Aplikasi Enkripsi Video MPEG dengan *Video Encryption Algorithm* (VEA) yang Dimodifikasi dengan Algoritma RC4

Yusuf Fahmi Adiputera[*1)], Adi Wibowo[*2)], Aris Sugiharto[*3)]

[**]Jurusan Ilmu Komputer/Informatika, Fakultas Sains dan Matematika,
Universitas Diponegoro
yfa.adiputera@gmail.com, bowo.adi@gmail.com, aris.sugiharto@undip.ac.id

Abstrak

Video adalah salah satu konten multimedia yang sering digunakan seiring maraknya penggunaan smartphone. Seiring dengan perkembangan teknologi maka faktor keamanan untuk menjaga kerahasiaan dari video menjadi hal yang penting agar orang yang tidak berkepentingan tidak dapat melihat gambar dari video. Salah satu metode untuk mengamankan gambar dari video adalah dengan melakukan enkripsi. Metode enkripsi video yang dapat digunakan yaitu algoritma video encryption algorithm (VEA), algoritma ini melakukan enkripsi pada frame I dari video MPEG. Dalam tugas akhir ini algoritma VEA dimodifikasi dengan algoritma RC4 untuk menambah keamanannya. Aplikasi ini dibangun dengan menggunakan metode pengembangan perangkat lunak unified process dan implementasinya menggunakan bahasa pemrograman Java. Dari hasil pengujian, diperoleh hasil bahwa algoritma RC4 dapat meningkatkan tingkat keamanan dari algoritma VEA yang dapat dilihat dari nilai MSE rata-rata video hasil enkripsi algoritma VEA yang dimodifikasi dengan algoritma RC4 lebih tinggi dari nilai MSE rata-rata video hasil enkripsi algoritma VEA, selain itu pengujian juga memperlihatkan bahwa waktu enkripsi yang linier dengan durasi video dan resolusi video.

Kata kunci : *VEA, RC4, kriptografi, video.*

Abstract

Video is one of the multimedia content that is frequently used as the increase popularity of smartphone. With the development of technology, security factor is becoming more important to prevent unauthorized person from seeing the video. One way to secure the frame of a video is by encrypting the frame. Encryption method used in this application is video encryption algorithm (VEA), this algorithm encrypt frame type I of MPEG video. In this application VEA is modified with RC4 algorithm to increase the security. This application is created using software development methods Unified Process and its implementation using the Java programming language. Testing results reveal that RC4 algorithm can increase the security of VEA which can be seen from average MSE of encrypted video by VEA and RC4 is higher than average MSE of encrypted video by VEA only. Testing results also reveal that encryption time is linier with the duration of video and resolution of video.

Keywords : *VEA, RC4, cryptograph, video*

1. PENDAHULUAN

Dewasa ini, konten multimedia telah berkembang sangat pesat dan digunakan dalam berbagai bidang. Salah satu konten multimedia yang umum digunakan dalam masyarakat adalah video. Video saat ini digunakan dalam bidang komunikasi sebagai sarana dalam video chat dan juga dalam bidang hiburan yaitu berupa film.

Salah satu format video yang sering digunakan adalah video MPEG atau *Moving Pictures Expert Group*, yang merupakan standar industri dalam *video processing* [1]. Seiring dengan perkembangan teknologi maka faktor keamanan untuk menjaga kerahasiaan dari video menjadi hal yang penting agar orang yang tidak berkepentingan tidak dapat melihat isi dari suatu video. Salah satu cara untuk menjaga kerahasiaan dari suatu data adalah dengan melakukan enkripsi pada video.

Beberapa algoritma untuk melakukan enkripsi video mpeg telah diteliti, antara lain *naïve algorithm, selective algorithm, zig-zag permutation algorithm*, dan *video enrcryption algorithm* (VEA). *Naïve algorithm* memperlakukan baris bit *file* MPEG seperti hanya data teks tradisional dan tidak menggunakan sedikit pun bentuk spesial dari struktur *file* MPEG. Algoritma enkripsi tersebut, bagaimanapun sangat rumit dan melibatkan komputasi yang besar. Implementasi perangkat lunak dengan cara ini tidak cukup cepat untuk memproses jumlah data yang dihasilkan oleh aplikasi multimedia [2]. *Selective algorithm* bekerja dengan melakukan enkripsi hanya pada header dari video MPEG. *Zig-zag permutation algorithm* memiliki ide dasar dengan memetakan blok 8x8 ke dalam vektor 1x64 dalam urutan yang zig-zag. Algoritma VEA bekerja dengan memperhatikan struktur *file* dari sebuah *file* MPEG dengan hanya beroperasi pada *sign bits* dari koefisien DCT pada frame I dari sebuah *file* video MPEG [3]. Algoritma VEA dipilih karena merupakan algoritma yang cepat dan efisien dalam melakukan proses enkripsi [2].

