Perancangan dan Implementasi Modifikasi Algoritma VEA (*Video Encryption Algorithm*) untuk Video *Streaming*

Dian Intania Savitri

Laboratorium Ilmu dan Rekayasa Komputasi
Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung
e-mail: if13081@students.if.itb.ac.id

ABSTRAK

Makalah ini akan membahas model enkripsi video *streaming* yang dibangun dengan tujuan mengamankan data video yang akan dikirim melalui jaringan komputer dengan metode *streaming*. Enkripsi data akan dilakukan di sisi *transmitter*, yaitu sisi pengirim video, dan kemudian data tersebut dikirim ke sisi *receiver* sebagai penerima video. Sebelum dapat dijalankan, data video yang dikirim harus didekripsi terlebih dahulu oleh sisi *receiver*. Kunci yang digunakan pada sisi *transmitter* dan *receiver* harus sama. Jika kunci yang digunakan tidak sama, maka *receiver* tidak dapat mendekripsi data tersebut dimana pada akhirnya video tidak dapat ditampilkan. Algoritma enkripsi video yang dipilih untuk diimplementasikan pada video *streaming* adalah VEA (*Video Encryption Algorithm*). Algoritma VEA umum digunakan untuk keperluan enkripsi video karena kemudahannya dalam implementasi, terutama karena algoritma ini mengenkripsi video bit per bit. Algoritma VEA selanjutnya dimodifikasi sedemikian rupa agar cocok diimplementasikan terhadap model enkripsi video *streaming*. Modifikasi dilakukan dengan menambahkan algoritma kriptografi kunci rahasia DES dan fungsi *hash* MD5. Selain itu juga, operasi yang akan dilakukan oleh modifikasi algoritma VEA bukan lagi bit per bit namun per paket-paket data yang akan dikirim *transmitter*. Paket-paket data ini akan dienkripsi menggunakan DES, dimana masukan kunci untuk DES di-*hash* terlebih dahulu menggunakan MD5. Model enkripsi video *streaming* ini dibangun dengan menggunakan bahasa pemrograman Java dan memanfaatkan kakas *Java Media Framework* (JMF).

Kata kunci: video *streaming*, enkripsi, DES, MD5, JMF.

Makalah diterima 15 Juni 2007. Revisi akhir 14 Juni 2007.

1. PENDAHULUAN

Keamanan data multimedia sangat penting dalam bisnis komersil maupun tradisional saat ini. Contohnya, pada aplikasi *video on demand*, hanya orang yang membayar yang dapat menonton video tersebut. Selain itu juga pada aplikasi *video conferencing*, hanya orang yang berkepentingan saja yang dapat ikut serta dalam konferensi tersebut dan mendapatkan datanya.

Salah satu cara untuk mengamankan aplikasi *distributed multimedia* seperti pada contoh-contoh di atas adalah dengan mengenkripsinya menggunakan algoritma kriptografi seperti DES (*Data Encryption Standard*) atau IDEA (*International Data Encryption Algorithm*) [2]. Masalahnya, algoritma kriptografi tersebut memiliki komputasi yang rumit. Implementasi dari algoritma kriptografi ini tidak cukup cepat untuk memproses sejumlah besar data yang dihasilkan oleh aplikasi multimedia. [3]

Dua hal yang dapat diperhatikan dari enkripsi data multimedia adalah: pertama, ukuran data multimedia biasanya sangat besar. Sebagai contoh, ukuran data dari video MPEG-1 berdurasi dua jam kira-kira 1 GB. Kedua, data multimedia harus diproses *real-time*. Memproses sejumlah besar data saat *real-time* dengan algoritma kriptografi yang rumit akan memperberat kinerja komputer serta jaringannya, dan juga tidak nyaman bagi orang yang menonton video tersebut secara *real-time* karena hal tersebut dapat berpengaruh juga pada *delay* video yang sedang ditontonnya.

Untuk beberapa jenis aplikasi video komersil, seperti program *pay-per-view*, informasi yang terdapat sangat banyak, tetapi nilai dari informasi tersebut sangat rendah. Serangan terhadap data ini sangat mahal tetapi tidak menguntungkan karena untuk memecahkan kode enkripsi video tersebut jauh lebih mahal dibandingkan membeli program untuk menonton video tersebut [3]. Walaupun begitu, algoritma enkripsi yang ringan yang dapat menghasilkan tingkat keamanan yang cukup memuaskan dan memiliki komputasi yang ringan sangat dibutuhkan pada aplikasi video.

