INFORMATICS AND DIGITAL EXPERT (INDEX) - VOL. 4 NO. 1 (2022) 33-39
Terbit *online* pada laman web jurnal : https://e-journal.unper.ac.id/index.php/informatics
| ISSN (Print) 2775-2208 | ISSN (Online) 2715-0453 |

Data Encryption Pada File Video Menggunakan *Algoritma Blowfish* Berbasis Android

M Hasan Thoriq Almuwaffaq[1], Asep Id Hadiana[2], Puspita Nurul Sabrina[3]

[123]*Informatika, Fakultas Sains dan Informatika, Universitas Jenderal Achmad Yani, Jl. Terusan Sudirman, Cimahi*
mhasanthoriqa@gmail.com[1], ahadiana@gmail.com[2], puspitasabrina14@gmail.com[3]

INFORMASI ARTIKEL

Sejarah Artikel:-
Diterima Redaksi : 31 Januari 2022
Revisi Akhir : 13 Juni 2022
Diterbitkan *Online* : 03 Juli 2022

Kata Kunci:
Data-Encryption, Algoritma *Blowfish*, Video, Android

Korespondensi:
Telepon / Hp : +62 81220622160
E-mail : **mhasanthoriqa@gmail.com**

A B S T R A K

Terdapat banyak penelitian mengenai cara mengamankan video dengan aman menggunakan algoritma *blowfish*, namun kebanyakan cara mengamankan data tersebut hanya melalui perangkat komputer, masih sangat sedikit penelitian mengenai cara mengamankan data melalui perangkan android. Penelitian ini bertujuan membangun sebuah aplikasi yang dapat menenkripsi dan mendekripsi data berupa video berbasis android menggunakan algoritma *blowfish*. Adapun tahapan yang digunakan dimulai dari pra proses yang meliputi mengubah video menjadi byte menggunakan algoritma base64, pemecahan dan penambahan bit, pemecahan 64 bit menjadi 32 bit, perhitungan *sub-key*, dan proses enkripsi dekripsi menggunakan algoritma *blowfish*. Pengujian pada penelitian ini menghasilkan waktu proses enkripsi dan dekripsi yang berbeda. Waktu proses dekripsi lebih cepat 8,7% dari waktu proses enkripsi. Perbedaan ukuran file video juga mempengaruhi lama proses enkripsi, karena semakin banyak byte semakin lama juga proses enkripsi. Pengujian dengan mengenkripsi file video berukuran 2,67GB, aplikasi dapat menjalankan proses enkripsi dan dekripsi tanpa terjadinya crash pada aplikasi. Dari uji keamanan pada file video yang terenkripsi menghasilkan bahwa file video aman walaupun terjadi penyerangan, seandainya attacker memiliki *ciphertext*dan mengetahui kuncinya tidak akan membuat attacker tersebut mengetahui isi video. Hasil dari penelitian ini adalah sebuah aplikasi yang dapat menenkripsi dan mendekripsi file video menggunakan algoritma *blowfish* berbasis android.

1. PENDAHULUAN

Dengan perkembangan teknologi, penggunaan multimedia meningkat pesat, dan kebutuhan untuk melindungi data yang dibagikan melalui jaringan internet telah menjadi masalah penting. Saat ini, aplikasi multimedia seperti video call, video sesuai permintaan, siaran video, konferensi, rekaman rapat kantor, rekaman pembelajaran oleh guru, dan lain-lain telah banyak digunakan[1]. Dengan peningkatan pesat baru-baru ini dalam penggunaan Internet dan teknologi telekomunikasi, pengguna multimedia dapat dengan mudah berbagi video melalui perangkat pintar, dan serangan keamanan juga meningkat. Cisco menyebutkan lalu lintas video seluler menyumbang 59 persen dari total lalu lintas data seluler pada tahun 2017. Lalu lintas video seluler sekarang menyumbang lebih dari setengah dari semua lalu lintas data seluler[2], artinya masalah keamanan menjadi sangat penting karena akan memastikan keamanan data. Khususnya multimedia agar tidak dicuri oleh pihak yang tidak berkepentingan.