Penelitian algoritma VEA sebelumnya di antaranya dengan menambahkan algoritma DES dan fungsi hash MD5 untuk penerapannya dalam video *streaming* [5]. Dalam tugas akhir ini algoritma VEA dimodifikasi dengan tambahan algoritma RC4 untuk meningkatkan keamanannya.

RC4 merupakan jenis algoritma kriptografi stream cipher (cipher aliran) yang beroperasi pada bit tunggal. Oleh karena termasuk dalam cipher aliran maka dalam proses enkripsinya memakan waktu yang sangat singkat [6]. Algoritma RC4 merupakan algoritma enkripsi simetris, yang berarti hanya satu kunci yang digunakan dalam proses enkripsi dan dekripsi [7]. Dengan algoritma enkripsi simetris maka tingkat keamanan tergantung pada pengguna dalam menyimpan kunci yang digunakan, jika kunci diketahui oleh penyerang maka data dapat dengan mudah didekripsi [8].

Berdasarkan hal tersebut di atas, maka pada tugas akhir ini akan diteliti tentang enkripsi video dengan menggunakan algoritma VEA yang dimodifikasi dengan algoritma RC4. Untuk menambah keamanan maka dilakukan enkripsi pada keseluruhan frame I dan P. Aplikasi ini diharapkan dapat menjadi alternatif dalam pengamanan video MPEG

2. METODE

2.1. *VIDEO ENCRYPTION ALGORITHM* (VEA)

Seperti halnya pada proses enkripsi pada teks, enkripsi pada video dapat menggunakan berbagai algoritma, seperti AES, DES, ataupun yang lainnya. Akan tetapi masalah timbul karena besarnya ukuran dari sebuah video, oleh karena itu digunakanlah algoritma VEA. Algoritma VEA menawarkan kecepatan dalam proses enkripsi video tanpa mengabaikan tingkat

keamanannya [5]. Algoritma VEA bekerja dengan memperhatikan struktur video MPEG, dengan hanya bekerja pada frame I pada *Group of Pictures* video mpeg, hal ini menyebabkan proses enkripsi menjadi lebih cepat dibandingkan dengan melakukan enkripsi pada semua frame yang ada.

Tahapan dalam algoritma ini adalah [9]:

1. Anggap data dari frame I adalah dalam bentuk sebagai berikut a1a2a3a4a...a2n
2. Pilih bytes dengan nomor ganjil dan nomor genap untuk membuat dua buah arus bytes yang baru.
3. Lakukan operasi XOR pada kedua buah arus

	a_1	a_3	...	a_{2n-1}
XOR	a_2	a_4	...	a_{2n}
	c_1	c_2	...	c_n

4. Pilih fungsi enkripsi E untuk mengenkripsi a2, a4, ..., a2n

Algoritma VEA jika dituliskan dapat dilihat pada kode 2.1

```
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*/
```

Kode 2.1. Algoritma VEA

2.2. ALGORITMA RC4

Sistem sandi RC4 dikembangkan oleh Ronald Rivest pada tahun 1984 untuk RSA Data Security, Inc. RC4 merupakan sistem sandi *stream* berorientasi *byte*. Masukan algoritma enkripsi RC4 merupakan sebuah *byte*, kemudian dilakukan operasi XOR dengan sebuah *byt*e kunci, dan menghasilkan sebuah *byte* sandi [6].

