Jurnal Sistem Informasi ISSN P : 2598-599X; E: 2599-0330

APLIKASI KRIPTOGRAFI PESAN TEKS MENGGUNAKAN ALGORITMA *ADVANCED ENCRYPTION STANDARD* 256 BIT (AES-256) DAN DIFFIE HELLMAN

Mulyadi[1]
[1]Program Studi Teknik Informatika
Jurusan Teknologi Informasi dan Komputer Politeknik Negeri Lhoksemawe
Email: mulyadi@pnl.ac.id

Abstrak

Keamanan merupakan aspek yang penting dalam memberikan suatu informasi yang bersifat rahasia. keamanan data tersebut akan lebih terjamin kerahasiaannya bila data yang akan kita kirimkan tidak lagi bersifat data asli (plainteks) melainkan sudah di ubah dalam (chiperteks), sehingga kerahasiaan pesan tetap terjaga dari orang yang tidak bertanggung jawab. Pada penelitian ini dirancang sebuah sistem aplikasi yang dapat mengenkripsi pesan teks dan kemudian mendekripsikannya. Dengan menggunakan algoritma Advanced Encryption Standardb256 bit (AES-256) dan Diffie Helman yang merupakan algoritma kriptografi simetris, Proses ini dapat dilakukan melalui aplikasi desktop sebelum data tersebut di kirimkan sudah dienkripsi terlebih dahulu dan yang menerima perlu melakukan proses dekripsi untuk mengetahui isi dari pesan asli tersebut. Sehingga hanya pihak- pihak yang tertentu yang dapat mengetahuinya. Pada penelitian ini dilakukan proses enkripsi dan dekripsi dilakukan terhadap text. Pengujian system telah dilakukan ujicoba terhadap text dengan jumlah yang berbeda. System akan megubah data asli (plainteks) kedalam data acak (chiperteks). Penerapan aplikasi AES ini menggunakan bahasa pemrograman Java.

Kata Kunci :SMS, kriptografi, AES,enkripsi-dekripsi.

1. Pendahuluan

Perkembangan teknologi komputer saat ini sudah semakin cepat. Teknologi ini telah membuat hampir semua komputer di dunia dapat saling berhubungan. Dalam hubungan inilah terjadi aktifitas pertukaran informasi- informasi penting sehingga banyak pihak yang tidak berkepentingan ingin mendapatkan informasi tersebut. Dengan demikian,

saat ini keamanan suatu informasi sudah dianggap sangatlah penting. Apalagi untuk data yang berada di dalam jaringan publik.
Metode pengamanan suatu informasi seperti algoritma Diffie Hellmandan, Rivest Shank Adleman (RSA), dan lain lain, telah banyak diteliti oleh peneliti untuk mengamankan dan menyembunyikan informasi-informasi rahasia.

Algoritma RSA memiliki tingkat keamanan yang terletak pada sulitnya memfaktorkan sebuah bilangan besar menjadi dua buah bilangan prima tetapi memiliki kelemahan pada banyak waktu yang dibutuhkan untuk memproses pembangkitan kunci, enkripsi dan dekripsi.

Algoritma Diffie Hellman memiliki tingkat keamanannya yang lebih tinggi karena kesulitan menghitung logaritma diskrit dalam *finite field*, dibandingkan kemudahan dalam menghitung bentuk eksponensial dalam *finite field* yang sama dan membutuhkan waktu komputasi yang lebih sedikit jika dibandingkan dengan algoritma RSA.

2.TINJAUAN PUSTAKA

2.1 Pengertian Kriptografi

Kriptografi merupakan seni dan ilmu untuk memproteksi pengiriman data dengan mengubahnya menjadi kode tertentu dan hanya ditujukan untuk orang yang hanya memiliki sebuah kunci untuk mengubah kode itu kembali yang berfungsi dalam menjaga kerahasiaan data atau pesan. (Pabokory, dkk 2015).

Selain pengertian tersebut terdapat pula pengertian ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi seperti kerahasiaan data, keabsahan data, integritas data, serta autentikasi data. Dalam menjaga kerahasiaan data dengan kriptografi, data sederhana yang dikirim (plainteks) diubah ke dalaPm bentuk data sandi (cipherteks), kemudian data sandi tersebut hanya dapat dikembalikan ke bentuk data sebenarnya hanya dengan menggunakan kunci (*key*) tertentu yang dimiliki oleh pihak yang sah saja. Untuk memenuhi hal tersebut, dilakukan proses penyandian (enkripsi dan deskripsi) terhadap data tersebut

Enkripsi dilakukan pada saat pengiriman dengan cara mengubah data asli menjadi data rahasia, sedangkan deskripsi dilakukan pada saat penerimaan dengan mengubah data rahasia menjadi data asli. Jadi data yang dikirimkan selama proses pengiriman adalah data rahasia, sehingga data asli tidak dapat diketahui oleh pihak yang tidak berkepentingan. Data

asli hanya dapat diketahui oleh penerima dengan menggunakan kunci rahasia. Enkripsi proses dimana informasi/data yang hendak dikirim diubah menjadi bentuk yang hampir tidak dikenali sebagai informasi awalnya dengan menggunakan algoritma tertentu yang disebut ciphertext (Ilyas, dkk 2014).

