Algoritma Enkripsi pada Video MPEG

Marvello Oni (13508031)

Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
e-mail: Marvello_Oni@Yahoo.com

ABSTRAK

Keamanan data multimedia sangat penting dalam melakukan kegiatan jual dan beli. Perkembangan awal kriptografi dipusatkan pada data berbentuk tulisan. Algoritma yang digunakan untuk itu mungkin tidak sesuai untuk file multimedia yang mempunyai ukuran yang besar. Untuk itu diperlukannya algoritma lain yang ringan dan aman.

Dalam makalah ini akan dibahas mengenai algoritma yang digunakan untuk mengenkripsi file MPEG sebagai salah satu format file multi media. Terdapat beberapa algortima yang diajukan untuk melakukan enkripsi untuk tipe file ini. Dalam makalah ini akan dibahas lima buah algoritma yaitu *naïve algorithm*, alogritma seleksi (*Selective Algorithm*), Algoritma Permutasi Zig-Zag (*Zig-Zag Permutation Algorithm*), dan Algoritma Enkripsi Video (Video Encryption Algorithm), dengan memperhatikan kecepatan enkripsi, tingkat keamanan dan kecepatan transmisi. Dengan demikian dapat dilihat bahwa penggunaan algoritma bergantung pada keperluan tingkat keamanan dari aplikasi multimedia.

Kata kunci: Enkripsi, Algoritma, multimedia.

1. PENDAHULUAN

Dewasa ini file multimedia sudah menjadi bagian dari komunikasi dan sangat penting dalam berbagai aspek kehidupan manusia. Untuk memperoleh manfaat sepenuhnya dari file multimedia diperlukan tingkat keamaanan yang dapat dipercaya terutama dalam kasus *video conference* dan *video broadcasting* dimana masalah perlindungan hak milik dan hak cipta sangat diperlukan. MPEG (IS0 Moving Picture Expert Group) adalah salah satu ekstensi file multimedia yang paling sering digunakan sehingga masalah enkripsi file MPEG menjadi isu yang penting.

Terdapat dua tantangan besar dalam enkripsi file multimedia. Pertama adalah besar ukuran file yang cukup besar (sebagai contoh, ukuran file MPEG yang berdurasi 2 jam mempunyai ukuran sekitar 1 GB). Kedua file multimedia perlu diproses secara *real-time* (sebagai contoh untuk file MPEG dengan *High-Definition* mempunyai *data rates* sekitar 45 Mbps atau lebih). Memproses data besar seperti ini dalam waktu yang sangat lama akan menjadi beban pada *codec*, memori, sarana penyimpanan data dan komunikasi pada jaringan. Karena itu algoritma harus dibuat sedemikian rupa sehingga mempunyai efisiensi setinggi mungkin dan tingkat atau rasio kompresi yang tinggi.

Solusi pertama yang dikenalkan adalah enkripsi data multi media menggunakan algoritma kriptografi dengan kode rahasia (*secret key cryptography algorithms*) seperti DES (Data Encryption Standard) or IDEA (International Data Encryption Algorithm). Tetapi algortima ini melibatkan kompitasi yang rumit dan memakan memori yang besar karena ukuran data multimedia yang dikomputasi cukup besar. Kemudian dikenalkanlah beberapa algoritma enkripsi yang dapat digunakan untuk file multimedia. Algoritma-algoritma ini walaupun bervariasi dalam pertahanan terhadap serangan mempunyai kesamaan yaitu menambahkan sebuah kode khusus untuk MPEG codec. Software-software dipasaran yang mengimplementasikan algoritma ini cukup cepat untuk memenuhi standar kehidupan nyata untuk aplikasi video MPEG. Dalam makalah ini kami hanya akan membahas *naïve algorithm*, alogritma seleksi (*Selective Algorithm*), Algoritma Permutasi Zig-Zag (*Zig-Zag Permutation Algorithm*), dan Algoritma Enkripsi Video (Video Encryption Algorithm, mengevaluasi dan membadingkannya.