Teknologi enkripsi merupakan salah satu teknologi keamanan yang cukup terkenal, salah satu ilmu yang dapat menyandikan data untuk menjaga kerahasiaan data[3]. Terlebih lagi pada tahun 2020 seluruh dunia sepakat untuk melakukan segala hal dari rumah seperti proses belajar mengajar, rapat kerja, ibadah, dan lain-lain[4]. Aktivitas tersebut dilakukan melalui conference video, atau membuat rekaman video dan dikirimkan kepada orang lain. Karena banyaknya aktivitas untuk mengirim video melalui Internet, hal ini dapat menimbulkan kejahatan manipulasi data, seperti mecuri video yang bukan miliknya. Oleh karena itu, teknologi kriptografi video sekarang sangat diperlukan untuk terhindar dari manipulasi data dan menjaga data tetap utuh.

Terdapat banyak penelitian tentang kriptografi, diantaranya Algoritma berbasis AES yang dimodifikasi untuk enkripsi video MPEG[5], Enkripsi Video Lapisan Ganda menggunakan Algoritma RSA[6], Skema enkripsi video menggunakan teknologi enkripsi hybrid[7], Enkripsi Untuk Pengkodean Video Efisiensi Tinggi Dengan Kemampuan Adaptasi Video[8], Implementasi Teknik Enkripsi dan Dekripsi Di File Video Menggunakan Algoritma *Blowfish*[9], Implementasi Algoritma *Blowfish* dan Metode Least Significant Bit Insertion Pada Video Mp4[10]. Walaupun metode ini memiliki keamanan yang baik, namun ada beberapa hal yang harus dikorbankan yaitu lamanya proses enkripsi.Selain itu, meskipun ada yang menggunakan algoritma yang sama seperti penelitian ini yaitu algoritma *Blowfish*, penelitian tersebut hanya digunakan untuk peralatan komputer saja. Oleh karena

itu Algoritma *Blowfish* diusulkan dalam penelitian ini untuk mengamankan data yang aman dengan proses yang cepat dan dapat dilakukan pada perangkat Android.

Penelitian ini membuat suatu aplikasi yang mampu melakukan penyandian (enkrip dan dekrip) berupa video yang berbasis android menggunakan metode Algoritma *Blowfish*.

2. LANDASAN TEORI

2.1. Kriptografi

Kriptografi yaitu metode analisis dan implementasi beberapa konsep matematika yang digunakan untuk memastikan keamanan saluran komunikasi ketika ada penyusup. Bagian utama dari kriptografi adalah enkripsi dan dekripsi. Pesan terenkripsi berisi semua informasi asli, hanya saja dalam format yang tidak dapat dibaca manusia sehingga tidak semua orang dapat melihat informasi tersebut[11].

Jika dilihat dari perspektif sejarah, yang tertua dan terpenting tujuan kriptografi adalah kerahasiaan, menjaga bagaimana data tersebut dapat tetap dirahasiakan ketika berada di tempat yang tidak aman seperti internet. Ada banyak metode kriftografi yang bisa digunakan, beberapa di antaranya adalah AES, DES, RC5, RSA dan *Blowfish* [12].

2.1.1 Enkripsi

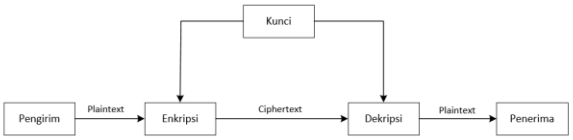
Enkripsi adalah sebuah metode yang mengubah informasi yang dapat terbaca menjadi sebaliknya atau istilahnya disebut dengan mengubah Plaintext menjadi *Ciphertext* dengan demikian orang yang hanya memiliki kunci akses untuk melihatnya. Data rahasia harus diubah menjadi bentuk terenkripsi sebelum dikirim melalui internet yang tidak memiliki jaminan keamanan sama sekali[13]. Tujuan enkripsi adalah untuk mengubah data dengan maksud untuk merahasiakannya. Dengan menggunakan algoritma untuk mengenkripsi data dan dapat didekripsi hanya menggunakan kunci khusus[14].

2.1.2 Dekripsi

Mendekripsi data adalah sebuah metode sebaliknya dari enkripsi. Dekripsi adalah untuk mengubah data yang terenkripsi ke dalam bentuk aslinya atau bisa disebut dengan mengubah *ciphertext* menjadi *plaintext* . Kunci akan diberikan kepada orang yang berwenang untuk mengubah data terenkripsi menjadi data asli atau data yang dapat dibaca manusia[15].

2.1.3 Kriptografi Kunci Simetris

Algoritma simetris merupakan algoritma enkrip pada proses enkripsi dan dekrispi dengan kunci yang sama. Algoritma ini memiliki syarat bahwa pengirim dan penerima mengetahui kunci sebelum melakukan enkripsi dan dekripsi. Keamanan algoritma simetris bergantung pada pengirim dan penerima, jika kunci yang dipakai pada proses enkrip dan dekrip bocor, keamanan data akan mudah diserang oleh peretas. Untuk menjaga keamanan data, kunci harus dirahasiakan. Algoritma simetris dikenal juga sebagai algoritma kunci tunggal, algoritma kunci rahasia atau algoritma satu kunci[16].