Enkripsi yaitu suatu proses pengaman suatu data yang disembunyikan atau proses konversi data (plaintext) menjadi bentuk yang tidak dapat dibaca (ciphertext). Enkripsi telah digunakan untuk mengamankan komunikasi di berbagai negara, namun, hanya organisasi-organisasi tertentu dan individu yang memiliki kepentingan yang sangat mendesak akan kerahasiaan yang menggunakan enkripsi.

Algoritma sandi (algoritma kriptografi) adalah algoritma yang berfungsi untuk melakukan tujuan kriptografis. Menurut Shannon, Algoritma kriptografi harus memiliki kekuatan untuk melakukan:

a. Konfusi / pembingungan (*confusion*), dari teks terang sehingga sulit untuk direkonstruksikan secara langsung
b. Difusi / peleburan (*difusion*), dari teks terang menjadi *ciphertext* sehingga karakteristik dari teks terang tersebut hilang.

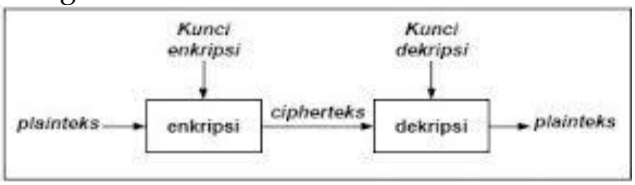
Pada implementasinya sebuah algoritma kriptografi harus memperhatikan kualitas layanan/ *Quality of Service* atau QoS dari keseluruhan sistem dimana dia diimplementasikan. Algoritma kriptografi yang handal adalah algoritma kriptografi yang kekuatannya terletak pada kunci dan distribusi kunci, bukan pada kerahasiaan algoritma itu sendiri. Teknik dan metode untuk menguji kehandalan algoritma kriptografi adalah kriptanalisa. Dasar matematis yang mendasari proses enkripsi dan dekripsi adalah relasi antara dua himpunan yaitu yang berisi elemen teks terang/ *plaintext* dan yang berisi elemen teks sandi / *ciphertext*. Enkripsi dan dekripsi merupakan fungsi transformasi antara himpunan-himpunan tersebut.

Misalkan P menyatakan plainteks dan menyatakan cipherteks, maka fungsi enkripsi E memetakan p ke C, E(P)=C D(C)=PatauD(E(P))=P
Karena proses enkripsi kemudian deskripsi mengembalikan pesan ke pesan asal, maka persamaannya dapat ditulis sebagai berikut,
D(E(P))=P.

Kriptografi modern menggunakan kunci, yang dalam hal ini algoritma tidak lagi di rahasiakan, tetapi kunci harus di jaga kerahasiaannya, kunci adalah parameter yang di gunakan untuk transformasi enciphering dan dechipering. Kunci biasanya berupa string atau deretan bilangan. Jika menggunakan kunci K, maka fungsi enkripsi dan deskripsi dapat ditulis sebagai :
EK(P)dan Cdan DK(C)=P

Dan kedua fungsi ini memenuhi :
DK(EK(P))=P

Skema enkripsi dan deskripsi dengan menggunakan kunci ditunjukkan pada gambar 1.



Gambar 1 Proses enkripsi dan deskripsi

Algoritma kriptografi dibagi menjadi dua bagian berdasarkan kunci yang digunakan:

a. Algoritma Simetri (menggunakan satu kunci untuk enkripsi dan deskripsinya).
b. Algoritma Asimetri (menggunakan kunci berbeda untuk enkripsi dan deskripsinya).

2.2 Advanced Encryption Standard (AES)

Dalam kriptografi modern, panjang kunci dalam ukuran jumlah bit yang digunakan, merupakan salah satu faktor yang sangat penting. Hal ini disebabkan karena penggunaan komputer yang sangat intensif dalam dunia kriptografi.

Advanced Encryption Standard (AES) mulai digunakan 1997. NIST mengumumkan bahwa AES digunakan sebagai pengganti dari algoritma enkripsi Data Encryption Standard (DES) yang telah lama dan kurang aman (Bodic, 2005). AES telah menjadi block chiper dimana dapat memrposes blok 128 bit dari input yang berupa plaintext dalam suatu waktu. AES juga telah pendukung pengaturan kunci 128, 192, dan 256 bit serta lebih efisien daripada DES (Hermawanm 2009).

