Mj + ti) <<< s)

HH(a,b,c,d,Mj,s,ti) menunjukan a = b + ((a + H(b,c,d) + Mj + ti) <<< s)

II(a,b,c,d,Mj,s,ti) menunjukan a = b + ((a + I(b,c,d) + Mj + ti) <<< s)

Konstanta ti didapat dari integer 232. abs(sin(i)), dimana i dalam radian.

5. Keluaran MD-5

Keluaran dari MD-5 adalah 128 bit dari *word* terendah A dan tertinggi *word* D masing-masing 32 bit.

DES (Data Encryption Standard)

Skema enkripsi yang paling umum digunakan saat ini adalah *Data encryption Standard* (DES). DES diadopsi pada tahun 1977 oleh *National Bureau of Standards*, atau sekarang disebut sebagai *National Institute* of *Standards and Technology* (NIST), sebagai *Federal Information Processing Standard* 46 (FIPS PUB 46). Di dalam DES, data dienkripsi di dalam 64 bit blok dengan menggunakan 56 bit kunci. Algoritma DES mengubah 64 bit input di dalam berbagai langkah menjadi 64 bit output. Langkah yang sama, dengan kunci yang sama, digunakan untuk mendekripsi *ciphertext* yang dihasilkan.

Pada tahun 1960, IBM memulai sebuah proyek di dalam kriptografi komputer yang dipimpin oleh Horst Feistel. Proyek ini selesai pada tahu 1971 dengan pengembangan algoritma yang disebut sebagai LUCIFER [FEIS73], yang dijual kepada Lloyd's of London untuk digunakan pada sistem penyaluran uang tunai, yang juga dikembangkan oleh IBM. LUCIFER adalah blok *cipher* yang beroperasi pada 64 bit blok, dengan menggunakan ukuran kunci 128 bit. Karena hasil yang menjanjikan, IBM kemudian mengembangkan sistem ini secara komersial. Usaha ini dipimpin oleh Walter tuchman dan Carl Meyer, dan tidak

hanya melibatkan IBM saja, tetapi juga konsultan luar dan nasehat yang bersifat teknikal dari NSA. Hasilnya, muncul LUCIFER versi baru yang lebih tahan terhadap *cryptoanalyst* tetapi dengan mengurangi ukuran kunci menjadi 56 bit sehingga dapat diimplementasikan pada sistem dengan prosesor tunggal.

Sementara itu, National Bureau of Standards (NBS) pada tahun 1973 mengeluarkan permintaan untuk standard chiper nasional. IBM mengirimkan hasil dari proye Tuchman-Meyer. Ini adalah algoritma terbaik yang diajukan dan diadopsi sebagai *Data Encryption Standard.*

DES bekerja dalam model bit , atau angka biner 0 dan 1. Setiap kelompok dari 4 bit membentuk *hexadesimal*, atau bilangan berbasis 16. Angka biner 0001 membentuk angka heksa 1, dan seterusnya. DES bekerja dengan mengenkripsi setiap group yang terdiri dari 64 bit data. Untuk melakukan enkripsi, DES membutuhkan kunci yang juga mempunyai ukuran 64 bit, namun dalam prakteknya bit ke 8 dari setiap kelompok 8 bit diabaikan, sehingga ukuran kunci menjadi 56 bit. Sebagai contoh, jika kita ingin mengenkripsi pesan "8787878787878787" dengan kunci "0E329232EA6D0D73", maka akan dihasilkan *ciphertext* "0000000000000000". Jika *ciphertext* tersebut didekripsi dengan menggunakan kunci yang sama, maka outputnya adalah pesan asli.

DES adalah sebuah "*block cipher*", artinya, DES bekerja dalam *plaintext* dengan ukuran yang telah diberikan (64 bit) dan mengembalikan *ciphertext* dengan ukuran yang sama pula.

PERANCANGAN

Pada sistem aplikasi ini proses enkripsi dan dekripsi dengan metode PBE (*Password Based Encryption*) with MD5 and DES dilakukan pada struktur data base XML yang telah dibuat, yaitu dengan cara mengenkripsi dan mendekripsi tag-tag XML yang diperlukan. Metode PBE *with* MD5 *and* DES adalah metode kriptografi simetrik yang telah disediakan pada JCE (*Java Cryptography Extension*) sebagai sebuah metode enkripsi dan dekripsi yang aplikatif terhadap pemrograman java.