Sistem sandi RC4 menggunakan *State*, yaitu larik *byte* berukuran 256 yang terpermutasi, dan tercampur oleh kunci. Kunci juga merupakan larik *byte* berukuran 256. Sebelum melakukan enkripsi, dan dekripsi, sistem sandi RC4 melakukan inisialisasi terhadap *State*, algoritma ini disebut dengan penjadwalan kunci [6]. Algoritma penjadwalan kunci dapat dilihat pada kode 2.2.

```
input: Kunci
output: {S[1],...,S[N]}
    for i=0 -> 255 do
      S[i]=i
    end for
    j=0
    for i=0 -> 255 do
      j=(j+S[i]+Kunci[i mod |Kunci|])
      mod 256
```

```
    swap(S[i],S[j])
end for
```

Kode 2.2. Algoritma Penjadwalan Kunci RC4 [6]

Setelah *State* S terinisialisasi oleh penjadwal kunci setiap *byte* pada teks asli dikenakan operasi XOR dengan kunci *byte* untuk menghasilkan *byte* pada teks sandi. Kunci *byte* yang digunakan pada enkripsi dibangkitkan dengan memanfaatkan *State* S. Untuk melakukan dekripsi pada sistem sandi RC4 menggunakan algoritma serupa dengan algoritma enkripsi sistem sandi RC4 [6]. Algoritma enkripsi RC4 dapat dilihat pada kode 2.3.

```
input: P{stream teks asli}
output: C{stream teks sandi}
   i=0, j=0{bisa diisi nilai lain}
   while P masih memiliki byte do
        i=(i+1)mod 256
        j=(j+S[i])mod 256
        swap(S[i],S[j])
        k=S[S[i]+S[j]]mod 256
        C=P XOR k
   end while
```

Kode 2.3. Algoritma Enkripsi RC4

2.3. ALGORITMA VEA YANG DIMODIFIKASI DENGAN RC4

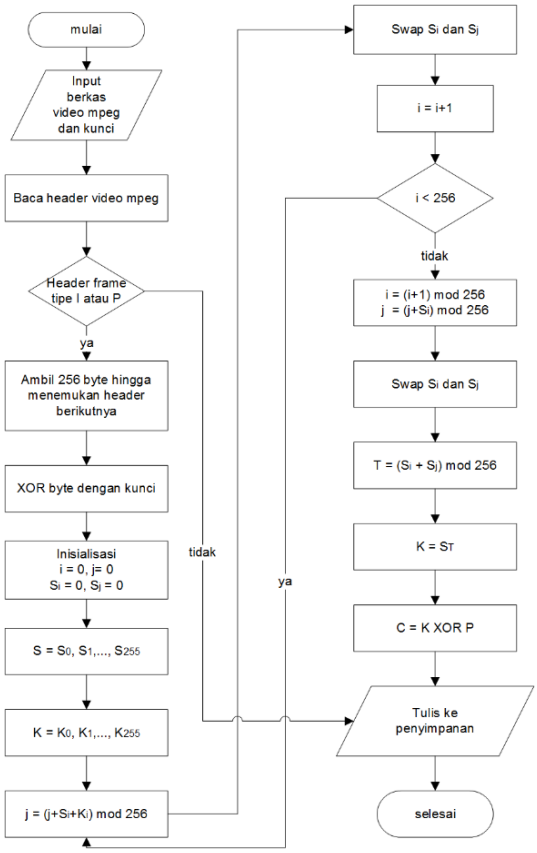
Modifikasi dari algoritma VEA adalah sebagai berikut:

1. Enkripsi dilakukan pada *frame* I dan P dari video MPEG
2. Proses enkripsi dilakukan dengan fungsi algoritma RC4

Langkah-langkah algoritma VEA yang dimodifikasi dengan algoritma RC4 adalah sebagai berikut:

1. *Input* berkas video MPEG dan kunci
2. Baca *header* video MPEG
3. Cek apakah *header* tersebut berasal dari *frame* I atau P
4. a. Jika ya, ambil 256 *byte* atau hingga menemukan *header* selanjutnya
   b. Jika tidak, maka data langsung ditulis ke penyimpanan
5. *byte* yang telah diambil dilakukan proses XOR dengan kunci
6. Hasil dari proses XOR dilakukan proses enkripsi dengan algoritma RC4
7. Inisialisasi i = 0 dan j = 0
8. Inisialisasi $S_i = 0$ dan $S_j = 0$
9. Buat 256 *byte state array* $S = S_0, S_1, ..., S_{255}$
10. Buat 256 *byte array* kunci $K = K_0, K_1, ..., K_{255}$
11. Dilakukan perhitungan $j = (j+S_i+K_j) \bmod 256$
12. Tukar S_i dan S_j, ulang hingga 256 kali
13. Set indeks i dengan rumus $i = (i+1) \bmod 256$
14. Set indeks j dengan rumus $j = (j+Si) \bmod 256$
15. Tukar S_i dan S_j
16. Set T dengan rumus $T = (S_i+S_j) \bmod 256$
17. *Byte* $K = S_T$
18. XOR K dengan *plaintext* P(*byte* dari video MPEG) untuk menghasilkan *chipertext*
19. Tulis hasil ke penyimpanan
20. Ulangi proses RC4 hingga menemukan *header* selanjutnya

Diagram alur proses enkripsi dari modifikasi algoritma dapat dilihat pada gambar 2.1.

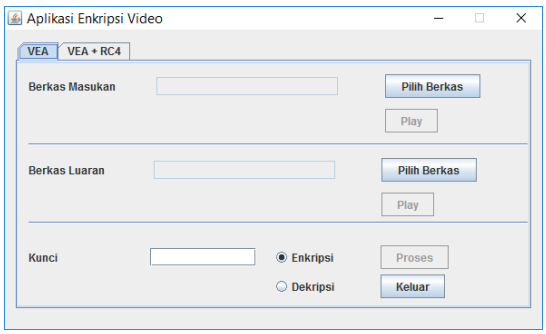


Gambar 2.1. Alur Proses Enkripsi

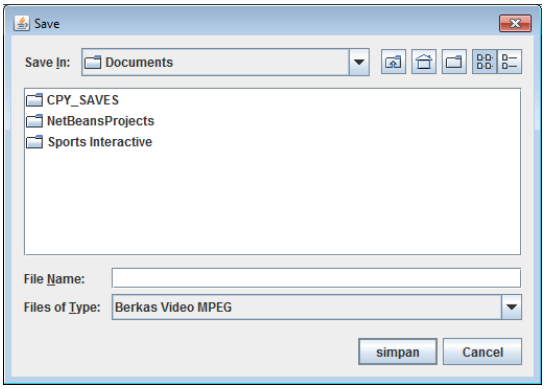
3. IMPLEMENTASI

Aplikasi ini dibangun berbasis desktop pada sistem operasi windows 10 Professional 64-bit dengan bahasa pemrograman Java yang dibangun dengan NetBeans IDE 8.2 dan menggunakan windows media player.

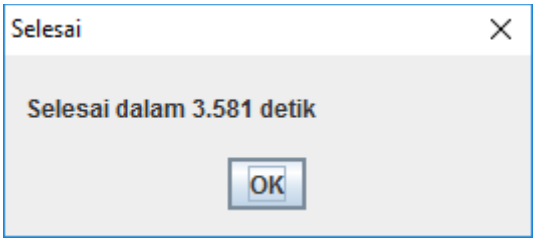
Tampilan antarmuka dari aplikasi ini dapat dilihat pada gambar 3.1, 3.2, dan 3.3



Gambar 3.1. Antarmuka aplikasi



Gambar 3.2. Antarmuka memilih Berkas



Gambar 3.3. Pemberitahuan Proses Selesai

4. PENGUJIAN

Pengujian dilakukan dengan menggunakan beberapa video "ponsel.mpg" yang berdurasi 1 menit 24 detik dengan resolusi yang berbeda, yaitu 256 x 144 piksel dengan ukuran 6,08 MB (6.385.664 byte), 427 x 240 piksel dengan ukuran 6,12 MB (6.422.528 byte), 569 x 320 piksel dengan ukuran 6,64 MB (6.963.200 byte), 853 x 480 piksel dengan ukuran 9,04 MB (9.480.192 byte), dan 1138 x 640 piksel dengan ukuran 12,4 MB (13.053.952 byte). Selain itu digunakan video "ponsel.mpg" dengan resolusi yang sama yaitu 853 x 480 piksel, namun dengan durasi yang berbeda, yaitu 10 detik dengan ukuran 1,06 MB (1.112.064 byte), 20 detik dengan ukuran 2,15 MB (2.258.944 byte), 30 detik dengan ukuran 3,35 MB (3.520.512 byte), 40 detik dengan ukuran 4,71 MB (4.945.920 byte), 50 detik dengan ukuran 5,73 MB (6.008.832 byte), dan 60 detik dengan ukuran 6,65 MB (6.979.584 byte). Cuplikan video "ponsel.mpg" dapat dilihat pada gambar 4.1.