Gambar 1. Arsitektur Kriptografi Kunci Simetris

2.2. Sistem Enkripsi Android

Sistem enkripsi utama di android adalah sistem enkripsi berbasis file system berbasis dm-crypt3, yang tersedia di kernel Linux mulai dari versi 2.6.x. Enkripsi sistem file Android diperkenalkan di Android 3.0, yang hanya digunakan untuk tablet. Versi ini kemudian diterapkan ke *smartphone*. Di versi 4.4, KDF yang digunakan telah diubah, yang meningkatkan tingkat keamanan kata sandi yang digunakan. Android yang memiliki versi dimulai dari android 4.4 memiliki keamanan yang sangat meningkat dari versi android sebelumnya[17].

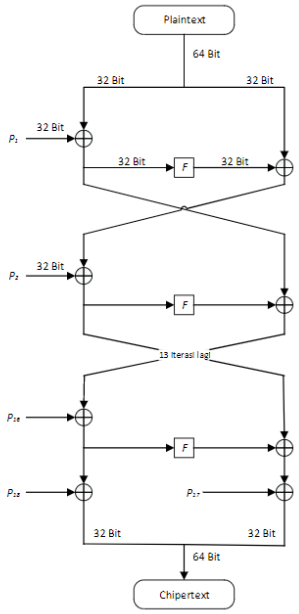
2.3. Algoritma *Blowfish*

Blowfish yaitu suatu algoritma kriptografi kunci simetris *code block* yang memiliki panjang tetap blok 64 bit(8 byte). *Blowfish* memiliki penerapan teknologi kunci ukuran apa pun. Ukuran kunci *blowfish* memiliki variasi 32 bit (4byte) dan variasi 448 bit (56 byte), dan memiliki default dengan variasi 128 bit (16 byte). algoritma *blowfish* memiliki dua sub algoritma utama yang mencakup ekspansi kunci dan data enkrip-dekrip. Ekspansi kunci dilaksanakan sebelum melakukan enkripsi-dekripsi, dengan memasukkan kunci yang bervariasi antara 32 sampai 448 bit. Bagian enkripsi dan dekripsi data menggunakan jaringan Feistel yaitu dengan melakukan 16 kali putaran[18]. Adapun alur algoritma *blowfish* seperti dibawah ini:

a. Inisialisasikan P-array yang berjumlah 18 (P1,P2, ... P18) yang masing-masing memiliki nilai 32 bit.
b. Inisialisasikan S-box yang berjumlah 4 yang masing-masing memiliki nilai 32 bit dengan masukan 256.
   S1,0,S1,1, ... ,S1,255
   S2,0,S2,1, ... ,S2,255
   S3,0,S3,1, ... ,S3,255
   S4,0,S4,1, ... ,S4,255
c. Plainteks diambil sebanyak 64 bit diasumsikan sebagai masukan, jika tidak mencapai 64 bit, bit-bit ini diisi sehingga operasi selanjutnya cocok dengan data.
d. Bagi 64 bit menjadi 2 bagian sehingga menjadi 32 bit dan masukan masing-masing 32 bit tersebut ke dalam XL dan XR
e. lakukan proses XL = XL xor dengan Pi dan XR = F(XL) xor dengan XR.

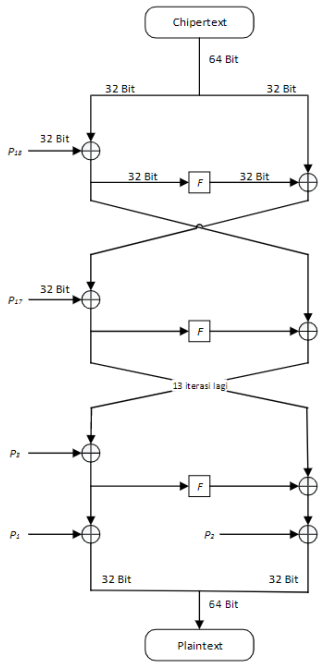
f. Hasil dari proses tersebut tukar XL menjadi XR dan XR menjadi XL
g. Buat perulangan sejumlah 16 kali, setelah 16 kali tukar kembali XL dengan XR
h. Setelah 16 kali putaran, tahap ke 17 lakukan operasi XR = XR xor dengan P17 dan XL = XL xor dengan P18.