Berikut ini dicantumkan waktu rata-rata untuk memecahkan kunci dengan teknik *brute-force.* Artinya, hanya setengah dari seluruh kemungkinan kunci yang ada dicoba dan dianggap mendapatkan satu buah kunci.

Tabel 1. Waktu yang diperlukan untuk memecahkan kunci

Ukuran Kunci	Jumlah kemungkinan kunci	1kunci/µs	106 kunci/µs
40 bit	240 = 1,1 x 1012	239 µs = 152,7 jam	55 µs
56 bit	256 = 7,2 x 1016	255 µs = 1142 tahun	10,01 jam
128 bit	2128 = 3,4 x 1038	2127 µs = 5,4 x 1024tahun	5,4 x 1018tahun

Jadi dengan panjang kunci 40 bit, jumlah kunci yang mungkin ada adalah sebanyak 240 macam kunci. Bila setengahnya (1/2 x 240 = 239) dicoba (untuk mendapatkan waktu rata-ratanya) dengan sebuah komputer yang memiliki kemampuan mencoba 1 juta kunci per detiknya (1 kunci per 1 μs), dibutuhkan waktu 239 μs untuk mencoba keseluruh 240 kunci yang ada. Jika satu komputer yang memiliki kemampuan eknripsi 1 juta kunci per mikrodetik, dibutuhkan waktu 55 μs. Waktu ini juga akan dicapai oleh 1000 komputer yang masing-masing mempunyai kemampuan enkripsi 1 milyar/detik.

Demikian pula untuk panjang kunci 128 bit. Bila digunakan 1 juta komputer yang masing-masing memiliki kemampuan enkripsi 1 triliyun per detik, diperlukan waktu 5,4 triliyun tahun. Tapi harus diingat, teori ini hanya berlaku dengan syarat tidak ada kelemahan lain dalam sistem yang menggunakan algoritma kriptografi.

AES adalah jenis algoritma kriptografi yang sifatnya simetri dan *cipher block*. *Cipher block* merupakan algoritma yang masukan dan keluarannya berupa satu blok dan setiap bloknya terdiri dari banyak bit (misalnya 1 blok terdiri dari 64 atau 128 bit). Dengan demikian algoritma AES mempergunakan kunci yang sama pada saat enkripsi dan dekripsi serta masukan dan keluarannya berupa blok dengan jumlah bit tertentu. AES mendukung berbagai variasi ukuran blok dan kunci yang akan digunakan. Namun AES mempunyai ukuran blok dan kunci yang tetap sebesar 128, 192, 256 bit. Pemilihan ukuran blok data dan kunci menentukan jumlah proses yang harus dilalui untuk proses enkripsi dan dekripsi. Berikut adalah perbandingan jumlah proses yang harus dilalui untuk masing-masing masukan.

Tabel 2. Jumlah putaran berdasarkan panjang kunci

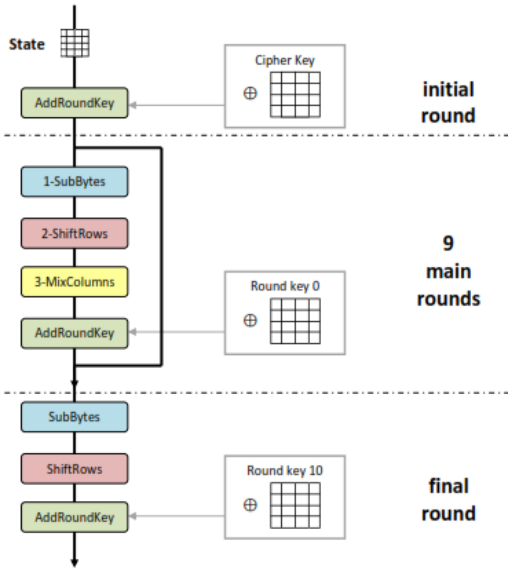
Jenis AES	Panjang Kunci (Nk Double words)	Ukuran Blok (Nb Double words	Jumlah Putaran (Nr)
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

Dari tabel di atas terlihat bahwa AES-128 menggunakan panjang kunci Nk = 4 *double word* yang setiap *double word* berisi 32 bit sehingga total kuncinya 128 bit. Ukuran blok masukan 4 *double word* (128 bit), sedangkan jumlah rondenya (Nr) sebanyak 10 ronde. AES-256 memiliki panjang kunci 256 bit atau 8 *double word*, blok masukan *plaintext* 128 bit dan jumlah ronde 14.

