Enkripsi Selektif Video MPEG dengan Algoritma Serpent

Arief Pratama [1)]
1) Jurusan Teknik Informatika Sekolah Teknik Elektro dan Informatika ITB, Bandung, email: if14070@students.if.itb.ac.id

Abstrak *– Makalah ini membahas perancangan keamanan video dengan menerapkan enkripsi selektif. Enkripsi selektif merupakan suatu metode untuk mengenkripsi sebagian data dari video, dan data lainnya dibiarkan sebagaimana adanya. Enkripsi dilakukan dengan algoritma Serpent, karena algoritma ini memiliki kekuatan dan kecepatan yang tinggi. Algoritma Serpent termasuk blok cipher dengan ukuran blok 128 bit. Video yang akan dienkripsi adalah video dengan format MPEG-1 dan MPEG-2. Kedua format ini memiliki struktur bitstream yang hampir sama, sehingga pemrosesannya dapat dilakukan dengan satu algoritma. Data visual yang akan dienkripsi adalah frame I dan motion-vektor frame P. Keamanan video dapat dicapai dengan mengenkripsi data ini. Hasil rancangan ini kemudian dibangun ke dalam perangkat lunak dengan menggunakan platform Java. Pengujian perangkat lunak ini dilakukan dengan mengenkripsi dan mendekripsi video dengan format MPEG-1 dan MPEG-2. Hasil pengujian menunjukkan bahwa keamanan video dapat direalisasikan dengan menggunakan metode enkripsi selektif.*

Kata Kunci*: MPEG, frame, kriptografi, enkripsi selektif, Serpent.*

1. PENDAHULUAN

Pada aplikasi video, khususnya aplikasi komersil, keamanan video telah menjadi hal yang sangat penting. Misalnya, pada aplikasi *video-on-demand*, hanya pengguna yang membayar yang berhak mendapat siaran. Sama halnya dengan sistem *video conference*, sangatlah penting bahwa hanya mereka yang terdaftar pada sistem tersebut yang dapat berkomunikasi. Jika ada pihak ketiga yang ingin mengakses video tanpa otorisasi, mereka hanya akan mendapatkan video yang datanya telah terenkripsi.

Salah satu perbedaan antara data video dengan data komputer lainnya adalah ukuran yang relatif besar pada video. Data ini biasanya dikompres ke dalam suatu format, misalnya MPEG, H261, atau YUV. Walaupun demikian, video yang telah terkompres ini masih berukuran cukup besar untuk penyimpanan di harddisk ataupun untuk transmisi. Misalnya video MPEG dengan durasi 60 menit dapat berukuran 600-700 *megabyte*.

Metode yang paling *naïve* dalam mengamankan data video adalah dengan mengenkripsi keseluruhan *bitstream* pada video dengan algoritma yang kuat seperti AES [6]. Metode ini akan menghasilkan tingkat keamanan yang paling tinggi dalam penyimpanan ataupun transmisi video. Walaupun demikian, performansinya sangat rendah. Pemrosesan yang besar sangat dibutuhkan dan hal ini dapat mengganggu aspek *realtime* jika diaplikasikan dalam transmisi video. Selain itu, video hasil enkripsi dengan metode *naïve* tidak akan bisa dijalankan dengan *decoder* standar, karena struktur *bitstream*-nya akan rusak setelah dienkripsi.

Data multimedia umumnya bernilai lebih rendah daripada data-data dijital lainnya (seperti informasi bank dan dokumen rahasia perusahaan). Implementasi keamanan video bisa saja bernilai lebih mahal daripada nilai video itu sendiri. Pihak penyedia layanan hanya akan memboros uang untuk keperluan keamanan video yang tidak begitu penting.