Tahap akhir, gabungan lagi XL dengan XR hingga keluarannya 64 bit.



Gambar 2. Proses Enkripsi Algoritma *Blowfish*

Untuk proses dekripsi tidak terlalu beda prosesnya, perbedaan terdapat pada P-array yang dilakukan dengan urutan terbalik.



Gambar 3. Proses Dekripsi Algoritma *Blowfish*

2.3.1 Jaringan Feistel

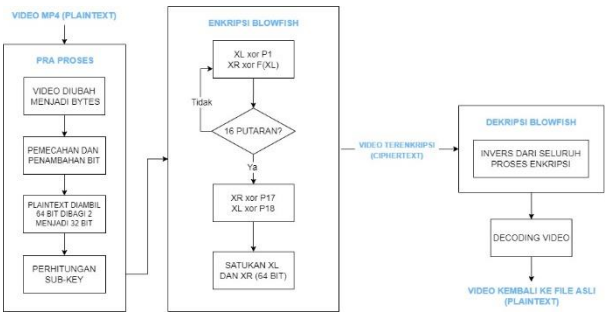
Jaringan Feistel adalah struktur simetris yang digunakan untuk mengkontruksi cipher blok. Jaringan Feistel memiliki keunggulan dalam proses enkripsi/dekripsi karena operasi enkripsi dan dekripsinya sama. Dalam jaringan Feistel, beberapa fungsi memerlukan penggunaan kunci dalam proses enkripsi/dekripsi. Jaringan Feistel adalah blok cipher yang dieksekusi berkali-kali secara iteratif. Fungsi-fungsi pada jaringan Feistel bersifat reversibel, artinya tidak perlu mengubah algoritma pada fungsi tersebut saat melakukan enkripsi dan dekripsi. Hal ini akan bermanfaat bagi pembuat fungsi karena dapat membuat fungsi serumit mungkin[19].

2.3. Algoritma Base64

Base64 yaitu algoritma untuk melakukan *encode* dan *decode* data menjadi format ASCII berdasarkan base 64, atau salah satu metode yang digunakan untuk mengkodekan data biner. Karakter keluaran dari konversi Base64 termasuk A-Z, a-z, dan 0-9, serta dua buah karakter terakhir + dan /, karakter “sama dengan” (=) sebagai pengisi pada 26 karakter simbol yang digunakan untuk mengatur dan mengimplementasikan data biner, yang akan dibang sesuai dengan proses menjalankan suatu algoritma. Algoritma ini banyak digunakan sebagai suatu media format data untuk pengiriman data, hal ini berdasarkan hasil Base64 merupakan plaintext, sehingga data ini menjadi lebih mudah untuk dikirim dibandingkan dengan penggunaan format data biner[20].

3. Metode Aplikasi

Aplikasi yang dibangun dalam proses enkripsi dan dekripsi harus melewati beberapa tahapan sebelum akhirnya menjadi video yang diamankan. Arsitektur perancangan aplikasi pengamanan video ditunjukkan pada Gambar 4.



Gambar 4. Perancangan Aplikasi Pengamanan Video

Tahap Pertama yaitu mengubah file video menjadi karakter base64. Dengan menghitung total byte yang ada dan jika ada satu string bytes akan dipecah menjadi 3 bytes atau sama dengan 24 bit, kemudian diubah ke dalam bentuk desimal dan dari bentuk desimal akan ditentukan ke dalam karakter base64. Setelah video menjadi karakter base64, karakter tersebut dilakukan perhitungan sub-key dengan meng-XOR kan kunci dengan P-Array dan akan menghasilkan keluaran kunci

yang terus berubah-ubah. Proses pemecahan bit dan penambahan bit dengan membagi total plainteks ke dalam 64 bit, apabila lebih atau kurang dari 64 bit akan dilakukan penambahan bit hingga menjadi sama rata 64 bit. Dari 64 bit tersebut dibagi menjadi 32 bit dan 32 bit tersebut dimasukan ke dalam XL dan XR.

Tahap Kedua yaitu proses enkripsi menggunakan algoritma *blowfish*. XL dari tahap pertama akan di XOR kan dengan Sub-Kunci yang telah dilakukan perhitungan. Setelah di XOR kan proses selanjutnya mengitung F(XL) dengan membagi menjadi 4 bagian dengan masing-masing bagian 8 bit dan masukkan ke dalam S-Box. Hasil perhitungan F(XL) akan di XOR kan lagi dengan XR untuk mencari nilai baru XR, tukar nilai XL dengan XR dan lakukan sebanyak 16 kali putaran. Setelah 16 kali putaran XOR-kan nilai XL dan XR (XR= XR xor dengan P17 dan XL xor dengan P18). Satukan XL dengan XR sehingga menjadi 64 bit.