2.2.1 Proses Enkripsi Algoritma AES

Proses enkripsi algoritma AES terdiri dari 4 jenis transformasi byte, yaitu *SubBytes, ShiftRows,* dan *AddRoundKey*. Setiap putaran menggunakan kunci internal yang berbeda. Sama seperti pada proses enkripsi, proses dekripsi AES juga terdiri empat langkah (Stalling, 2005; FIPSP, 2001). Pada awal proses enkripsi, input yang telah disalin ke dalam state akan mengalami transformasi AddRoundKey. Setelah itu state akan mengalami transformasi SubBytes, ShiftRows, MixColumns, dan AddRoundKey secara berulang-ulang sebanyak round/putaran (Nr). Proses ini dalam algoritma AES disebut sebagai round function. Round yang terakhir, state tidak diberikan transformasi MixColumns (Voni, dkk 2009).

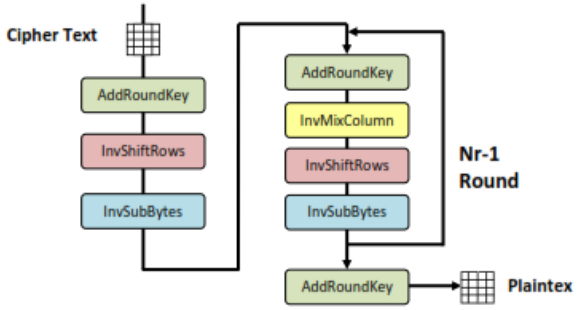
Berikut akan dijelaskan proses enkripsi dari algoritma AES secara garis besar:



Gambar 1. Proses Enkripsi Algoritma AES

2.2.2 Proses Dekripsi Algoritma AES

Transformasi *cipher* dapat dibalikkan dan diimplementasikan dalam arah yang berlawanan untuk menghasilkan *inverse cipher* yang mudah dipahami untuk algoritma AES. Transformasi *byte* yang digunakan pada *invers cipher* adalah *InvShiftRows, InvSubBytes, InvMixColumns,* dan *AddRoundKey*. Algoritma dekirpsi dapat dilihat pada skema berikut ini :



Gambar 2. Ilustrasi proses dekripsi AES

2.3 Diffie Hellman

Algoritma ini pertama kali diperkenalkan oleh Whitfield Diffie dan Martin Hellman pada tahun 1975. Mereka berdua adalah peneliti pada universitas Stanford. Mereka memperkenalkan algoritma ini untuk memberi solusi atas pertukaran informasi secara rahasia. Algoritma ini tidak berdasarkan pada proses *enkripsi* dan *dekripsi*, melainkan lebih kepada proses matematika yang dilakukan untuk menghasilkan kunci rahasia yang dapat disebarkan secara bebas tanpa harus khawatir karena kunci rahasia tersebut hanya dapat *didekripsi* hanya oleh pengirim dan penerima pesan. Dasar dari algoritma ini adalah matematika dasar dari aljabar eksponen dan aritmatika modulus**.**

2.3.1 Algoritma Diffie Hellman

Langkah-langkah dalam pertukaran kunci dengan menggunakan algoritma Diffie-Hellman adalah sebagai berikut:

a. Pilih bilangan prima yang besar, *p*dan bilangan integer yang tidak melebihi dari nilai *p*,*g,* biasa disebut bilangan basis atau generator. Kedua bilangan tersebut dapat diketahui secara publik.
b. Pilih sebuah bilangan acak oleh pengirim, *x*, bilangan ini tidak boleh diketahui oleh orang lain.
c. Pilih sebuah bilangan acak oleh penerima, *y*, bilangan ini tidak boleh diketahui oleh orang lain.
d. Pengirim menghitung A = gx mod p. Bilangan A ini dapat diketahui secara publik.
e. Penerima menghitung B = gy mod p. Bilangan B ini dapat diketahui secara publik.
f. Lakukan pertukaran bilangan A dan B terhadap pengirim dan penerima.
g. Lalu Pengirim menghitung ka = Bx mod p.
h. Penerima menghitung kb = Ay mod p.

i. Berdasarkan hukum aljabar nilai kx sama dengan kb atau bisa disebut ka= kba = k. sehingga pengirim dan penerima tersebut mengetahui kunci rahasia tersebut "k".

Bukti dari ka=k=k :

ka=kb
Bx mod p=Ay mod p
(gy mod p)x
mod p=(gx mod p)y mod p
(gy)x mod p=(gx)y mod p
gyx mod p=gxy mod p

Contoh penggunaan dari algoritma ini adalah:

1. Alice dan Bob menetapkan p = 23 dan g = 5.
2. Eve (penyadap) tahu nilai p dan$g$.
3. Alice memilih nilai x = 6 dan Bob memilih nilai y = 15.
4. Alice menghitung nilai A = 56 mod 23 = 8.
5. Bob menghitung nilai B = 515 mod 23 = 19.
6. Alice dan Bob bertukar nilai A dan B.
7. Eve menyadap mereka dan tahu nilai A dan B.
8. Alice melakukan perhitungan ka = 196 mod 23 =2.
9. Bob melakukan perhitungan kb = 815 mod 23 =2.
10. Eve mengetahui nilai p ,g, A, dan B tetapi dia tidak dapat mengetahui kunci rahasia, k dari Bob dan Alice.