Banyak algoritma enkripsi video yang telah dibangun sampai saat ini, tetapi algoritma yang umum digunakan terutama untuk aplikasi video *streaming* adalah algoritma

Video Encryption, atau sering disebut juga VEA (*Video Encryption Algorithm*). Alasan banyaknya penggunaan algoritma ini adalah karena tingkat keamanannya yang cukup memuaskan, komputasi yang ringan, dan cocok diimplementasikan di lingkungan video *streaming* karena algoritmanya yang dapat berbasis *stream cipher* maupun *block cipher*, tergantung kebutuhan saat *streaming* video tersebut. Implementasi dari algoritma ini diharapkan dapat memenuhi kebutuhan keamanan dalam video *streaming*, terutama apabila *Video Encryption* ini dapat dikombinasikan dengan algoritma kriptografi kunci rahasia seperti DES atau AES yang sudah cukup terkenal tingkat keamanannya. Sebelum enkripsi video dapat diterapkan pada video *streaming*, akan lebih baik jika model dari penerapan enkripsi video pada video *streaming* dirancang terlebih dahulu, agar enkripsi video dapat berjalan dengan baik.

2. VEA

Sama dengan enkripsi pada data teks, enkripsi video juga memiliki algoritma sendiri. Umumnya video dapat dienkripsi langsung dengan menggunakan algoritma enkripsi kunci rahasia yang telah banyak beredar saat ini, seperti DES, AES, dan lain-lain, tetapi enkripsi seperti itu membutuhkan waktu yang cukup lama, karena ukuran video yang cukup besar. Algoritma enkripsi video yang umum digunakan adalah VEA (*Video Encryption Algorithm*). Algoritma ini memiliki berbagai macam modifikasi yang disesuaikan berdasarkan kebutuhan, karena kemudahannya dalam implementasi. Salah satu tipe modifikasinya diberi nama MVEA (*Modified VEA*).

Berikut adalah skema global dari algoritma VEA:

1. Buka file MPEG.
2. Baca *frame* file MPEG, baca tipe *frame*-nya.
3. Baca *stream* bit dari *frame* tersebut.
4. Jika *frame* dari *stream* bit bukan *frame* I, maka *stream* bit langsung ditulis ke file tujuan.
5. Jika *stream* bit tersebut merupakan *stream* bit dari *frame* I, maka bit-bit tersebut di-XOR-kan dengan kunci.
6. Tulis hasil enkripsi ke file tujuan.
7. Baca *frame* selanjutnya, kembali ke langkah nomor 2 sampai *End-of-File*.

Diagram alir dari algoritma VEA dapat dilihat di Gambar 1.

VEA (*Video Encryption Algorithm*) merupakan sebuah algoritma enkripsi video yang berbasis pada *cipher* aliran (*stream cipher*). Kunci rahasia VEA, *k*, di-*generate* secara random dalam bentuk *bitstream* dengan panjang *m*, yang dapat ditulis sebagai $k = b_1 b_2 ... b_m$. *Bitstream* dari video dapat direpresentasikan dengan:

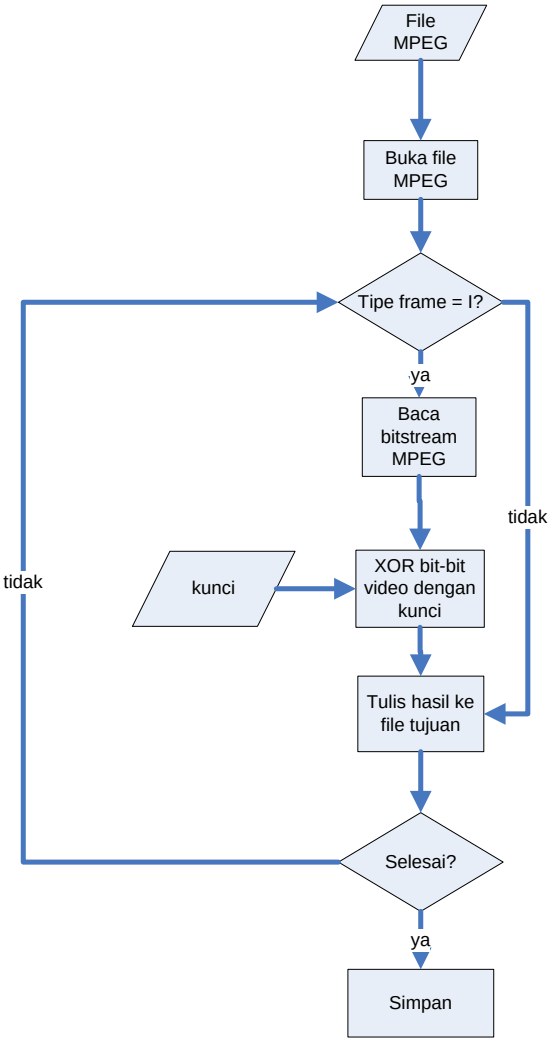
$$S = ...s_1...s_2...s_m...s_{m+1}...s_{m+2}...s_{2m}.. \quad (1)$$