Tahap Ketiga yaitu proses dekripsi, proses dekripsi pada perhitungan algorima *blowfish* dilakukan hampir sama dengan enkripsi. Namun P-array (P1, P2, ... , P18) dilakukan dengan urutan terbalik. Keluaran proses dekripsi akan menghasilkan karakter base64, dari daftar karakter tersebut akan di decoding lagi sehingga akan menghasilkan video asli.

4. HASIL DAN PEMBAHASAN

Dalam penelitian ini terdapat beberapa hasil yang terdapat dari pengujian enkripsi dekripsi, pengujian waktu proses enkripsi dekripsi dan pengujian terhadap file video yang terenkripsi.

4.1. Hasil Enkripsi

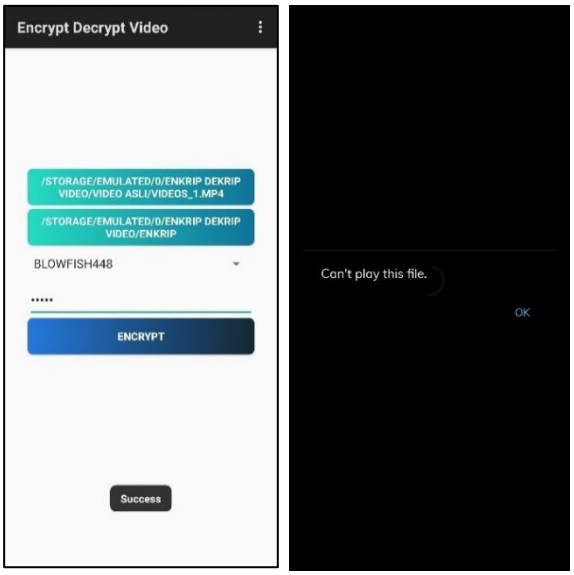
Pada Gambar 5 memperlihatkan hasil penelitian ini yaitu aplikasi yang dapat mengamankan file video. Jika dilihat dari hasilnya, video yang sudah dienkripsi tidak dapat diputar dan menjadi seperti file yang rusak, video tidak dienkripsi frame by frame karena pada saat proses enkripsi bit pada video diacak sehingga menjadikan video tidak dapat terbaca.

4.2. Hasil Dekripsi

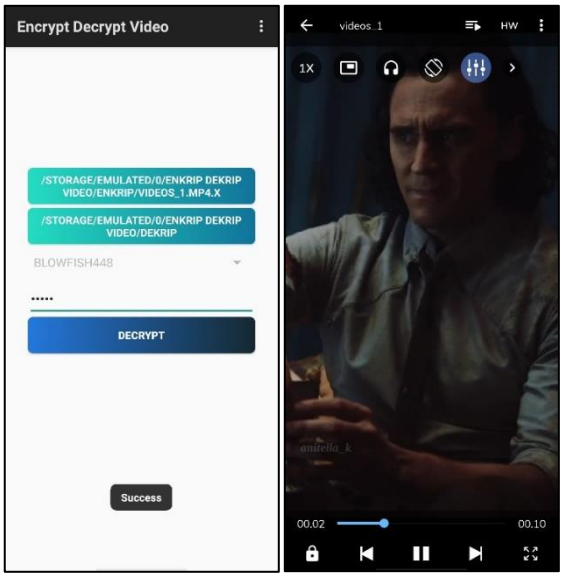
Dekripsi video hampir sama dengan proses enkripsi, perbedaannya terdapat pada prosesnya saja, dekripsi membalikkan proses enkripsi, dari bit video yang diacak dikembalikan menjadi bit asli. Hasil proses dekripsi video dapat dilihat pada Gambar 6.

4.3. Pengujian Waktu Proses Enkripsi Dekripsi

Pengujian pertama video diuji dengan menggunakan file format video dan ukuran video yang berbeda, terdapat 11 format video yang diuji untuk menghitung lama proses waktu enkripsi dengan menggunakan file format video yang berbeda. Hasil waktu proses enkripsi dan dekripsi ditunjukkan oleh Tabel 1 dan Tabel 2.