Alice dan Bob dapat mengetahui kunci rahasia tersebut dan dapat bertukar pesan dengan aman tanpa harus diketahui oleh Eve. Eve hanya dapat mengetahui nilai p,g, A, dan B tetapi tidak dapat menghitung kunci rahasia dari mereka berdua. Sehingga Eve tidak dapat mengetahui pesan rahasia apa antara Alice dan Bob.

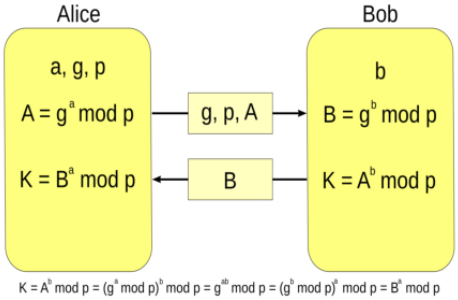
Algoritma ini tidak hanya terbatas pada 2 pengguna saja. Jumlah pengguna yang ingin menggunakan pertukaran kunci menggunakan algoritma Diffie Hellman ini tidak dibatasi. Hal ini hanya berlaku jika memenuhi 2 prinsip yang harus dilakukan:

a. Bilangan p dan g yang telah disetujui oleh semua anggota.
b. Setiap anggota harus melakukan pertukaran data yang diperlukan oleh anggota lainnya sehingga semua data dapat didapatkan secara merata g.

3. METODELOGI PENELITIAN

3.1 Gambaran Umum Aplikasi Kriptografi Diffie Hellman

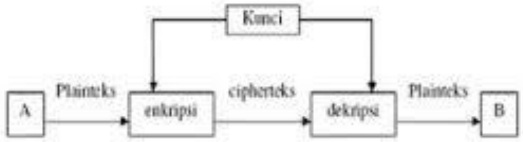
Secara Umum Aplikasi Kriptografi Diffie Hillman ini mempunyai fungsi untuk mengamankan pesan atau file dengan cara pertukaran kunci publik untuk mendapatkan kunci privat, untuk mendapatkan kunci privat pengirim dan penerima harus saling bertukar kunci publik dan menghitung kembali dengan nilai yang sama unyuk mendapatkan kunci privat. Kunci privat yang di hasilkan pengirim dan penerima saat pencarian akan bernilai sama. Gambar 3.1 merupakan blok diagram dari sistem aplikasi kriptografi pertukaran kunci yang akan dibuat.



Gambar 3. Gambaran Umum Diffie Hellman

3.2 Gambaran Umum Aplikasi Kriptografi AES

Secara Umum Aplikasi Kriptografi AES ini mempunyai fungsi untuk mengamankan pesan atau file dengan cara teknik mengubah pesan asli (plainText) menjadi pesan atau file rahasia (ciphertext) yang tidak bias dimengerti atau dibaca oleh orang yang tidak berhak, Gambar 3.2 merupakan blok diagram dari sistem aplikasi kriptografi yang akan dibuat.

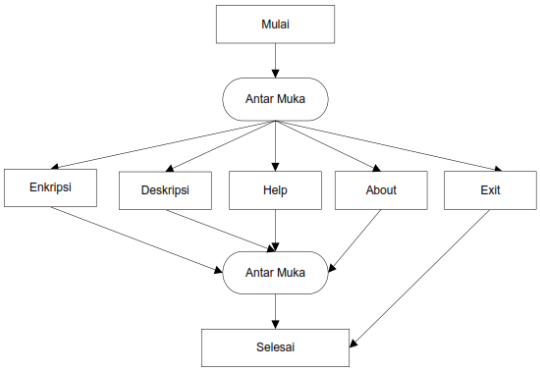


Gambar 4. Blok Diagram Kriptografi

Sistem aplikasi membutuhkan inputan berupa pesanteks atau file sebelum melakukan proses enkripsi. Proses Enkripsi merupakan proses pengubahan pesan asli menjadi pesan rahasia dimana pada proses enkripsi menggunakan algoritman AES yang merupakan algoritma simetris dimana algoritma ini membutuhkan kunci (key) yang sarna ketika pada proses enkripsi dan dekripsi. Proses dekripsi merupakan proses perubahan pesan rahasia menjadi pesan asli atau plaintext.