Untuk mengatasi hal ini, metode enkripsi selektif digunakan. Enkripsi selektif (*selective encryption*) merupakan sebuah teknik untuk mengenkripsi sebagian porsi dari data video dan data lainnya dibiarkan sebagaimana adanya [6]. Metode ini memanfaatkan informasi mengenai struktur *bitstream* pada video dalam melakukan enkripsi. Dengan adanya informasi struktur data ini, *bitstream* hasil enkripsi dapat berupa video dengan format yang sama dengan video aslinya. Pada enkripsi selektif, pemrosesan CPU lebih kecil dan enkripsi dapat dilakukan dalam waktu singkat. Pada transmisi video, enkripsi selektif sangatlah berguna agar aspek *realtime* terpenuhi. Tingkat keamanan yang dihasilkan metode ini lebih rendah daripada metode *naïve*, namun performansi yang lebih tinggi dihasilkan.

Pada enkripsi selektif, algoritma *cipher* apapun dapat digunakan. Salah satunya adalah algoritma Serpent. Serpent adalah *cipher* dengan ukuran blok 128 bit yang dibuat oleh Ross Anderson, Eli Biham, dan Lars Knudsen sebagai salah satu kandidat AES. Pada mesin Pentium 200MHz, algoritma Serpent dapat berjalan dengan kecepatan 45 Mbit/s. Ini lebih cepat daripada DES yang memiliki kecepatan

15 Mbit/s pada mesin yang sama. Algoritma Serpent tidak dipatenkan, sehingga dapat digunakan oleh publik untuk keperluan apapun. [4]

Algoritma Serpent dipilih karena dua alasan. Pertama, algoritma ini merupakan salah satu *cipher* terkuat yang pernah ada, sehingga faktor keamanan video dapat diserahkan sepenuhnya kepada algoritma ini. Kedua, algortima Serpent memiliki performansi yang tinggi, sehingga proses enkripsi/dekripsi tidak akan mempengaruhi pemrosesan (*encoding/decoding*) video.

Pada video MPEG, enkripsi selektif dilakukan pada *frame* agar data hasil enkripsinya masih dapat di-*playback* dengan *decoder* standar. Secara umum, ada tiga jenis *frame* pada MPEG, yaitu *frame* I, P, dan B. *Frame* I adalah *frame* yang utuh di mana semua datanya disimpan secara keseluruhan tanpa ada referensi ke *frame* lain. *Frame* I digunakan oleh *frame* P dan B sebagai referensi dalam proses *motion-compensation*.

Motion-compensation adalah proses dalam merekonstruksi suatu *frame* dengan menggunakan *motion vector*. *Motion vector* dapat didefinisikan sebagai perbedaan posisi suatu objek dalam suatu *frame* dengan posisi objek tersebut pada *frame* yang lain [3].

Gambar di bawah menunjukkan gambaran umum proses *motion-compensation* suatu *frame* P. Pada *frame n*, terdapat suatu objek pada posisi kiri atas. Pada *frame n+i*, posisi objek tersebut pindah ke kanan bawah. Di sinilah kompresi temporal terjadi. *Frame n+i* tidak perlu menyimpan data *pixel* objek tersebut. Yang perlu disimpan hanyalah perpindahan posisi objek tersebut dari *frame n*.

Gambar 1 Motion compensation

2. PERSOALAN KEAMANAN VIDEO

Beberapa kasus berikut dapat menjelaskan mengapa keamanan data video begitu penting pada saat ini:

1. Konferensi video.
   Pada sistem komunikasi saat ini konferensi video sangat sering digunakan, misalnya *videophone*, *webcam*, dan sebagainya. Apapun jenis teknologi yang digunakan, jenis data yang terkirim dapat bermacam-macam dan pihak *eavesdropper* dapat dengan sengaja atau tidak sengaja mengetahui data ini.
2. Kamera pengawasan (*surveillance*).
   Sistem kamera pengawasan pada saat ini dapat ditemukan hampir di semua tempat umum yang dikunjungi oleh banyak orang. Hal ini untuk menghindari tindakan kriminal atau teroris yang dapat membahayakan publik. Video yang dihasilkan kamera ini tentu saja hanya bisa dilihat oleh pihak-pihak tertentu.
3. DVD.
   Proteksi pada DVD dapat dilakukan dengan menggunakan konsep *trusted hardware*: DVD hanya bisa dijalankan dengan hardware khusus. Konsep ini diimplementasikan dengan enkripsi, yaitu hanya *player* atau *recorder* yang terdaftar yang memiliki informasi kunci dan algoritma untuk mendekripsi data video dalam DVD.