yang dalam hal ini s_i (i = 1,2,...) adalah seluruh bit-bit dari video.

Fungsi enkripsi VEA, E_k, dapat ditulis dengan:

$$E_k(S) = ...(b_1 \oplus s_1) ... (b_m \oplus s_{m+1}) ... (b_m \oplus s_{2m}) ... \quad (2)$$

yang dalam hal ini $\oplus$ adalah operasi XOR.



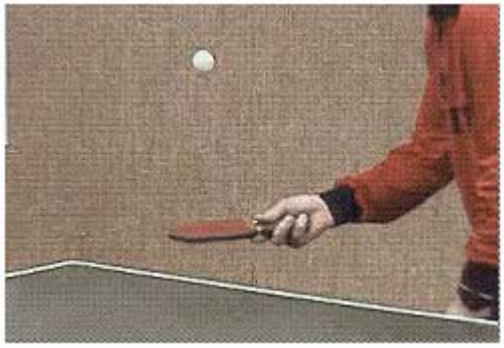
Gambar 1. Diagram Alir Algoritma VEA untuk video MPEG

Detail algoritma *Video Encryption* (VEA) dalam bentuk algoritma umumnya adalah sebagai berikut:

```
Algoritma VEA(
       int m,        /*key length*/
       bit key[m],   /*secret key*/
       char *mpeg_video,    /*input file*/
       char *vea_mpeg_video)/*output file*/
{
       int n;        /*buffer size*/
       bit video[n]; /*input buffer*/
       file in;             /*for input*/
       file out;            /*for output*/
```

```
       int k,l,i = 0;
       in = open(mpeg_video,“r“);
       out = open(vea_mpeg_video,“w“);
       while(!eof(in)){
          l = read(video,n,in);
       /*read l bits*/
          for(k=0,k<l;k++){
               switch(video[k]){
                 case(beginning of a GOP):
                     i=0;
       /*resynchronization*/
                     break;
                 case():
                     video[k] = video[k] xor
                      key[i];
                     i = ++i mod m;
               }     /*end switch*/
          }   /*end for*/
          write(video,l,out);
     } /*end while*/
       close(in);
       close(out);
}
/*end procedure*/
```

Contoh hasil enkripsi video yang didapat dengan menggunakan algoritma VEA dapat dilihat pada Gambar 2 sampai Gambar 4.



Gambar 2. Gambar Asli

Gambar 3. Contoh I Hasil Enkripsi Menggunakan VEA

Gambar 4. Contoh II Hasil Enkripsi Menggunakan VEA

3. MODIFIKASI ALGORITMA VEA

Basis algoritma *Video Encryption* yang berupa *cipher* aliran (*stream cipher*) akan tetap dipertahankan. Biasanya aliran *cipher* berupa bit-bit 0 atau 1, yang kemudian oleh algoritma VEA di-XOR-kan dengan bit-bit kunci. Untuk keperluan tugas akhir ini yang mengimplementasikan video *streaming*, VEA juga akan dimodifikasi sesuai kebutuhan model *streaming* video yang dirancang. Aliran data yang akan dienkripsi berupa paket-paket data, dan kemudian paket-paket data ini dienkripsi dengan algoritma kriptografi *cipher blok* DES. Input kunci untuk DES dibangun dengan menggunakan fungsi *hash* MD5. Oleh karena itu, secara garis besar, beberapa modifikasi dari algoritma VEA adalah:

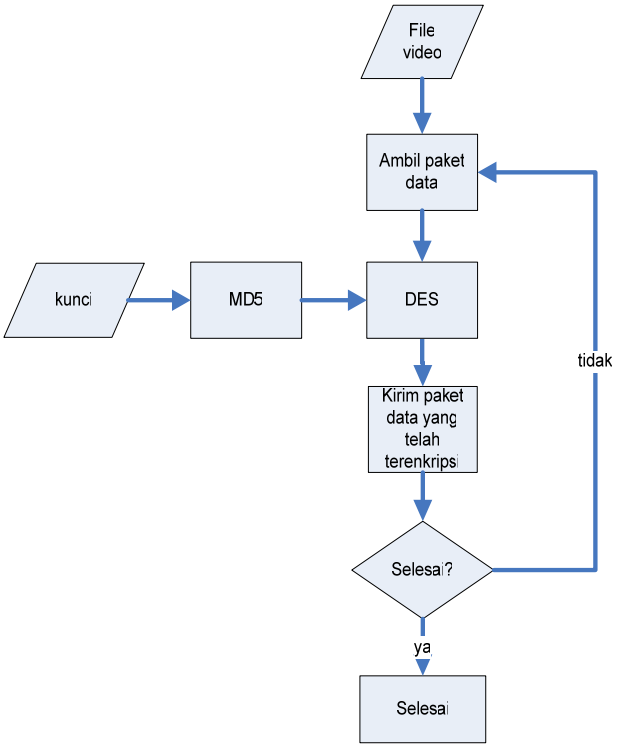
1. Aliran data untuk cipher aliran (*stream cipher*) dari VEA bukan berupa bit-bit data 0 atau 1, melainkan berupa paket-paket data hasil keluaran *RTP Socket Adapter*.
2. Tiap paket data dienkripsi menggunakan algoritma kriptografi kunci rahasia DES (modifikasi dari algoritma VEA original yang hanya meng-XOR-kan bit-bit aliran data dengan kunci saja).
3. Kunci untuk mengenkripsi paket data hasil masukan user di-*hash* terlebih dahulu menggunakan fungsi *hash* MD5 sebelum digunakan oleh DES.

Berikut adalah skema global dari modifikasi algoritma VEA:

1. Ambil satu buah paket data dari file video. Paket tersebut merupakan keluaran dari *RTP Socket Adapter*.
2. Enkripsi paket data tersebut menggunakan algoritma kriptografi DES. Masukan kunci untuk DES berasal dari kunci hasil masukan pengguna yang telah di-hash menggunakan MD5 terlebih dahulu.
3. Kirim paket data yang telah terenkripsi tersebut ke *receiver*.

4. Baca paket data selanjutnya, kembali ke langkah nomor 1 sampai selesai.

Diagram alir dari modifikasi algoritma VEA dapat dilihat pada Gambar 5.



Gambar 5. Diagram Alir Modifikasi Algoritma VEA

4. MODEL ENKRIPSI VIDEO *STREAMING*

Model fungsional perangkat lunak memberikan gambaran umum mengenai proses-proses yang terjadi dalam perangkat lunak beserta detail bagaimana proses-proses tersebut diimplementasikan. Model fungsional juga memberikan gambaran tentang aliran data yang terjadi antar proses-proses yang ada maupun antar proses dengan entitas luar. Aliran data tersebut akan mendefinisikan masukan dan keluaran yang terdapat pada masing-masing proses yang terjadi, sehingga hubungan antar proses dapat terlihat dengan jelas.

Rancangan model enkripsi video pada video *streaming* dapat dilihat pada Gambar 6 sampai Gambar 9. Gambar 6 merupakan gambaran umum proses video *streaming* yang terjadi pada sisi *transmitter*. Gambar 7 merupakan rancangan model enkripsi video yang diterapkan pada saat video akan dikirim ke *receiver*, dimana proses enkripsi video diletakkan di bagian *send stream*. Bagian yang diberi warna abu-abu merupakan bagian proses enkripsi yang ditambahkan pada model video *streaming*. Gambar 8 merupakan gambaran umum proses video *streaming* di sisi *receiver*, sedangkan Gambar 9 merupakan penerapan dekripsi video pada sisi *receiver*-nya. Bagian yang diberi warna abu-abu merupakan bagian proses dekripsi yang ditambahkan pada model video *streaming*.