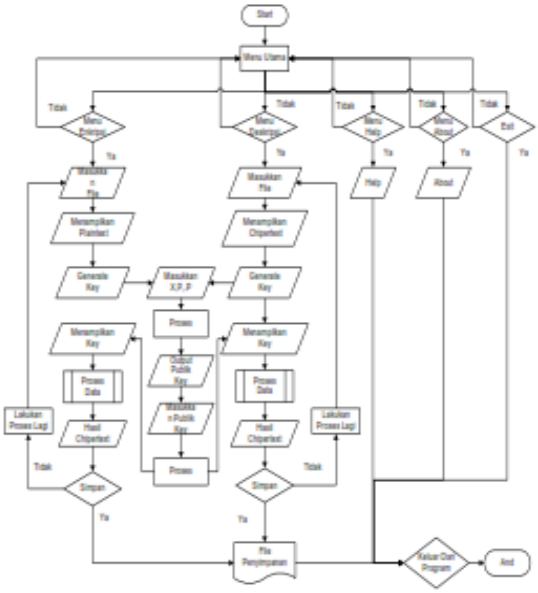
3.3 Perancangan Flowchart

Rancangan ini di gunakan untuk mendesain dan mempresentasikan program. Sebelum pembuatan program, fungsinya adalah untuk mempermudah programmer dalam menentukan alur program yang akan dibuat. Berikut ini merupakan blok diagram secara umum dari aplikasi Diffie Hellman dan AES



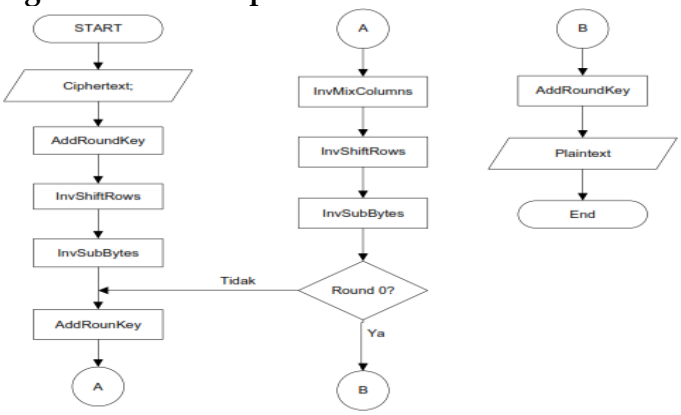
Gambar 5. Blok Diagram Aplikasi Secara Umum

3.4 Perancangan Flowchart Aplikasi Enkripsi dan Dekripsi Pesan Teks



Gambar 6. *Flowchart* **Keseluruhan proses aplikasi kriptografi Diffie Hellman dan AES (Advanced Encryption standart)**

3.5 Perancangan Proses Enkripsi



Gambar 7. *Flowchart* proses Enkripsi

4. HASIL DAN PEMBAHASAN

Hasil pengujian sistem didapat dengan melakukan pengujian aplikasi berdasarkan kesesuaian *output* berupa *ciphertext* dengan *input* yang diberikan berupa *plaintext* dan *key*.

4.1Tampilan Menu Utama

Tampilan pada menu utama aplikasi enkripsi dan dekripsi teks ini terdiri dari tombol menu *Enkripsi, Dekripsi, Help, About, Exit.* Setiap pilihan tombol menu yang terdapat pada menu utama memiliki fungsi dan *activity* masing-masing. Pemanggilan *activity* pada aplikasi ini menggunakan a.setVisible(true); Sebagai contoh, berikut adalah program untuk memanggil *activity* pada tombol menu Enkripsi yang terdapat pada menu utama.

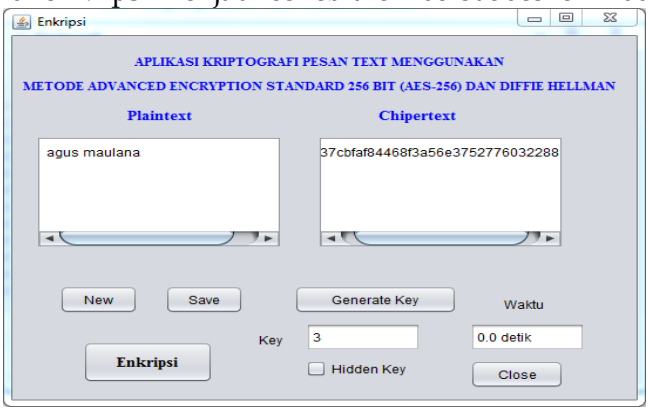
```
private void jButton1ActionPerformed(java.awt.event.ActionEvent
evt) {
        // TODO add your handling code here:
        Enkripsi a = new Enkripsi();
        a.setVisible(true);
    }
```

Adapun tampilan menu utama pada aplikasi ini dapat dilihat pada Gambar 8 berikut :