Gambar 4.1. Cuplikan video "ponsel.mpg"

Video-video tersebut akan dienkripsi dengan menggunakan kunci "adiputera". Cuplikan video hasil enkripsi dapat dilihat pada gambar 4.2.

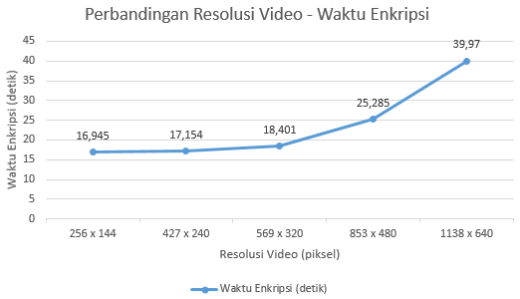
Gambar 4.2. Cuplikan Video "ponsel.mpg" hasil enkripsi

Setelah dilakukan enkripsi pada semua video "ponsel.mpg" yang mempunyai resolusi berbeda didapat perbandingan waktu enkripsi dengan resolusi video yang dapat dilihat pada tabel 4.1.

Tabel 4.1. Perbandingan waktu enkripsi dengan resolusi video

No.	Resolusi (piksel)	Waktu Enkripsi (detik)
1.	256 x 144	16,945
2.	427 x 240	17,154
3.	569 x 320	18,401
4.	853 x 480	25,285
5.	1138 x 640	39,97

Berdasarkan tabel tersebut dapat dibuat sebuah grafik yang menunjukkan hubungan resolusi dan waktu enkripsi. Grafik dapat dilihat pada gambar 4.3



Gambar 4.3. Grafik Perbandingan Resolusi Video dengan Waktu Enkripsi

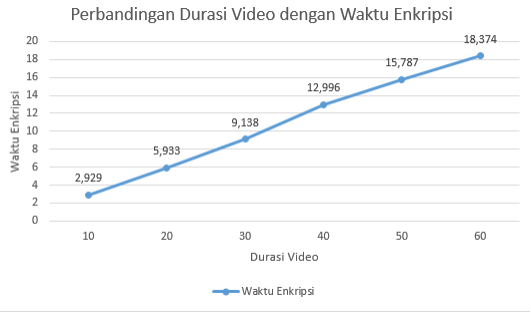
Berdasarkan grafik tersebut dapat disimpulkan bahwa semakin besar resolusi video maka waktu yang dibutuhkan untuk melakukan enkripsi akan semakin besar pula.

Pada video "ponsel.mpg" yang mempunyai resolusi sama namun dengan durasi yang berbeda, yaitu 10 detik hingga 60 detik hasilnya dapat dilihat pada tabel 4.2.

Tabel 4.2. Perbandingan waktu enkripsi dengan durasi video

No.	Durasi (detik)	Waktu Enkripsi (detik)
1.	10	2,929
2.	20	5,933
3.	30	9,138
4.	40	12,996
5.	50	15,787
6.	60	18,374

Berdasarkan tabel tersebut dapat dibuat sebuah grafik yang menunjukkan hubungan antara durasi video dengan waktu enkripsi. Grafik dapat dilihat pada gambar 4.4.



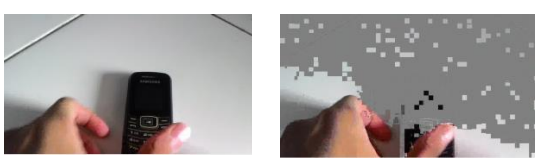
Gambar 4.4. Grafik Perbandingan Durasi Video dengan Waktu Enkripsi

Selain diuji berdasarkan waktu enkripsi, dihitung nilai rata-rata *Mean Squared Error* (MSE) dengan menggunakan aplikasi MSU Video Quality Measurement Tools untuk membandingkan video hasil enkripsi dengan video asli. Hasil nilai MSE dapat dilihat pada tabel 4.3.