Berikut adalah keterangan dari Gambar 6 sampai Gambar 9:

4.1 *Transmitter*

File video:
File video yg akan dikirimkan melalui *streaming*. Format yg dapat didukung:

1. MPEG-1
2. Cinepak
3. H.261
4. H.263
5. Indeo32
6. Indeo41
7. MJPEGA (Motion JPEG-A)
8. MJPEGB (Motion JPEG-B)
9. MJPEG (Motion JPEG)

Media Locator:
Menemukan lokasi file dari *string* yg diberikan oleh *user*. Contoh *string*: “file:/c:/data/movie.mpg”.

Processor:
Membaca file dari *media locator*, dan menyiapkan file tersebut agar dapat dikirim oleh *Send Stream*.

Data Source:
Representasi file film yang siap dikirimkan oleh *Send Stream*. Yang dimaksud dengan representasi yang siap dikirim adalah format file yang sudah RTP *compliant*. Misalnya *frame* MPEG harus diubah menjadi format MPEG_RTP agar dapat dikirim dan dibaca oleh pihak penerima (*receiver*). Proses pengubahan dilakukan oleh *processor*, menggunakan *codec-codec* yg sesuai.

RTP Manager:
Menginisialisasi koneksi (menentukan port mana yg dibuka, IP address dan port yang menjadi tujuan pengiriman file), dan menciptakan *Send Stream* objek.

Send Stream:
Membaca *Data Source* Objek, dan mengirimkan *frame-frame* film, termasuk mengatur penggunaan *buffer* untuk mengirimkan *frame-frame* tersebut.

RTP Socket Adapter:
Mengambil *frame* per *frame* dari *data source*, *frame* ini akan dikirimkan ke *encryptor* untuk dienkripsi

Encryptor:

Mengenkripsi *frame-frame* dengan menggunakan algoritma dan kata kunci yang diberikan. Saat ini *encryptor* yg dibuat menggunakan algoritma DES. Kata kunci akan diubah menggunakan MD5 menjadi kata kunci baru dengan panjang yang tetap. Kunci yg sudah di-*hash* dengan MD5 digunakan sebagai kunci untuk DES. Encryptor akan menyimpan hasil enskripsi di *OutputStreamBuffer*.

Output Stream Buffer:
Mengirimkan semua data yg tersimpan di dalam *buffer*-nya. *Output Stream Buffer* menggunakan *Java Data Socket* untuk mengirimkan paket-paket file. Metode pengiriman menggunakan UDP.

4.2 *Receiver*

RTP Manager:
Menciptakan Receive Stream.

Receive Stream:
Menerima *RTP stream*, mendeskripsi, dan mengubah format RTP menjadi format video yang dapat dimainkan oleh *Player*.

Input Stream Buffer:
Menerima *RTP Stream*, mengatur penggunaan *buffer* untuk proses penerimaan, dan mengirimkan data yang diterima ke *Decryptor*.

Decryptor:
Mendekripsi data yg didapat menggunakan kunci yang diberikan.

RTP Socket Adaptor:
Mengubah format RTP menjadi format film yg dapat dikenali oleh *player* (Misalnya dari MPEG_RTP menjadi MPEG)

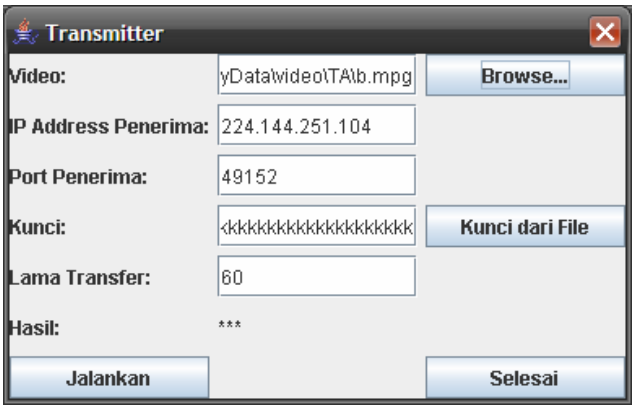
Player:
Menerima format film yg dapat dimainkan, menciptakan *controller* yg memungkinkan dimunculkannya *interface* bagi user untuk mengatur *event-event* saat film ditampilkan. *Frame* yg diterima akan diparsing dan diberikan kepada *DirectDraw codec* agar dapat ditampilkan di layar monitor.