Gambar 5. Hasil Enkripsi

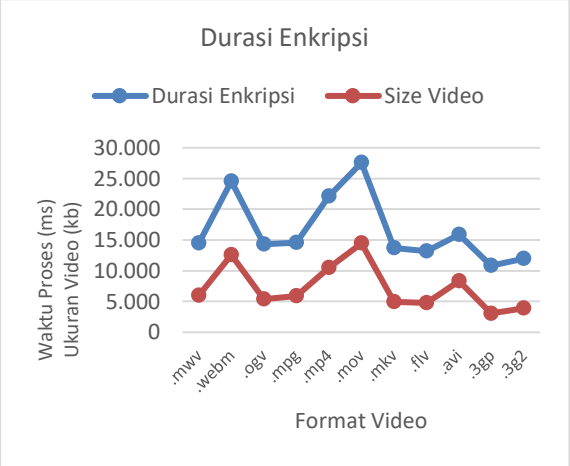


Gambar 6. Hasil Dekripsi

Tabel 1. Waktu Proses Enkripsi Terhadap Format Dan Ukuran Video Yang Berbeda

Format	Durasi Video	Size Video (kb)	Durasi Enkripsi (ms)
.mwv	02:06	5970	14.498
.webm	02:06	12610	24.549
.ogv	02:06	5400	14.317
.mpg	02:06	5910	14.587
.mp4	02:06	10540	22.149
.mov	02:06	14460	27.575
.mkv	02:06	4910	13.682
.flv	02:06	4820	13.232
.avi	02:06	8320	15.851
.3gp	02:06	3080	10.829
.3g2	02:06	3910	11.993

Hasil rata-rata pengujian dari tabel diatas jika ditampilkan dalam bentuk grafik dapat dilihat pada Gambar 7.



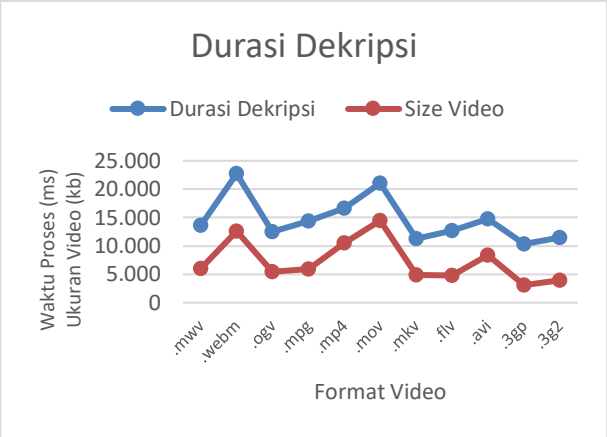
Gambar 7. Waktu Proses Enkripsi Terhadap Format Dan Ukuran Video Yang Berbeda

TABEL 2. WAKTU PROSES DEKRIPSI TERHADAP FORMAT DAN UKURAN VIDEO YANG BERBEDA

Format	Durasi Video	Size Video (kb)	Durasi Dekripsi (ms)
.mwv	02:06	5970	13.638
.webm	02:06	12610	22.714
.ogv	02:06	5400	12.477
.mpg	02:06	5910	14.377
.mp4	02:06	10540	16.646
.mov	02:06	14460	21.063
.mkv	02:06	4910	11.275
.flv	02:06	4820	12.674
.avi	02:06	8320	14.770
.3gp	02:06	3080	10.337
.3g2	02:06	3910	11.471

Hasil rata-rata pengujian dari tabel diatas jika ditunjukkan dalam tampilan grafik terdapat pada Gambar 8.

Jika dilihat dari grafik diatas, waktu proses dekripsi lebih cepat 8,7% dibanding waktu proses enkripsi. Hasil pengujian waktu proses enkripsi dan dekripsi menunjukkan bahwa format file yang berbeda tidak mempengaruhi waktu enkripsi. Walaupun dari segi grafik tidak sama tetapi hal itu dikarenakan ukuran file yang berbeda. Aplikasi yang dibangun tidak melihat format file video apa yang digunakan tetapi dari ukuran file yang digunakan. Karena pada saat proses enkripsi, video diubah menjadi bit sehingga jika semakin banyak jumlah bit untuk dienkripsi maka semakin lama waktu prosesnya.



Gambar 8. Waktu Proses Dekripsi Terhadap Format Dan Ukuran Video Yang Berbeda

Pengujian dengan ukuran file video yang sangat besar dengan ukuran file video 2,67GB juga dapat dijalankan aplikasi tanpa terjadinya crash.