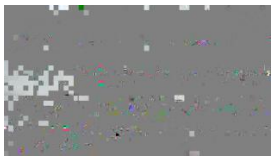
Tabel 4.3. Rata-rata Nilai MSE

No	*Frame per second*	Durasi (detik)	Resolusi (piksel)	Nilai MSE Rata-Rata	
				VEA	VEA + RC4
1.	30	10	853x480	2255,133	5681,735
2.	30	20	853x480	3093,520	6079,276
3.	30	30	853x480	4297,970	6757,931
4.	30	84	427x240	4415,500	5421,235
5.	30	84	569x320	4403,007	6059,011

Perbandingan hasil enkripsi dengan menggunakan VEA dan VEA yang dimodifikasi dengan algoritma RC4 dapat dilihat pada gambar 4.5 dan gambar 4.6

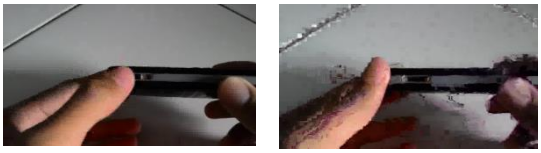


(a) (b)



(c)

Gambar 4.5. frame ke-50 dari video (a) asli, (b) hasil enkripsi dengan VEA, (c) hasil enkripsi dengan VEA yang dimodifikasi dengan algoritma RC4



(a) (b)

(c)

Gambar 4.6. frame ke-500 dari video (a) asli, (b) hasil enkripsi dengan VEA, (c) hasil enkripsi dengan VEA yang dimodifikasi dengan algoritma RC4

5. KESIMPULAN

Kesimpulan yang dapat diambil dari pembuatan aplikasi enkripsi video mpeg dengan *video encryption algorithm* (VEA) yang dimodifikasi dengan algoritma RC4 adalah sebagai berikut:

1. Tugas akhir ini menghasilkan aplikasi enkripsi video mpeg dengan menggunakan algoritma *video encryption algorithm* (VEA) yang dimodifikasi dengan algoritma RC4 yang dibangun dengan menggunakan bahasa pemrograman Java dan dikembangkan dengan metode *unified process*.
2. Dari hasil pengujian menunjukkan waktu komputasi enkripsi yang linier terhadap ukuran resolusi video maupun durasi video, hal ini dikarenakan implementasi algoritma secara linier.

3. Dari hasil penghitungan nilai MSE menunjukkan bahwa algoritma VEA yang dimodifikasi dengan algoritma RC4 mempunyai tingkat keamanan yang lebih baik dibanding dengan algoritma VEA, hal ini ditunjukkan dengan lebih tingginya nilai MSE dari algoritma VEA yang dimodifikasi dengan algoritma RC4

REFERENSI

[1] I. Agi dan L. Gong, An Empirical Study of Secure MPEG Video Transmissions, California: Institute of Electrical and Electronics Engineers, 1996.

[2] C. Shi, S.-Y. Wang dan B. Bhargava, MPEG Video Encryption in Real-time Using Secret Key Cryptography, West Lafayette: Purdue University, 1999.

[3] B. Bhargava, C. Shi dan S.-Y. Wang, MPEG Video Encryption Algorithms, West Lafayette: Purdue University, 2002.

[4] L. Qiao dan K. Nahrstedt, A New Algorithm for MPEG Encryption, Urbana: University of Illinois, 1997.

[5] D. I. Savitri, Perancangan dan Implementasi Modifikasi Algoritma VEA (Video Encryption Algorithm) untuk Video Streaming, Bandung: Institut Teknologi Bandung, 2007.

[6] R. Sadikin, Kriptografi Untuk Keamanan Jaringan, Yogyakarta: Penerbit Andi, 2012.

[7] H. Kim, J. Han dan S. Cho, "An Efficient Implementation of RC4 Cipher for Encrypting," Seoul, 2007.

[8] K. S. Kadam dan A. B. Deshmukh, "A Review on Video Encryption Technologies," *International Journal of Advanced Research in Computer and Communication Engineering*, no. 3, pp. 420-423, 2016.

[9] M. Oni, Algoritma Enkripsi pada Video MPEG, Bandung: Institut Teknologi Bandung, 2009.