5. IMPLEMENTASI

Berikut adalah beberapa spesifikasi kebutuhan perangkat lunak:

1. *Transmitter* dapat menerima masukan berupa file video untuk dikirim ke *receiver*.
2. *Transmitter* melakukan proses enkripsi paket data, sedangkan *receiver* melakukan proses dekripsi paket data yang diterima.
3. Kunci yang digunakan pada *transmitter* dan *receiver* harus sama. Jika kunci tidak sama, maka *receiver* tidak dapat melakukan proses dekripsi paket data, yang mengakibatkan video tidak dapat ditampilkan di *receiver*.

Lingkungan yang digunakan untuk membangun perangkat lunak dari model enkripsi video *streaming* yang telah dirancang adalah lingkungan berbasis Windows. Bahasa pemrograman yang digunakan untuk membangun perangkat lunak tersebut adalah Java versi 1.5.11. Sedangkan *framework* yang digunakan untuk membangun video *streaming* adalah JMF (*Java Media Framework*) versi 2.1.1e. Gambar 10 sampai Gambar 11 memperlihatkan antarmuka dari perangkat lunak.



Gambar 10. Implementasi Antarmuka *Transmitter*

Gambar 11. Implementasi Antarmuka *Receiver*

6. PENGUJIAN

Pengujian perangkat lunak dilakukan dengan menggunakan perangkat keras yang memiliki spesifikasi sebagai berikut:

1. Prosesor Intel T2400 (Pentium IV) @ 1,83 GHz (2CPU)
2. Memori 512 Mbytes
3. Harddisk 80 GB
4. Dibutuhkan satu buah kabel LAN untuk menghubungkan dua buah komputer.

Rancangan pengujian yang akan dilakukan meliputi pengujian terhadap tiga hal, yaitu:

1. Kebenaran perangkat lunak, yaitu *transmitter* dapat melakukan proses enkripsi video, *receiver* melakukan proses dekripsi video dan kemudian menampilkan video tersebut ke layar. Jika kunci antar *transmitter* dan *receiver* tidak sama, maka *receiver* tidak dapat mendekripsi data sehingga video tidak dapat ditampilkan di *receiver*.
2. Performansi model enkripsi video *streaming* yang menggunakan algoritma enkripsi DES dan MD5 berdasarkan perbedaan panjang kunci yang diberikan.
3. Performansi antara video *streaming* yang menggunakan algoritma enkripsi DES dan MD5 dengan video *streaming* yang tidak menggunakan enkripsi dalam *streaming* videonya.

Tabel 1 sampai Tabel 2 memperlihatkan hasil pengujian yang dilakukan.

Tabel 1 Hasil Uji Performansi Algoritma Enkripsi Video

Panjang Kunci (karakter)	Waktu rata-ratayang dibutuhkan (detik)
1	3.01
2	3.00
4	2.56
10	3.50
16	3.44
32	3.85
100	2.95
500	3.74
1000	3.94

Tabel 2 Hasil Uji Performansi Model Enkripsi Video *Streaming*

Model Video *Streaming*	Waktu yang dibutuhkan (detik)
Menggunakan Enkripsi	3.00
Tanpa Enkripsi	2.95

7. KESIMPULAN

Beberapa kesimpulan yang dapat diambil adalah:

1. Secara umum, penerapan modifikasi algoritma VEA terhadap video *streaming* tidak membebani kinerja dari video *streaming*-nya itu sendiri. Pengujian menunjukkan bahwa perbedaan waktu yang dibutuhkan antara model enkripsi video *streaming* dengan model video *streaming* yang tidak menggunakan enkripsi tidak berbeda jauh.
2. Performansi dari algoritma enkripsi video yang digunakan untuk membangun model enkripsi video *streaming*, yaitu modifikasi dari algoritma VEA, tidak terpengaruh oleh panjang kunci yang diberikan. Performansi yang dihasilkan konstan, walau dengan panjang kunci yang beragam baik dari panjang kunci yang hanya 1 karakter sampai ribuan karakter.
3. Penerapan modifikasi algoritma VEA terhadap video *streaming* memberikan banyak dampak positif, terutama karena pengiriman datanya yang menjadi lebih aman, komputasi ringan, dapat diterapkan kepada file video walaupun ukuran file video tersebut sangat besar, dan selain itu juga penerapan model enkripsi video *streaming* tidak membebani kinerja *streaming* video.