4.4. Pengujian Uji Keamanan Pada File Video

Pengujian uji keamanan pada file video yang terenkripsi, video yang sudah terenkripsi harus dapat dipastikan tidak dapat diserang atau dilihat oleh *attacker* yang ingin mencoba melihat video asli walaupun *attacker* tersebut memiliki file video terenkripsinya dan mengetahui kuncinya.

Pengujian dilakukan dengan asumsi *attacker* memiliki video terenkripsi dan mengetahui kuncinya. Pengujian menggunakan 2 tools yaitu menggunakan openssl yang dijalankan dengan perangkat komputer dan menggunakan aplikasi Enkripsi Dekripsi File (EDF) yang dapat didownload di playstore. OpenSSL adalah sebuah toolkit kriptografi yang dapat melakukan enkripsi dekripsi file dengan berbagai pilihan algoritma yang tersedia dan dapat dijalankan dengan menggunakan command prompt. Sedangkan aplikasi EDF adalah toolkit kriptografi yang dapat digunakan pada perangkat android, aplikasi EDF mempunyai 3 algoritma yang dapat digunakan untuk mengamankan file, yaitu AES, *Blowfish* dan DES.

Percobaan pertama menggunakan openssl yang dijalankan dengan menggunakan cmd. Hasil pengujian menggunakan openssl dapat dilihat pada Gambar 9.

```
OpenSSL> enc -bf-cbc -d -in sample-flv-file.flv.X -out sampleplain.flv
enter bf-cbc decryption password:
bad magic number
error in enc
```

Gambar 9. Pengujian Uji Keamanan Menggunakan OpenSSL

Hasil menunjukkan “bad magic number” yang berarti proses dekripsi file gagal dilakukan. "Bad magic number" merupakan sebuah kesalahan dari sebuah file yang di mana beberapa byte pertama dari file tersebut tidak benar atau tidak sesuai saat memuat file, mengedit file, atau mencoba menjalankan file. Sehingga ketika terdapat ketidaksesuaian byte pada file akan menyebabkan kesalahan pada file tersebut.

Percobaan kedua menggunakan aplikasi enkripsi dekripsi file (EDF) yang dijalankan pada perangkat android. Hasil pengujian menggunakan aplikasi EDF dapat dilihat pada Gambar 10

.

Gambar 10. Pengujian Uji Keamanan Menggunakan Aplikasi EDF

Hasil menunjukkan "pad block corrupted" yang berarti proses dekripsi file gagal dilakukan

Dari pengujian tersebut terlihat bahwa aplikasi yang dibangun pada penelitian ini bersifat berdiri sendiri. Artinya seandainya *attacker* memiliki file video terenkripsi dan mengetahui kunci yang digunakan untuk mengenkripsi *attacker* tidak dapat melihat video aslinya. Sehingga tidak sembarang orang dapat melakukan penyerangan, hanya orang yang memiliki file terenkripsi dan mengtahui kuncinya.

5. KESIMPULAN

Penelitian ini menghasilkan suatu aplikasi untuk mengamankan file video dengan kunci simetris untuk mengamankannya dan menggunakan algoritma *Blowfish* untuk proses enkripsi dan dekripsinya. Aplikasi ini telah diuji dari segi pengujian aplikasi serta pengujian kualitas aplikasi menunjukkan bahwa aplikasi telah sesuai harapan. Selain itu, aplikasi ini dapat mengamankan file video sebelum dikirim ke penerima dengan format video yang berbeda tanpa mengubah isi dari video tersebut.

Pada pengujian waktu proses enkrip dan dekrip. Enkripsi dan dekripsi memiliki waktu yang berbeda, waktu proses dekripsi lebih cepat 8,7% dari waktu proses enkripsi. Perbedaan ukuran file video mempengaruhi lama proses enkripsi, karena semakin banyak bit semakin lama juga proses enkripsi.

Pengujian aplikasi dalam hal ketahanan juga dilakukan, pengujian dilakukan dengan mengenkripsi file video dengan ukuran besar, dengan menggunakan file video berukuran 2,67GB. Dari hasil pengujian ketahanan aplikasi, aplikasi dapat menjalankan enkripsi dan dekripsi walaupun dengan ukuran besar tanpa terjadinya crash pada aplikasi

Dari uji keamanan pada file video yang terenkripsi dapat disimpulkan bahwa file video terbilang aman walaupun terjadi penyerangan, seandainya *attacker* memiliki *ciphertext*dan mengetahui kuncinya juga tidak akan membuat *attacker* mengetahui isi video tersebut.