2. Algoritma Enkripsi MPEG

2.1 *Naïve Algorithm*

Cara ini disebut dengan pendekatan *Naïve algorithm*[2]. *Naïve algorithm* memperlakukan baris bit file MPEG seperti hanya data teks tradisional dan tidak menggunakan sedikit pun bentuk special dari struktur file MPEG.

Tetapi cara ini dianggap tidak memuaskan karena tidak dapat memenuhi standar kecepatan untuk enkripsi video dalam waktu yang nyata.

2.2. Algoritma Seleksi (*Selective Algorithm*)

Terdapat beberapa sumber yang menggunakan fasilitas struktur berlapis MPEG [3,4]. Algoritma ini semuanya dapat digolongkan pada katagori Algoritma Seleksi (*Selective Algorithm*) .

Dasar dari Algoritma seleksi adalah berdasarka pada struktur frame IPB pada file MPEG. Algoritma ini hanya mengenkripsi frame **I** saja karena secara konseptual, frame P dan B menjadi tidak berguna bila kita tidak mengetahui frame I yang berkorespodensi.

Tetapi, Agi and Gong [2] telah menunjukan bahwa sebagian besar dari video masih dapat diihat karena adanya korelasi antar frame dan sebagian besar berasal dari blok-I yang tidak terenkripsi pada frame P dan B. Karena itulah hanya mengenkripsi frame I saja tidak menghasilkan tingkat keamanan yang memuaskan. Meyer dan Gadegast [5] telah mendesign sebuah baris bit yang menyerupasi MPEG bernama SECMPEG, yang menggunakan enkripsi seleksi dan informasi tambahan pada *header*, dan mempunyai waktu eksekusi software yang cepat.

SECMPEG dapat menggunakan algoritma enkripsi standard seperti DES dan RSA dan menimplementasikan empat tingkat keamanan. Tingkat pertama adalah dengan mengenkripsi semua header. Tingkat kedua adalah mengenkripsi sebuah header ditambah dengan koefisien DC dan bagian bawah dari AC pada blok I. Tingkat ketiga adalah dengan mengenkripsi sebuah frame I dan semua blok I pada frame P dan B. Tingkat keempat adalah mengenkripsi semua data. SECMPEG tidak cocok atau sesuai dengan MPEG standar. Sebuah *Encoder* khusus diperlukan untuk melihat sebuah baris data SECMPEG yang tidak dienkripsi.

2.3. Algoritma Permutasi Zig-Zag (*Zig-Zag Permutation Algorithm*)

Pada bukunya L. Tang [1] mengusulkan pendekatan untuk mengenkripsi arus MPEG dengan mengunakan permutasi ZigZag. Ide dasarnya adalah daripada memetakan blok 8x8 kedalah vector 1x64 dalam urutan yang “zig-zag”, lebih baik menggunakan sebuah daftar permutasi yang acak untuk memetakan sebuah blok 8x8 yang individual kedalam vector 1x64[1].

Algoritma Permutasi ZigZag terdiri atas tiga langkah :