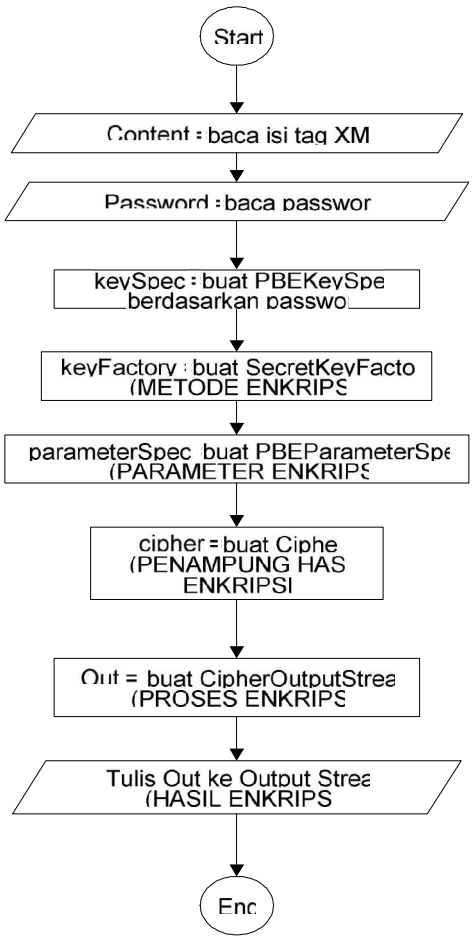
Penulisan program menggunakan metode PBE with MD5 and DES dapat dilihat pada Listing 1 di bawah ini :

```
.....
private static String METHOD = "PBEWithMD5AndDES";
    private static final byte[] salt = {
            (byte) Oxf5, (byte) Ox33, (byte) OxO1, (byte)
            Ox2a, (byte) Oxb2, (byte) Oxcc, (byte) Oxe4,
            (byte) Ox7f
            };
   private static int iterationCount = 1OO;
   private static Cipher cipher;
   private static byte[] outputArray;
   private static ByteArrayOutputStream cryptedText;
   private static ByteArraylnputStream decryptedText;
   private static String hasilEkripsi = "";
   private static String encoding = "lSO—8859—1";//
……
```

Listing 3.1 Enkripsi dan Dekripsi Metode PBE with MD5 and DES

Enkripsi

Tag XML yang telah dipilih kemudian akan dienkripsi menggunakan metode PBE with MD5 and DES untuk menghasilkan data *chipertex*. Data *chipertex* merupakan data hasil pengolahan enkripsi dari sebuah data *plaintex* yang merupakan tampilan sebuah data dari sebuah tag XML. Pada proses ini sebuah data pada tag XML akan diubah dan dikombinasikan dengan kunci *password* yang dimasukkan sehingga akan menghasilkan tampilan data *chipertex* yang merupakan tampilan karakter acak yang tidak dapat dibaca. Penulisan metode enkripsi pada *flowchart* dapat dilihat pada Gambar 6:

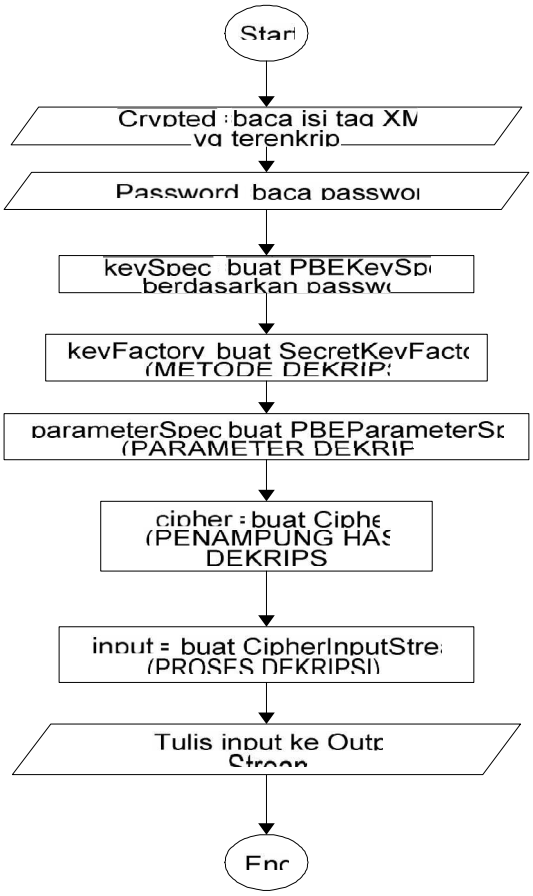


Gambar 6. Diagram alir Enkripsi

Dekripsi

Data *chipertex* yang merupakan data hasil pengolahan enkripsi dari sebuah data *plaintex* kemudian akan didekripsi kembali menjadi tampilan data *plaintex*. Pada proses dekripsi ini data *chipertex* akan dikombinasikan dengan kunci *password* sehingga akan menghasilkan data *plaintex* yang sama seperti

aslinya. Kunci *password* yang digunakan juga harus sama dengan kunci *password* pada saat melakukan proses enkripsi. Penulisan metode dekripsi pada *flowchart* dan program dapat dilihat pada Gambar 7:



Gambar 7. Diagram alir Dekripsi

PEMBAHASAN

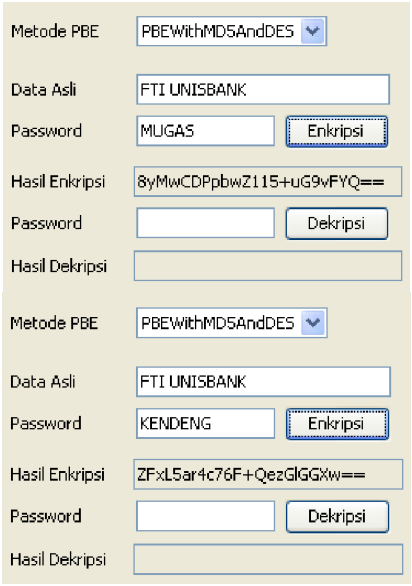
Proses enkripsi dan dekripsi dilakukan dengan menggunakan *Password Based Encryption* yang diperoleh dari JCE *(Java Cryptography Extension*). Ada 3 class yang digunakan pada penggunaan teknik kriptografi ini yaitu :

1. PBEParameterSpec Class
   Javax.crypto.spec.PBEParameterSpec
   class merupakan class yang disediakan sebagai pembawa *salt* dan *itetarion count* untuk membantu proses enkripsi dan dekripsinya.
2. The PBEKeySpec Class
   Javax.crypto.spec.PBEKeySpec
   merupakan class yang digunakan untuk memproses *password* yang digunakan dalam proses enkripsi dan dekripsinya.

3. The SecretKeyFactory Class
   Javax.crypto.SecretKeyFactory class merupakan class yang digunakan untuk mengkonversikan kunci yang digunakan pada proses enkripsi dan dekripsinya.

Seperti pada class JCE yang lainnya, SecretKeyFactory dibuat menggunakan metode getInstance ( ) method. Penulisan pada kode program pada java dituliskan :
private static String *METHOD* = "PBEWithMD5AndDES";

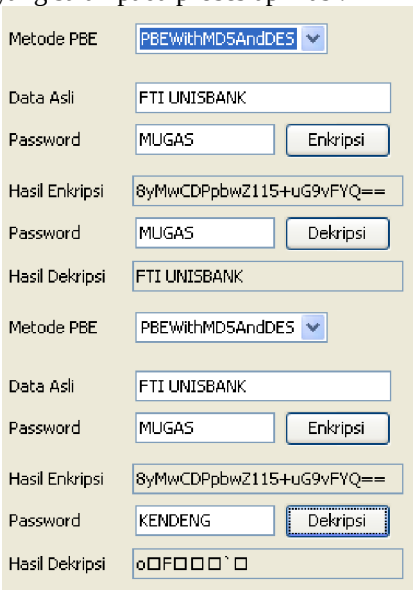
Penggunaan kunci *password* yang berbeda pada proses enkripsi dengan menggunakan data plainteks yang sama akan dihasilkan data chiperteks yang berbeda pula. Gambar 8 menunjukkan simulasi hasil data chiperteks yang berbeda dari 2 buah proses enkripsi menggunakan data plainteks yang sama dan dengan kunci *password* yang berbeda.