DAFTAR PUSTAKA

[1] P. Agrawal, S. Sahu, and A. Choudhary, "A New Method of MPEG Video Encryption Using Frame Shuffling," Jun. 2018.

[2] Cisco, "Cisco public Cisco Visual Networking Index: Global Mobile Data Traffic The Cisco® Visual Networking Index (VNI) Global Mobile Data," Feb. 2019.

[3] N. Syahputri, "RANCANG BANGUN APLIKASI KRIPTOGRAFI PENGAMANAN TRANSMISI DATA MULTIMEDIA MENGGUNAKAN ALGORITMA DATA ENCRYPTION STANDARD (DES)," *MAJALAH ILMIAH METHODA*, pp. 57–58, 2019, doi: 10.46880/methoda.Vol9No2.pp57-63.

[4] S. Niblock, "Towards a psychosemiotics of journalism, mental distress and Covid-19," *Social Semiotics*. Routledge, Jun. 22, 2020. doi: 10.1080/10350330.2020.1779456.

[5] P. Deshmukh and V. Kolhe, *Modified AES Based Algorithm for MPEG Video Encryption*. Chennai: Institute of Electrical and Electronics Engineers, 2014.

[6] A. Chadha, S. Mallik, A. Chadha, R. Johar, and M. M. Roja, "Dual-Layer Video Encryption using RSA Algorithm," Apr. 2015.

[7] Q. Han, L. Wang, Y. Lee, and J. Qin, "Video encryption scheme using hybrid encryption technology," 2020.

[8] G. van Wallendael, A. Boho, J. de Cock, A. Munteanu, and R. van de Walle, "Encryption for High Efficiency Video Coding with Video Adaptation Capabilities," Aug. 2013.

[9] N. Fahriani and H. Rosyid, "IMPLEMENTASI TEKNIK ENKRIPSI DAN DEKRIPSI DI FILE VIDEO MENGGUNAKAN ALGORITMA *BLOWFISH*," *Teknologi Informasi dan Ilmu Komputer (JTIIK)*, vol. 6, no. 6, pp. 1–2, Dec. 2019, doi: 10.25126/jtiik.201961465.

[10] D. Abdullah and D. Nugroho Saputro, "IMPLEMENTASI ALGORITMA *BLOWFISH* DAN METODE LEAST SIGNIFICANT BIT INSERTION PADA VIDEO MP4," *Jurnal Pseudocode*, vol. 3, no. 2, p. `1-2, Sep. 2016.

[11] K. Keerthi and B. Surendiran, *Elliptic Curve Cryptography for Secured Text Encryption*. IEEE, 2017.

[12] R. Rahim and A. Ikhwan, "Cryptography Technique with Modular Multiplication Block Cipher and Playfair Cipher," *International Journal of Scientific Research in Science and Technology IJSRST*, vol. 2, no. 6, pp. 1–2, 2016, [Online]. Available: www.ijsrst.com

[13] S. Singha and M. Sen, "Encoding Algorithm Using Bit Level Encryption and Decryption Technique," 2016.

[14] A. Khan, K. Kr.Mishra, N. Santhi, and J. Jayakumari, *A New Hybrid Technique for Data Encryption*. IEEE, 2015.

[15] M. Kumar, R. Satya Sri, G. Katamaraju, P. Rani, N. Harinadh, and C. Saibabu, *File Encryption and Decryption Using DNA Technology*. IEEE, 2020.

[16] S. Chandra, B. Mandal, S. S. Alam, and S. Bhattacharyya, "Content Based Double Encryption Algorithm Using Symmetric Key Cryptography," in *Procedia Computer Science*, 2015, vol. 57, pp. 1228–1230. doi: 10.1016/j.procs.2015.07.420.

[17] P. Teufl, A. Fitzek, D. Hein, A. Marsalek, A. Oprisnik, and T. Zefferer, "Android Encryption Systems," 2014. [Online]. Available: http://code.google.com/p/cryptsetup/wiki/DMCrypt

[18] E. Waly, R. Stmik, and T. Mandiri, "IMPLEMENTASI ALGORITMA *BLOWFISH* UNTUK PRIVACY DATA E-VOTING," 2019.

[19] Y. Rahmatullah and K. M. Shahih, "Block Cipher Menggunakan Permutasi Diagonal dan Feistel Berbasiskan AES-128," 2015.

[20] K. Angela Putri Sembiring, "Implementasi dan Penggunaan Algoritma Base64 dalam Pengamanan File Video," pp. 25–52, 2020.