1. Membuat sebuah daftar permutasi dengan jumlah bilangan 64. Hal ini dapat dilakukan diluar algoritma.
2. Selesaikan prosedur pembagian setelah blok 8x8 selesai diquatisasi. Anggap koefisien DC dapat dimislkan dengan 8 digit bilangan biner $d_7d_6d_5d_4d_3d_2d_1d_0$. Kemudian bagi bilangan tersebut menjadi 2 buah bagian yaitu $d_7d_6d_5d_4$ dan $d_3d_2d_1d_0$, yang mana keduanya berada dalam rentang [0..15] dalam decimal. Kemudian koefisien DC diubah menjadi $d_7d_6d_5d_4$ dan akhir koefisien AC diubah menjadi $d_3d_2d_1d_0$. Prosedur pembagian ini didasari atas obeservasi berikut: 1) biasanya, nilai dari koefisien DC lebih besar daripada nilai dari koefisien AC, karena itu koefisien DC dapat dengan mudah dikenali bahkan setelah permutasi. Dengan membagi koefisien DC menjadi 2 buah bilangan yang lebih kecil, akan lebih sulit untuk membedakannya dari koefisien AC. 2) setelah dibagi, sebuah ruang ekstra diperlukan untuk menyimpan bilangan yang dibagi, sebagai contoh $d_3d_2d_1d_0$. Ini akan menambah panjang arus MPEG. Tetapi harus diperhatikan bahwa akhir koefisien AC adalah bilangan dengan tingkat signifikan paling kecil, yang bisa diubah menjadi 0 tanpa degradasi visual yang signifikan. Jadi tempat ini bisa digunakan untuk menyimpan $d_3d_2d_1d_0$.
3. Kemudian masukan daftar permutasi acak kedalam blok yang telah terbagi.

Ada dua buah metode tambahan untuk mencoba meningkatkan algortima permutasi zig-zag. Metode pertama adalah dengan mengelompokkan koefisien DC dari tiap 8 blok menjadi satu dan mengaplikasikan DES pada tiap kelompok. Kemudian prosedur pembagian dan permutasi dilakukan pada tiap blok. Kedelapan koefisien DC dapat dikelompokan dengan urutan sekuensial atau acak. Metode kedua adalah dengan menggunakan urutan lemparan koin biner bersama dengan dua buah daftar permutasi. Cara ini membuat metode dasar lebih aman terhadap serangan *known-plaintext*. Pembuat disarankan untuk mengikuti tambahan berikut: 1) dua buah daftar permutasi dibuat dan 2) untuk tiap blok 8x8, sebuah koin dilempar. Bila ekor maka daftar permutasi satu dimasukan; bila kepala maka daftar permutasi dua dimasukan kedalam blok. Daftar biner lemparan koin bersama dengan kedua daftar permutasi adalah kunci rahasianya.

2.4. Algoritma Enkripsi Video (*Video Encryption Algorithm*)

Arus MPEG berbeda dengan data tekstual tradisional karena ia mempunyai tipe data yang khusus dan ia dikompresi. Perhatikan bahwa kesamaan dari kompresi dan enkripsi adalah keduanya mencoba membuang informasi yang kurang bermanfaat. Karena itu pembelajaran sturktur MPEG dan sifat statistiknya membawa kepada Algoritma Enkripsi Video.

MPEG adalah algoritma yang membuang informasi yang tidak dibutuhkan dari sekuens gambar. Ini berarti ia mempunyai distribusi nilai *byte* yang lebih merata dan ia berbeda dengan data tekstual. Dalam VEA proses dilakukan *byte* ke *byte* dikarenakan : 1) lebih mudah untuk memproses data dengan pengetahuan tentang *byte;* 2) sebuah *byte* tunggal tidak berarti dalam arus video karena biasanya isi video dikodekan dalam beberapa *byte*. Ini

berbeda dengan data teks yang tiap *byte*-nya mempunyai arti sendiri; 3) tingkat keacakan dikenalkan dalam tingkat *byte* karena kode panjang variable Huffman digunakan dalam Algoritma Kompresi MPEG. Perhatikan bahwa dalam pembagian arus *bit* MPEG kedalam arus *byte*, setiap unit mempunyai nilai bilangan bulat antara 0 dan 255.[6]

Tahapan dalam algoritma ini adalah:

(i) Anggap data dari frame I adalah dalam bentuk sebagai berikut $a_1a_2a_3a_4..a_{2n-1}a_{2n}$

(ii) Pilih bytes dengan nomor ganjil dan nomor genap untuk membentuk dua buah arus bytes yang baru. Kita sebuat daftar ganjil dan daftar genap.