8. REFERENSI

[1] Apostolopoulos, John. Tan, Wai-tian. Wee, Susie., "*Video Streaming: Concepts, Algorithms, and Systems.*", HP Laboratories Palo Alto, 2002.

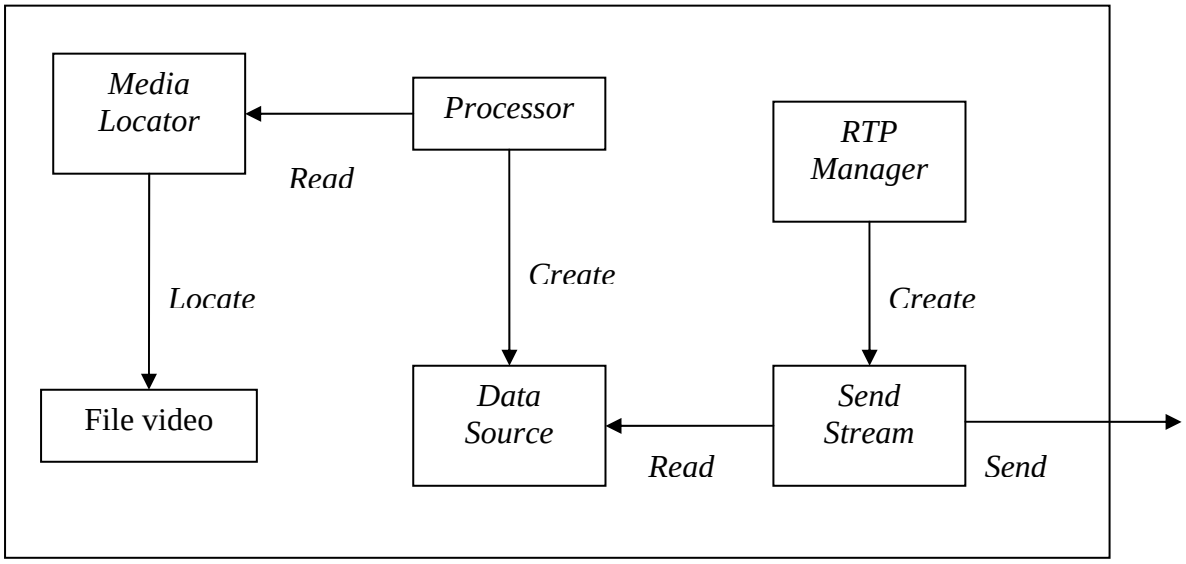
[2] Bhagarva, Bharat. Shi, Changgui. Wang, Sheng-Yih, "*MPEG Video Encryption Algorithms*", Purdue University, 2002.

[3] Apostolopoulos, John. Tan, Wai-tian. Wee, Susie., "*MPEG Video Encryption in Real-time Using Secret Key Cryptography.*", Purdue University, 1999.

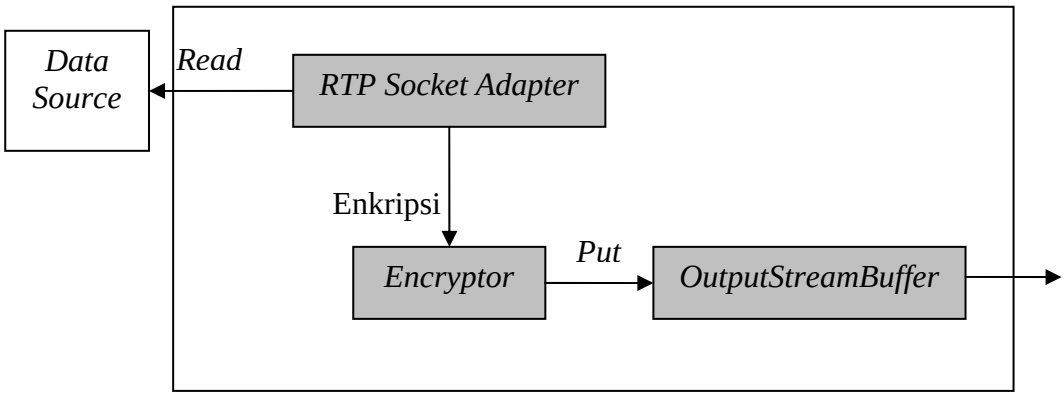
[4] Corporation, Compaq Computer, "*Video Streaming Technology*", ECG Emerging Markets and Advanced Technology, 1998.