Solusi terbaik untuk menangani persoalan-persoalan seperti di atas adalah dengan menggunakan kriptografi. Pendekatan klasik kriptografi untuk menangani persoalan keamanan adalah dengan memilih *cipher* yang terbukti aman dan kuat untuk mengenkripsi data, tidak peduli datanya apa, atau aplikasinya di mana. Ada banyak teknik enkripsi yang dapat dipilih seperti *stream cipher*, *block cipher*, dan algoritma kunci publik. Pilihan algoritma untuk teknik enkripsi tersebut juga banyak, misalnya Serpent, DES, RC4, RSA, dan lain-lain.

Ada beberapa persyaratan yang dibutuhkan dalam memberikan kemanan pada video yaitu tersebut mencakup keamanan, kecepatan, dan *bitstream compliance* [6].

1. Keamanan

Keamanan adalah syarat yang paling utama. Tingkat keamanan ini dapat dicapai oleh *cipher* yang digunakan untuk mengenkripsi video. Semakin kuat *cipher*, maka tingkat keamanan akan semakin tinggi.

2. Kecepatan

Ukuran suatu video umumnya jauh lebih besar daripada data-data yang lain. Faktor ukuran ini berperan besar dalam mempengaruhi kecepatan proses enkripsi video. Pada kasus konferensi video, aspek realtime dapat dipengaruhi jika proses enkripsi mengakibatkan delay yang tinggi. Pada kasus *surveillance*, aktivitas seorang petugas keamanan tidak boleh terpengaruhi oleh *delay* transmisi video yang tinggi. Sedangkan pada kasus DVD, proses dekripsi video tidak boleh mempengaruhi *frame rate*.

Dalam hal kecepatan, ada dua isu yang sering menjadi bahan perhatian, yaitu daya energi dan

memori. Jika target aplikasi enkripsi ini adalah perangkat mobile (*laptop* atau *smartphone*), maka proses enkripsi harus cepat agar tidak memakan daya yang terlalu banyak, yang konsekuensinya dapat mengganggu aktivitas pengguna. Jika enkripsi video ini akan diimplementasikan pada hardware, penggunaan memori harus sehemat mungkin agar biayanya dapat diminimalisir.

3. Bitstream compliance

Enkripsi dengan metode *naïve* suatu data video yang memiliki format (MPEG-1 atau MPEG-2) akan menghasilkan data yang tidak ada hubungannya dengan format aslinya. *Bitstream* yang dihasilkan hanyalah kumpulan *byte-byte* tidak terstruktur. Player MPEG tentu saja tidak akan bisa menjalankan video ini. Program tersebut akan langsung *crash* atau tidak bisa menampilkan apapun karena *decoder*-nya tidak dapat mengidentifikasi informasi *header*.

Dari sisi keamanan, hal ini tentu saja dapat diterima. Namun, jika suatu player tidak dapat menjalankan video yang terenkripsi, maka hal ini tidak ada lagi hubungannya dengan keamanan data visual video. Teknik seperti ini hanya akan menyediakan keamanan bagi konsumen yang tidak memiliki pengetahuan mengenai video karena seorang penyerang (attacker) tetap akan mencari cara untuk memecahkan enkripsinya. Keamanan seperti ini disebut dengan *security by obscurity*—kemanan yang dicapai karena datanya tidak jelas.