(iii) Lakukan operasi XOR pada kedua buah arus

$$\begin{array}{lcccc} & a_1 & a_3 & \dots & a_{2n-1} \\ \text{Xor} & a_2 & a_4 & \dots & a_{2n} \\ \hline & c_1 & c_2 & \dots & c_n \end{array}$$

(iv) Pilih fungsi enkripsi E (contoh : DES) untuk mengenkripsi $a_2a_4\dots a_{2n}$. Hasil c*ipher* dari enkripsi tersebut dapat ditulis sebagai $c_1c_2\dots c_nE(a_2a_4\dots a_{2n})$.

Sangat mudah menunjukan bahwa , kalau $a_2a_4\dots a_{2n}$ tidak mempunyai pola yang berulang, maka tingkat keamanan bergantung pada fungsi E karena $a_2a_4\dots a_{2n}$ adalah *one-time pad* yang diketahui sangat aman.

3. Analisis Tingkat Keamanan

3.1 *Naïve Algorithm*

Tidak dapat diragukan bahwa ini adalah algoritma yang mempunyai tingkat keamanan paling tinggi. Ini dikarenakan tidak ada yang algoritma efektif yang dapat digunakan untuk menjebol AES (Advance Encryption Standard) apalagi apabila menggunakan kunci yang cukup panjang seperti 256 bit.

3.2. Algoritma Seleksi (*Selective Algorithm*)

Algoritma ini tidak menghasilkan tingkat keamanan yang memuaskan. Agi and Gong [2] telah menunjukan bahwa video yang dienkripsi masih dapat terlihat sebagian adanya korelasi antar frame dan adanya bagian blok I yang di terenkripsi pada frame P dan B. Karena itulah kita harus mengenkripsi keseluruhan blok I dalam seluruh arus MPEG untuk memperoleh tingkat keamanan yang paling baik. Ide ini di realisasikan pasa SECMPEG.

Agi and Gong [2] juga menyarankan untuk mengganti frekuensi dari frame I. Dalam kasus yang ekstreme adalah dengan meng-*encode* keseluruhan frame menjadi frame I atau animasi JPEG. Ini setara dengan *Naïve Algorithm* yang menghasilkan tingkat keamanan yang paling baik.

Perhatikan bahwa menyembunyikan informasi bagian kepala dari struktur MPEG bukanlah cara yang baik. Satu alas an adalah karena bagian kepala menyimpan informasi paling standar seperti kode awal frame, ukuran frame, jenis frame, dll. Seorang *attacker* bisa dengan mudah menebak informasi seperti ini. Alasan yang lain adalah dalam sebagaian besar sistem VOD, arus MPEG di index berdasarkan frame dengan tujuan untuk melakukan sinkronisasi atau adaptasi. Karena itulah bagian awal dari setiap frame bisa diketaui.

3.3. Algoritma Permutasi Zig-Zag (*Zig-Zag Permutation Algorithm*)

Terdapat masalah yang serius dalam tingkat keamanan pada algoritma ini.

Seperti yang ditunjukan diatas bahwa algoritma ini pada dasarnya tidak bisa bertahana terhadap *known-plaintext attack*. Karena itulah disarankan untuk menggunakan urutan lemparan koin biner untuk melawan *known-*plain*text attack* tetapi ternyata hal ini ternyata keliru.

Misalkan kita mengetahui frame tertentu pada video yang dalam kasus ini dianggap *plaintext*. Dengan membandingkan antara *plaintext* dan frame terenkripsi yang berkorespodensi kita dapat dengan mudah mengetahui kunci 1 dan kunci 2 (kecuali hanya digunakan sebuah kunci yang tentunya bertentangan dengan tujuan awalnya). Dengan demikian kita dapat menggunakan kunci 1 atau kunci 2 untuk mendekripsi blok pertama secara terpisah. Dengan koefisien AC yang bukan 0 yang mempunyai kecenderungan berada pada bagian atas kiri dari sebuah blok, yang pasti bit yang mempunyai kecenderungan ini adalah yang benar dan sebagian besar dari yang tidak mempunyai kecenderungan ini karena permutasi yang acak. Dengan melakukan cara ini terhadap blok yang lain kita hanya membutuhkan waktu hampir dua kali dari waktu dekripsi normal.