[5] Munir, Rinaldi, M.T., "*Diktat Kuliah IF5054 Kriptografi.*", STEI ITB, 2006.

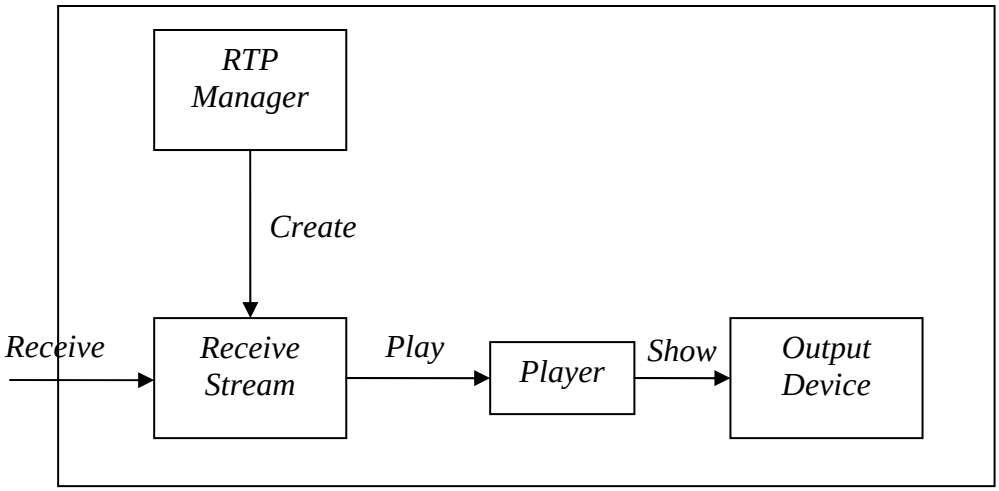
[6] Sun Microsystems,Inc, "*Java Media Framework API Guide*", Sun Microsystems.Inc, 1999.



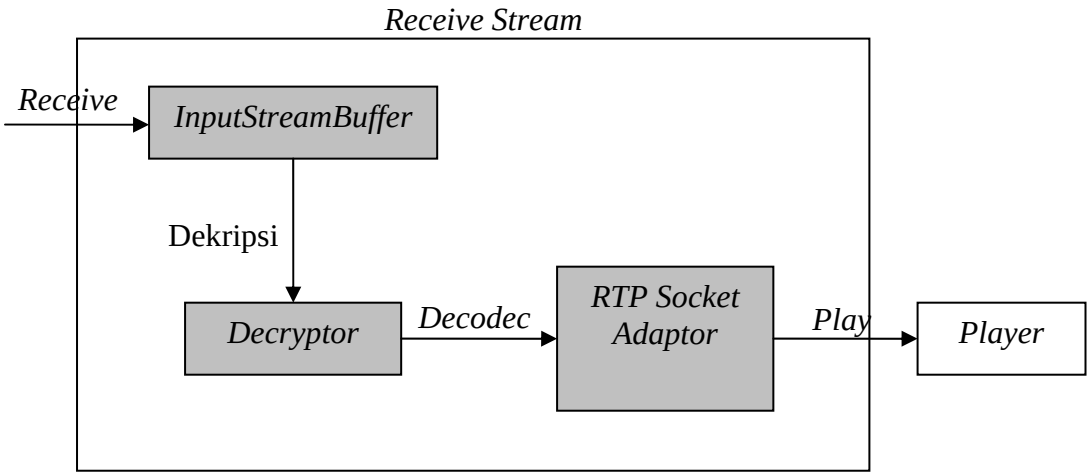
Gambar 6. Model proses yang terjadi di *Transmitter*



Gambar 7. Detail proses yang terjadi di dalam *Send Stream*



Gambar 8. Model proses yang terjadi di *Receiver*



Gambar 9. Detail proses yang terjadi di dalam *Receive Stream*