Gambar 8. Hasil enkripsi dengan kunci *password* berbeda

Data plainteks akan diproses menjadi data chiperteks melalui proses pengolahan pada proses enkripsi menggunakan metode *Password Based Encryption.* Data plainteks akan diolah dengan kunci *password*, *salt* dan *iteration count* sehingga akan didapatkan data terenkripsi yang berupa chiperteks. *Salt* digunakan untuk menambah sebuah *string* dari *byte-byte* yang random pada *password. Iteration count* digunakan untuk menambah perhitungan waktu yang dibutuhkan untuk mengkonversi sebuah *password* menjadi sebuah kunci. Data plainteks dan kunci tersebut akan diproses pada proses *chiper* untuk menghasilkan *encrypted data* yang disebut dengan chiperteks.

Pada proses dekripsi, data chiperteks yang merupakan hasil pengolahan proses enkripsi dapat dikembalikan lagi menjadi data plainteks atau data asli sebelumnya. Proses ini merupakan kebalikan dari proses enkripsi. *Encrypted data* yang berupa data chiperteks bersama-sama dengan *password* akan diproses untuk menghasilkan data plainteks kembali. *Password, salt, iteration count,* dan chiperteks akan diubah menjadi sebuah kunci dan akan diproses menjadi data plainteks seperti semula. Pada proses dekripsi, *password* akan sangat menentukan pada hasil proses pengembalian data chiperteks menjadi data plainteks. *Password* yang sama dengan *password* saat proses enkripsi akan dapat mengembalikan data chiperteks menjadi data plainteks aslinya. Gambar 9. memperlihatkan simulasi perbedaan data hasil dekripsi antara *password* yang benar dan yang salah pada proses aplikasi.



Gambar 9. Hasil dekripsi menggunakan *password* berbeda

KESIMPULAN

1. *Password Based Encryption* digunakan sebagai sebuah metode kriptografi simetrik yang menggunakan kunci seperti *password* dalam melakukan proses enkripsi untuk mengubah data *plaintex* menjadi *chipertex*, dan menggunakan kunci yang sama untuk melakukan proses dekripsi sehingga akan dihasilkan data yang sama dengan data *plaintex* aslinya.
2. Password Based Encryption dengan Message Digest (MD5) dan Data

Encryption Standart (DES) merupakan metode kriptografi menggunakan algoritma yang menggabungkan antara metode hashing dan enkripsi standar.

3. MD5 adalah algoritma *message digest* yang dikembangkan oleh Ronald Rivest dimana MD5 ini mengambil pesan dengan panjang sembarang dan menghasilkan message digest 128 bit. Pada MD5 pesan diproses dalam blok 512 bit dengan empat round berbeda.
4. DES adalah sebuah "*block cipher*", artinya DES bekerja dalam *plaintext* dengan ukuran yang telah diberikan (64 bit) dan mengembalikan *ciphertext* dengan ukuran yang sama pula.

DAFTAR PUSTAKA

Bruce S., 1996, *Applied Cryptography,Protocols,Algorithms,and Source Code in C.* John Wilwey & Sons, Inc

David Hook, 2005, *Beginning Cryptography With Java.* Wiley

Publishing,Inc, Indianapolis, USA

Hartono, 2003, Pemakaian Kriptografi Kunci Publik Denagn Algoritma RSA Untuk Keamanan Data XML. Universitas Gadjah Mada Yogyakarta.

Stallings W.,1999, *Cryptography and Network Security Principles and Practice second edition.*Prentice Hall, New Jersey, USA

Theodore W. Leung, 2004, *Professional XML Development with Apache Tools: Xerces, Xalan, FOP, Cocoon, Axis, Xindice,* Wrox Press, Wiley Publishing,Inc, Indianapolis, USA

Umniati, Mukodim, 2002, Perbandingan Algoritma dalam Enkripsi antara *Conventional Cryptosystems* dan *Public Key Cryptosystems. Proceeding*, Komputer dan Sistem Intelijen (KOMMIT 2002)