Gambar 8. Tampilan *form* menu utama

4.2 Tampilan Menu *Enkripsi*

Dalam menu *Enkripsi* seperti yang terlihat pada Gambar 4.2 terdapat objek *TextArea* yang pertama yaitu untuk memasukkan plaintext yang akan kita enkripsikan ke dalam chipertext yang akan di tampilakn di *TextArea*lain. Tombol button new digunakan untuk mengulangi kembali proses dari tahap awal. Sedangkan tombol button save berfungsi untuk menyimpan pesan teks yang telah di
ubah ke dalam chipertext. Tombol button generate key berfungsi untuk menghitung kunci menggunakan algoritma diffie hellman. Aplikasi ini bekerja setelah *user* memasukkan *message (plaintext)* dan *key*. Gambar 4.2 memperlihatkan hasil Enkripsi, dimana pada plaintext di tulis agus maulana dan di enkripsi menjadi 837cbfaf84468f3a56e3752776032288.



Gambar 9. Proses enkripsi *plaintext*

Proses ini dilakukan ketika *user* sudah memasukkan pesan dan *key*, selanjutnya *user* menekan tombol enkripsi yang terletak di bawah tombol *open*. Pada saat tombol ditekan, sistem akan menjalankan perintah berikut :

```
private void jButton3ActionPerformed(java.awt.event.ActionEvent
evt) {
        // TODO add your handling code here:
        long start;
        long end;
        start = System .currentTimeMillis();
        end = System.currentTimeMillis();

        hasil =
AES.encode(jTextArea1.getText().toString(),jTextField1.getText(
).toString());
        jTextArea2.setText(hasil);

        double total = (double) ((end - start)/1000.0);
        jTextField3.setText(+total+" detik ");


    }
```

Perintah di atas akan dijalankan pada saat tombol enkripsi ditekan. Sistem akan memanggil *method void* dengan instruksi "private void jButton3ActionPerformed(java.awt.event.ActionEvent evt)
Hasil
AES.encode(jTextArea1.getText().toString(),jTextField1.getText().toString();
jTextArea2.setText(hasil);
Proses enkripsi ini memanggil *subclass* dengan nama *encode* pada *class* AES. *Class* AES inilah yang melakukan proses enkripsi berdasarkan *input* yang diterima berupa variabel *"plaintext"* dan *"key"*. Setelah mendapatkan *ciphertext,* kemudian *ciphertext* tersebut disimpan dalam variabel "hasil" untuk kemudian ditampilkan pada *JTextArea2.* dengan instruksi "jTextArea2.setText(hasil);".

4.3 Tampilan Menu Deskripsi

Dalam menu *Dekripsi* seperti yang terlihat pada Gambar 4.3 terdapat objek *TextArea* yang pertama yaitu untuk memasukkan chipertext yang akan di dekripsikan ke dalam plaintext.

Gambar 10. Proses Dekripsi *plaintext*

Inputan yang berupa chipertext 837cbfaf84468f3a56e3752776032288 dirubah kembali menjadi plaintext agus maulana, Proses ini dilakukan ketika *user* sudah memasukkan pesan dan *key,* selanjutnya *user* menekan tombol dekripsi. Pada saat tombol ditekan, sistem akan menjalankan perintah berikut :

```
private void jButton3ActionPerformed(java.awt.event.ActionEvent
evt) {
        // TODO add your handling code here:
        long start;
        long end;
        start = System .currentTimeMillis();
        end = System.currentTimeMillis();

        String panjang = jTextField1.getText().toString();
        hasil =
AES.decode(jTextArea1.getText().toString(),jTextField1.getText(
).toString());
        jTextArea2.setText(hasil);

        double total = (double) ((end - start)/1000.0);
        jTextField3.setText(+total+" detik ");
    }
```

Sistem akan memanggil *method void* dengan instruksi "private void jButton3ActionPerformed(java.awt.event.ActionEvent evt)
hasilAES.encode(jTextArea1.getText().toString(),jTextField1.getText().toString());
jTextArea2.setText(hasil);

Proses enkripsi ini memanggil *subclass*dengan nama*encode*pada *class*AES. *Class*AES inilah yang melakukan proses enkripsi berdasarkan *input* yang diterima berupa variabel *"plaintext"* dan *"key"*. Setelah mendapatkan *ciphertext,* kemudian *ciphertext*tersebut disimpan dalam variabel "hasil" untuk kemudian ditampilkan pada *JTextArea2.* dengan instruksi "jTextArea2.setText(hasil);".