Dasar dari algoritma ini pun juga lemah terhadap *ciphertext only attack*. Serangan yang berdasarkan bahwa koefisien AC tidak nol selalu berkumpul pada bagian sudut kiri atas dari blok I.

Dalam sebuah test sederhana yang dilakukan yang menghitung angka tidak nol dati koefisien DC dan AC dari semua blok didalam sebuah frame I. Dua buah hasil dapat dilihat sebagai berikut :

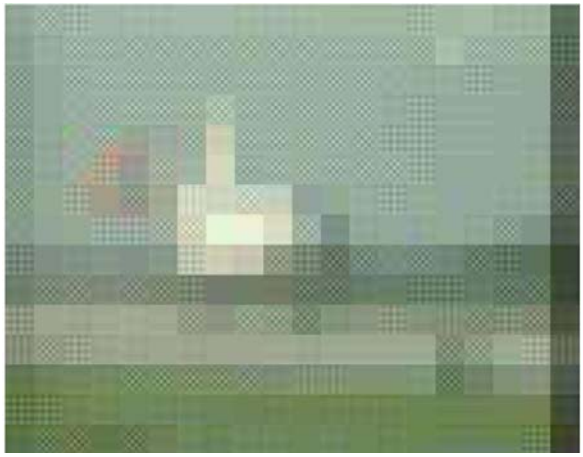
1314	1017	860	660	500	384	307	219
1103	881	662	504	344	226	134	95
917	704	513	356	249	140	101	52
800	593	417	318	206	104	58	36
734	472	374	247	139	79	23	6
591	413	293	181	92	38	5	1
544	347	246	130	50	12	1	0
480	302	157	91	20	2	0	1314

Tabel 1 bus.mpg : jumlah total dari blok adalah 1320

320	106	48	28	12	25	24	17
202	70	22	6	5	5	1	0
134	42	16	7	4	1	1	0
134	36	15	8	6	1	1	0
88	17	12	3	3	0	0	0
62	26	8	3	2	0	0	0
44	22	5	6	1	0	0	0
47	16	5	2	0	0	0	320

Tabel 2 space.mpg : jumlah total dari blok adalah 320

Dari table diatas kita dapat melihat bahwa : 1) koefisien DC selalu nilai tertinggi dari kemunculan bilangan tidak nol, makadari itu bika kita melakukan analisis yang sama pada frame I yang acak, kita dapat dengan mudah menentukan posisi dari DC. Perhatikan bahwa angka kiri atas (DC) dan kanan bawah (AC_{63}) selalu sama. Ini dikarenakan bagaimana cara koefisien DC dibagi. Kita mengubah seluruh nilai koefisien AC menjadi nol dan menngunakan koefisien DC untuk menciptakan gambar. Karena kita tidak mengetahui urutan dari keduanya, kita perlu mendekripsi dua kali, pertama menggunakan DC + AC_{63} dan yang lain menggunakan AC_{63}+DC. Gambar 1 menunjukan gambar yang didekripsi hanya menggunakan DC.



Gambar 1 Frame didekripsi hanya dengan DC

Frekuensi dari AC_1 dan AC_2 berada pada keenam nilai tertinggi. Sebagai contoh frekuensi dari AC_1 pada bus.mpg adalah 1017 yang merupakan nilai ketiga tertinggi. AC_2 pada bus.mpg adalah 1103 yang merupakan nilai kedua tertinggi. AC_1 pada space.mpg adalah 106 yang merupakan nilai kelima tertinggi. Karena itu untuk menentukan nilai dari AC_1 dan AC_2, kita perlu mencoba (6-1)*(6-2) = 20 kombinasi (6-1 dikarenakan posisi dari DC diketahui; 6-2 karena nilai dari AC_1 ditentukan). Gambar 2 menunjukan frame yang diddekripsi menggunakan DC dan AC_1, AC_2.