Untuk memastikan bahwa tingkat keamanan benar-benar ditekankan pada data visual video, maka video hasil enkripsi harus dapat dijalankan dengan *decoder* standar. Ini dapat dicapai dengan membiarkan *header* data tidak berubah, dan enkripsi hanya dilakukan pada bagian *bitstream* yang merepresentasikan data.

3. SOLUSI KEAMANAN VIDEO DENGAN ENKRIPSI SELEKTIF

Permasalahan yang dijabarkan di atas dapat ditangani dengan menerapkan enkripsi selektif dengan menggunakan *cipher* yang kuat, yaitu Serpent.

Berikut ini dijelaskan bagaimana solusi ini dapat memenuhi ketiga persyaratan di atas:

1. Keamanan

Dengan memilih Serpent sebagai algoritma enkripsi, maka faktor keamanan terjamin dengan sendirinya karena Serpent telah terbukti keampuhannya. Faktor keamanan yang juga dipengaruhi oleh jumlah *frame* I dalam video. Semakin banyak *frame* I, maka tingkat keamanan akan lebih tinggi.

2. Kecepatan

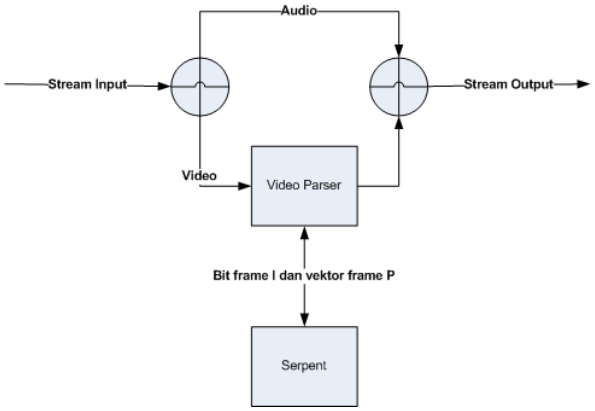
Dengan hanya mengenkripsi sebagian porsi dari video, maka kecepatan yang diraih akan lebih tinggi dan penggunaan *resource* akan lebih rendah.

3. Bitstream compliance

Pada enkripsi selektif MPEG, *header* video dibiarkan sebagaimana adanya, sehingga *decoder* standar akan dapat mengenali *bitstream* hasil enkripsi.

Sebelumnya dijelaskan bahwa MPEG memiliki tiga jenis *frame*, yaitu I, P, dan B. Data yang akan dienkripsi adalah semua *frame* I dan *vektor motion-compensation frame* P. Dengan adanya ketergantungan *frame* P pada I dan ketergantungan *frame* B pada I dan P, maka data-data pada *frame* P dan B akan terlindungi sekaligus, dengan demikian keseluruhan video dapat terlindungi.

Proses enkripsi selektif dapat dimodelkan dengan gambar berikut:



Gambar 2 Model enkripsi selektif pada MPEG

Pada gambar di atas, *stream* input akan dipisahkan menjadi *stream* audio dan video. *Stream* video akan diproses oleh video *parser* yang akan mengambil bit-bit yang merepresentasikan *frame* I dan vektor *frame* P.

Bit-bit *frame* I merupakan semua data yang ditemukan setelah header *frame* I ditemukan. *Header* di sini adalah *header layer picture* dan *layer slice*, sehingga data yang dapat dienkripsi adalah data pada *layer* makroblok (termasuk di dalamnya *layer* blok).