5. KESIMPULAN

Setelah melakukan perancangan dan pembahasan mengenai aplikasi enkripsi dan dekripsi pesan teks pada aplikasi java menggunakan algoritma AES-256 pada bab terdahulu, maka dapat diambil simpulan sebagai berikut :

1. AES merupakan algoritma kriptografi yang menggunakan fungsi polinomial dalam tahapan proses enkripsi dan dekripsinya.
2. *Ciphertext* yang dihasilkan pada aplikasi ini dalam bilangan hexadecimal. Hal ini dikarenakan *output* yang diterima tidak diubah lagi ke dalam bentuk *string.*
3. Algoritma Diffie-Hellman lebih memfokuskan dalam perubahan nilai kunci dan proses matematis dalam penentuan kunci akhir yang sama. Sedangkan Algoritma AES lebih memfokuskan pada saat *enkripsi* dan *dekripsi*.
4. *Ciphertext* yang dihasilkan pada aplikasi ini dalam bilangan hexadecimal. Hal ini dikarenakan *output* yang diterima tidak diubah lagi ke dalam bentuk *string.*
5. Algoritma AES-256 bekerja berdasarkan ukuran blok. Jadi AES melakukan proses enkripsi per 128 bit atau sama dengan 16 *byte.* Apabila *plaintext*> 128 bit, sistem akan mengambil 128 bit pertama untuk dienkripsi kemudian sistem akan melakukan proses enkripsi lagi pada bit sisanya. Selanjutnya hasil dari kedua proses tersebut digabungkan, sehingga panjang karakter pada *plaintext* akan mempengaruhi *ciphertext* yang dihasilkan ini.

6. SARAN

Untuk pengembangan lebih lanjut aplikasi enkripsi dan dekripsi pesan teks menggunakan algoritma AES-256 ini
dapat dilakukan:

1. Aplikasi ini dapat di kembangkan dengan menambahkan AES 128 *byte* dan 192 *byte* dan dengan menggunakan bahasa pemrograman lain.
2. Implementasi algoritma *hash function* MD5 untuk penerapan kunci. Algoritma MD5 memiliki keluaran karakter berjumlah 32. Jadi jumlah karakter *key* yang dimasukkan dapat berjumlah berapapun, karena nantinya algoritma MD5 akan mengeluarkan nilai *hash* sebanyak 32 karakter.
3. Aplikasi ini dapat di kembangkan pada untuk aplikasi android agar dapat di gunakan secara luas.

DAFTAR PUSTAKA

Advanced Encryption Standard " Jurnal Informatika Mulawarman, Vol. 10, no. 1, Februari 2015.

Ariyus, Dony. 2008. *Pengantar Ilmu Kriptografi Teori Analisis dan Implementasi.* Yogyakarta : ANDI.

Bodic, G., L., 2005, Mobile Meesaging Technologies and Services SMS, EMS and MMS Second Edition, John Willey and Sons Ltd, England

Bellare, Mihir & Rogaway, Phillip. 2005. *Introduction to Modern Cryptography.* USA : Universiry of California.

Dony, A. 2005, Kriptografi Keamanan Data dan Komunikasi, Yogyakarta: Penerbit Andi Offset.

Federal Information Processing Standards Publication 197, *Announcing the Advanced Encryption Standard,* U.S. Doc/NIST, November 26, 2001.

Fitrianti, U., & Ula, M. (2017). Implementasi algoritma levenshtein distance dan algoritma knuth morris pratt pada aplikasi asmaul husna berbasis android. *Jurnal Sistem Informasi*, *1*(2).

F.N. Pabokory, I. F. Astuti and A.H. Kridalaksana, "Implementasi Kriptografi Pengamanan

Fuadi, W., Ula, M., & Sadli, M. (2015). The Introduction Types of Vocal Sound in Choir in Realtime Using Hankel Transformation and Macdonald Function. Academic Research International, 6(1), 1.

Goldreich, Oded. 2004. *Foundation of Cryptography: Basic Application.* Cambridge University.

Hermawan, A.P., 2009, Kompresi Pesan SMS Menggunakan Algoritma Modified Half-Byte, Thesis, Ilmu Komputer, Universitas Gadjah Mada, Yogyakarta.

Henriquez, Dkk. 2007. *Cryptographic Algorithms on Reconfigurable Hardware.* Springer.

I.A. Ilyas and S.Widodo, "Kriptografi File Menggunakan Metode Aes Dual Password" Prosiding Seminar Ilmiah Nasional Komputer dan Sistem Intelijen (KOMMIT 2014), Depok, 2014, Vol. 8. Page 263.`

Munir, Rinaldi. 2006. Kriptografi. Bandung: Penerbit Informatika.

Stallling, William, 2005, Cryptography and Network Security Principles and Practices, 4thPrentice Hall, New Jersey

Voni Yuniati, dkk. "Enkripsi dan Dekripsi Dengan Algoritma AES-256 Untuk Semua Jenis File". Jurnal nformatika Volume 5 No. 1 April 2009.