Gambar 2 Frame didekripsi dengan DC, AC_1 dan AC_2

Dengan cara yang sama kita dapat memperoleh nilai AC_3 sampai AC_5 yang berada pada kesepuluh nilai terbesar. Untuk memperoleh nilai yang bernar diperlukan maksimal 210 kombinasi. Gambar 3 menunjukan frame yang didekripsi menggunakan DC dan lima buah koefisien AC yang pertama.

Gambar 3 Frame didekripsi dengan DC dan lima koefisien AC pertama

Gambar ini cukup bagus untuk mengalahkan tujuan sebenarnya dari enkripsi dalam keadaan apapun. Untuk

perbandingan gambar 4 adalah frame asli yang tidak dienkripsi.

Gambar 4 Frame yang sebenarnya

Berdasarkan hal ini kita dapat menyimpulakn bahwa algoritma ini tidaklah aman. Umumnya kita tidak mau melakukan enkripsi sebelum proses kompresi karena mempertahankan statement bahwa "*Ciphertext* yang dihasilakn oleh algoritma enkripsi yang baik (hampir) tidak akan mempunyai informasi yang tidak berguna dan secara statistic dapat dibedakan dari angka acak. Tidak ada algoritma kompresi yang dapat lebih lanjut dengan sukses melakukan kompresi pada angka acak." Statement ini negasinya menyatakan bahwa jika kompresi dilakukan pada hasil *ciphertext* dari sebuah algoritma enkripsi dan bisa mengurangi *cipertext* tersebut maka tingkat keamanan dari algoritma enkripsi tersebut sangatlah kurang. Hal ini lah yang terjadi pada algoritma Algoritma Permutasi Zig-Zag (*Zig-Zag Permutation Algorithm*).

3.4. Algoritma Enkripsi Video (*Video Encryption Algorithm*)

Algoritma ini aman. Daftar kedua berfungsi sebagai *one-time pad* yang unik, mengenkripsi daftar yang kedua . Algoritma ini aman dari serangan *known-plaintext* ataupun *ciphertext only*. Dari percobaan yang dilakukan sebuah algoritma memerlukan waktu minimal beberapa bulan untuk menentukan kunci yang mempunyai panjan 56 bit. Dengan demikian bahwa algoritma ini cukup aman.

4. Analisis Tingkat Kecepatan dan Ukuran

4.1 *Naïve Algorithm*

Algoritma ini dapat dikatakan lambat, karena harus mengenkripsi keseluruhan arus MPEG demi tingkat keamana yang tinggi. Algoritma ini tidak menyebabkan penambahan ukuran.

4.2. Algoritma Seleksi (*Selective Algorithm*)

Dengan hanya mengenkripsi blok I saja mengurangi waktu enkripsi menjadi 70%-50% dari waktu yang seharusnya. Tetapi bila dilakukan perubahan frekuensi maka akan membuat waktu enkripsi menjadi bertambah karena waktu yang diperlukan berbanding lurus dengan panjang dari arus MPEG. Bila kita melakukan enkripsi pada keseleruhan frame I dan blok I pada frame P dan B akan membuat waktu enkripsi menjadi mendekati waktu normal dan akan sama dengan waktu yang diperlukan pada *naive algorithm* apabila data MPEG hanya mempunyai frame I saja.

4.3. Algoritma Permutasi Zig-Zag (*Zig-Zag Permutation Algorithm*)

Algortima mempunyai kecepatan yang tinggi hampir sama dengan kecepatan *encoding* atau *decoding* MPEG, tetapi algoritma ini menyebabkan penambahan ukuran pada *ciphertext* karena membuat daftar permutasi dan ukuran akan bertambah lebih besar bila ingin meningkatkan tingkat keamanan.