Pada umumnya, sebuah *group of pictures* (GOP) memiliki 12 sampai 15 *frame*, dan di antara *frame* tersebut terdapat satu *frame* I. Dengan demikian,

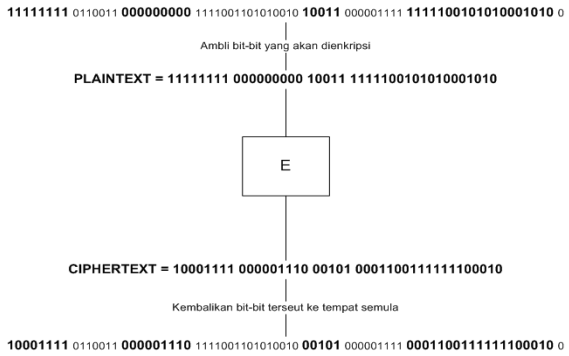
jika diasumsikan suatu video memiliki 12 frame untuk setiap GOP, maka jumlah *frame* I adalah 1/12 atau 8,33% dari jumlah keseluruhan *frame*.

Untuk vektor frame P, jumlah data yang dienkripsi adalah 4 bit untuk setiap *frame* P. Data ini bukanlah makroblok *frame* P. Data vektor ini nantinya akan digunakan oleh *decoder* sebagai referensi untuk mengambil data dari *frame* I sebelumnya.

Ukuran bit-bit pada setiap *frame* I tidak sama. Semua tergantung kepada jumlah *slice*, dan jumlah makroblok yang dimiliki oleh setiap *slice* tersebut. Setiap makroblok pun ukurannya berbeda. Karena itu, algoritma Serpent dapat digunakan sebagai berikut:

1. Menggabungkan data-data tersebut untuk membentuk blok-blok plainteks berukuran 128 bit.
2. Setiap blok terebut dienkripsi dengan Serpent untuk menghasilkan cipherteks.
3. Cipherteks ini dibagi-bagi kembali sesuai dengan posisi aslinya pada stream video.
4. Kirimkan stream tersebut ke *multiplexer* yang akan menggabungkan kembali *stream* video dan audio.

Gambar berikut dapat mendeskripsikan bagaimana algoritma Serpent diaplikasikan pada model enkripsi selektif ini. Pada gambar di bawah ini, bit-bit yang ditebalkan adalah bit-bit data yang harus dienkripsi.

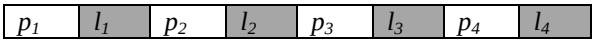


Gambar 3 Penggunaan Serpent pada enkripsi selektif

Untuk enkripsi blok terakhir, jika ukuran bitnya tidak cukup 128 bit, maka data tersebut akan diabaikan. Dengan demikian, ukuran file hasil enkripsi akan sama dengan file aslinya.

Proses enkripsi dan dekripsi diimplementasikan dengan menggunakan algoritma yang sama dalam penyeleksian data yang akan diproses (enkripsi/dekripsi). Perbedaanya terletak pada penanganan bit-bit yang akan diproses tersebut.

Untuk menjelaskan algoritma yang digunakan, sebuah video MPEG dapat dimodelkan sebagai berikut:



Gambar 4 Gambaran umum video yang akan dienkripsi

p_i adalah data yang tidak dienkripsi/didekripsi
l_i adalah data yang akan dienkripsi/didekripsi

Dengan menggunakan gambar di atas, maka algoritma berikut dapat menjelaskan proses enkripsi dan dekripsi.

```
p = 0;
l = 0
repeat
    while size<128 do
        read(p,l);
        bufferp[p++] := p;
        bufferl[l++] := l;
        size := size + length(l);
    {size >= 128 atau semua pi dan li telah
    dibaca}


    {gabungkan isi bufferl}
    merged := bufferl[0];
    for i:=1 to length(bufferl)-1 do
        merged:= mergearray(merged, bufferl[i]);

    while size>=128 do
        n := 1;
        Enksripsi/dekripsi blok ke-n dalam array
        merged;
        n := n + 1;
        size := size - 128;
    {data bufferl yang tidak dienkripsi ada
    sebanyak 'size'}

    Kembalikan semua isi merged ke bufferl
    kecuali data yang tidak dienkripsi/didekripsi

    sizetowrite = length(bufferl);
    if size>0 then
        bufferl[p++]   :=   data   yang   tidak
        dienkripsi/didekripsi
        bufferp[l++] := buffer kosong

        for i:=0 to sizetowrite do
            write(bufferp[i]);
            write(bufferl[i]);
            p := p - 1;
            l := l -1;

until semua pi dan li telah dibaca
```

Fungsi *read(p,l)* membaca *p* dan *l* dari file. Proses pembacaan ini menggunakan referensi dari dokumen standar MPEG-1 (ISO/IEC 11172-2) dan MPEG-2 (ISO/IEC 13818-2) untuk mengidentifikasi *p* dan *l* tersebut.