4.4. Algoritma Enkripsi Video (*Video Encryption Algorithm*)

Algortima ini tergolong cepat karena kita hanya memerlukan 1 buah Xor untuk memperoleh fungsi C dan 16 buah Xor untuk memperoleh fungsi E. Bila dibandingkan dengan DES standard maka kita memperoleh bahwa algoritma ini akan mengurangi waktu enkripsi sebesar 47%.

Algoritma ini sedikit menambah ukuran dari data karena algoritma ini melibatkan beberapa kunci. Tetapi karena kecilnya penambahan sehingga dapat dianggap tidak signifikan.

5. Kesimpulan

Dalam makalah ini kita membahas beberapa algoritma yang dapat digunakan untuk mengenkripsi data multimedia. Dalam perbandingannya kita hanya memperhatikan masalah tingkat keamanan, kecepatan enkripsi dan penambahan ukuran pada *ciphertext*.
Dalam hal tingkat keamanan maka kita dapat menggunakan *naïve algorithm*, tetapi dalam tingkat kecepatan dan penambahan ukuran maka kita dapat menggunakan algoritma enkripsi video. Bila dilihat secara

keseluruhan maka kita dapat menggunakna algoritma enkripsi video (VEA) karena dapat memenuhi kebutuhan dari *software* yang digunakan pada umunya, tetapi dari keempat algoritma tersebut setiap algoritma mempunyai kelebihan dan kekuranggannya masing-masing sehingga masih dituntut perlunya penelitian lebih lanjut sehingga diperoleh sebuah algoritma yang mempunyai kelebihan dalam semua aspek yang ada. Tetapi tidak tertutupi kemungkinan algoritma diatas dapat memenuhi kebutuhan sebuah program yang khusus.

REFERENSI

[1] L. Tang, "Methods for Encrypting and Decrypting MPEGVideo Data Efficiently", Proceedings of the 4'th ACM International Conference on Multimedia, Boston, MA, 1996, pp. 2 19-229.

[2] I. Agi dan L. Gong. An Empirical Study of MPEG Video Transmission. In *Proceedings of the Internet Society Symposium on Network and Distributed System Security*, San Diego, CA, Feb . 1996, pp 137-144.

[3] Y. Li, Z. Chen, S. Tan, dan R. Campbell. Security enhanced mpeg player. In *Proceeding of IEEE first International Workshop on Multimedia Software Development (MMSD '96)*, Berlin, Germany, March 1996.

[4] T.B. Maples dan G.A. Spanos. Performace Study of a Selective Encryption Scheme for the Security of Networked, Real-time Video. In *Proceedings of 4th International Conference on Computer Communications and Networks*, Las Vegas, Nevada, September 1995.

[5] J. Meyer dan F. Gadegast. Security Mechanism for Multimedia Data with the Example mpeg-1 Video. Avaible on WWW via http://www.powerweb.de/phade/phade.html.1995.

[6] L. Qiao and K. Nahrstedt, "Comparison of MPEG Algorithms ," International Journal on Computers and Graphics, Special Issue on Data Security in Image Communication and Network, vol. 22, num. 3, Permagon Publisher, 1998.

[7] G. Simmons. *Contemporary Cryptology The Science of Information Integrity*. IEEE Press, 1992.

[8] B. Bhargava, C. Shi, and Y. Wang, "MPEG Video Encryption Algorithms," Multimedia Tools and Applications, 2003.

[9] J. But, G. Armitage,"An Evaluation of Current MPEG-1 Ciphers and their Applicability to Streaming Video,"Australian Telecommunications Networks & Applications Conference 2004, (ATNAC2004), Sydney, Australia, December 8-10, 2004.

[10] X. Liu and A. M. Eskicioglu, "Selective Encryption of Multimedia Contents in Distribution Networks: Challenges and New Directions," IASTED International Conference on Communications, Internet, and Information Technology (CIIT 2003), Scottsdale, AZ, November 17-19,2003.