Karena algoritma yang sama digunakan penyeleksian bit-bit yang akan dienkripsi dan didekripsi, dan bahwa proses enkripsi dan dekripsi pada Serpent menggunakan algoritma yang sama, maka kedua proses ini memiliki kompleksitas algoritma yang sama.

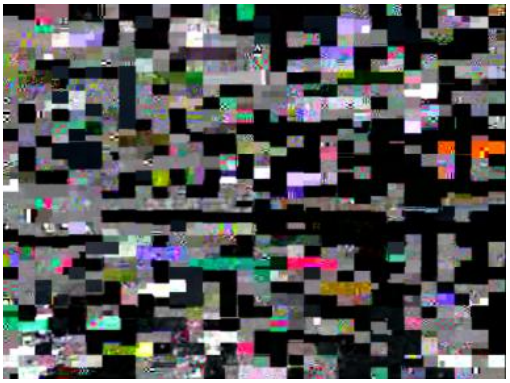
4. PENGUJIAN

Dalam pelaksanaan pengujian, digunakan tiga sample video yang di-*download* dari

http://www.movie-list.com. Dengan menggunakan *Total Video Converter*, ketiga video ini dikonversi ke dalam format MPEG-1 dan MPEG-2. Hasil pengujian ini dapat dilihat pada Tabel 1 dan Tabel 2. MPEG-1 dan MPEG-2 memiliki ekstensi file yang sama (*.mpg).

Gambar 5 Frame asli

Gambar 6 Frame hasil enkripsi (MPEG-1)



Gambar 7 Frame hasil enkripsi (MPEG-2)

Gambar di atas merupakan tampilan video hasil pengujian. Dapat dilihat bahwa *frame* video hasil enkripsi tidak dapat dikenali lagi.

Tabel 1 Hasil Enkripsi MPEG-1

Nama File	video1.mpg	video2.mpg	video3.mpg
Ukuran File (byte)	14159872	13336576	11755520
Ukuran total data yang dienkripsi (byte)	317465	311358	285715
Rasio	2,24%	2,33%	2,43%
Waktu (detik)	2,12	1,90	1,73
Bitrate (Mbps)	53,48	56,04	54,27

Tabel 2 Hasil Enkripsi MPEG-2

Nama File	video1.mpg	video2.mpg	video3.mpg
Ukuran File (byte)	38834040	38232124	27551020
Ukuran total data yang dienkripsi (byte)	4127073	4406633	2325178
Rasio	10,63%	11,53%	8,44%
Waktu (detik)	6,02	6,39	4,07
Bitrate (Mbps)	51,58	47,88	54,18

Pada kedua tabel ini, *rasio* adalah perbandingan jumlah data yang dienkripsi dengan ukuran file. *Waktu* merupakan lama durasi proses enkripsi. *Bitrate* merupakan jumlah data input yang dapat diproses dalam satu detik. Proses di sini meliputi pembacaan file input, penyeleksian bit-bit yang akan dienkripsi, enkripsi bit-bit tersebut, dan penulisan hasilnya ke file output.

$$Rasio = \frac{Data\ yang\ dienkripsi}{Ukuran\ File}$$

$$Bitrate = \frac{(Ukuran\ File \times 8)}{Waktu}$$

Berdasarkan tabel di atas, maka hasil pengujian dapat disimpulkan ke dalam poin-poin berikut:

1. Semua modul perangkat lunak berjalan dengan baik.
2. Rata-rata jumlah data yang dienkripsi hanya 2,33% untuk MPEG-1 dari ukuran file dan untuk MPEG-2 sebanyak 10,2%. Hal ini menunjukkan tingginya efisiensi yang diberikan oleh metode enkripsi selektif terutama untuk MPEG-1.
3. Ukuran *frame* I MPEG-2 jauh lebih besar dari MPEG-1.
4. Proses enkripsi dapat berjalan dalam waktu yang cepat.
   Pada tabel di atas dapat dilihat bahwa rata-raa *bitrate* berada di atas 50 Mbps. Dengan demikian enkripsi selektif tidak akan memberikan delay yang berarti bagi pemrosesan video (*playback* atau transmisi) di mana *bitrate*-nya hanya 1.4 Mbps untuk MPEG-1 atau 3-5 Mbps untuk MPEG-2.

5. Pada gambar video hasil enkripsi, dapat dilihat bahwa gambar hasil enkripsi tidak dapat dikenali lagi. Jika diasumsikan ada pihak penyerang yang mengetahui algoritma yang digunakan untuk penyeleksian data yang akan dienkripsi, maka tingkat keamanan sepenuhnya bergantung pada algoritma Serpent.

Dengan demikian enkripsi selektif dapat memberikan tingkat keamanan yang cukup bagi video.

4. KESIMPULAN

1. Enkripsi selektif dapat diterapkan pada video MPEG-1 dan MPEG-2. Pada MPEG-1 dan MPEG-2, kompresi temporal menyebabkan adanya 3 jenis *frame*, yaitu *frame* I, *frame* P, dan *frame* B. Dengan adanya ketergantungan *frame* P dan B pada *frame* I, maka enkripsi tidak perlu dilakukan pada semua *frame*.
2. Cipher apapun dapat digunakan dalam enkripsi selektif, baik itu stream cipher ataupun block cipher. Implementasi enkripsi selektif akan lebih mudah jika menggunakan stream cipher daripada blok cipher. Blok cipher memerlukan data yang berukuran tetap (untuk Serpent sebanyak 128 bit), sedangkan data itu sendiri tersebar pada stream dengan ukuran yang berbeda-beda. Dengan demikian, diperlukanlah proses penggabungan data-data tersebut sebelum enkripsi dilakukan. Hal ini dapat mengakibatkan meningkatnya kompleksitas algoritma.
3. Dengan enkripsi selektif, jumlah data yang perlu dienkripsi cukup kecil.
4. Video hasil enkripsi memiliki format yang sama dengan video aslinya. Dengan demikian video hasil enkripsi ini dapat dijalankan dengan decoder standar, namun data-data visualnya tidak dapat dikenali lagi.
5. Jika diasumsikan penyerang (attacker) mengetahui algoritma yang digunakan untuk menyeleksi data yang akan dienkripsi, maka faktor keamanan sepenuhnya bergantung kepada algoritma Serpent.

DAFTAR REFERENSI

[1] Ghanbari, Mohammed. *Standard Codecs: Image Compression to Advanced Video Coding*. The Institution of Electrical Engineers, London, United Kingdom. 2003

[2] Munir, Rinaldi. *Diktat Kuliah IF5054 Kriptografi*. Program Studi Informatika STEI-ITB, 2006

[3] Richardson, Iain E. G. *Video Codec Design*. John Wiley & Sons Ltd. 2002

[4] Andreas, Ross. Biham, Eli. Knudsen, Lars. *Serpent: A Proposal for the Advanced Encryption Standard*. Cardis, 1998

[5] *http://www.cl.cam.ac.uk/~rja14/serpent.html*, diakses 17 Desember 2008 11.04 WIB

[6] Uhl, Andreas dan Andreas Pommer. Image and *Video Encryption From Digital Rights Management to Secured Personal Communication*. Springer. 2005