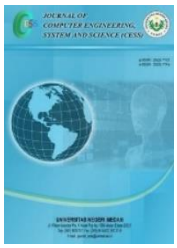
CESS
(Journal of Computing Engineering, System and Science) 7(2) July 2022 355-367

e-ISSN: 2502-714x
p-ISSN: 2502-7131

Contents list available at www.jurnal.unimed.ac.id

CESS
(Journal of Computing Engineering, System and Science)

journal homepage: https://jurnal.unimed.ac.id/2012/index.php/cess



Pengamanan Data Video Streaming Menggunakan Algoritma AES-Rijndael

Video Streaming Data Security Using AES-Rijndael Algorithm

Zebedeus Cheyso[1*], I Made Agus Dwi Suarjaya[2], Gusti Made Arya Sasmita[3]

[1,2,3] Jurusan Teknologi Informasi, Fakultas Teknik, Universitas Udayana
Jl. Raya Kampus UNUD, Bukit Jimbaran, Kuta Selatan, Badung, Bali 80361
email: [1]euscheyso62@gmail.com, [2]agussuarjaya@it.unud.ac.id, [3]aryasasmitha@unud.ac.id

Submitted: 29 Mei 2022 | Revision: 06 Juni 2022 | Accepted: 06 Juli 2022

A B S T R A K

Keamanan merupakan salah satu aspek yang paling penting dalam jalannya proses pertukaran informasi di masa ini. Tidak semua informasi yang dipertukarkan dapat diperbolehkan menjadi konsumsi publik apalagi jika jatuh ke tangan pihak yang tidak bertanggung jawab yang dapat mengubah, memanipulasi dan menyebarkan informasi tersebut tanpa izin. Salah satu bentuk informasi yang perlu dilakukan langkah pengamanan ialah data video *streaming*. Metode enkripsi adalah pilihan yang paling tepat untuk mengamankan kerahasiaan data tersebut. Enkripsi merupakan sebuah ilmu untuk mengkonversikan sebuah data menjadi bentuk yang tidak dapat dimengerti guna menyembunyikan informasi yang hanya dapat diakses oleh pihak yang berhak. Penelitian ini membangun sebuah prototipe untuk mengamankan data video *streaming* dan jalur transmisinya dengan memanfaatkan algoritma enkripsi AES-Rijndael yang yang diterapkan dalam arsitektur *client-server* dengan memanfaatkan protokol transmisi RTP, RTSP dalam kontrol alur media, dan *hashing* SHA-256 untuk memvalidasi setiap *frame* video, enkripsi AES-Rijndael akan digunakan untuk mengubah data dari format *file* video berekstensi Mjpeg. Algoritma AES-Rijndael mampu mengamankan data dan informasi video dengan baik, ketika *client* memasukkan kunci dekripsi yang berbeda, *frame* video yang dikirimkan dari *server* tidak dapat didekripsi dan ditampilkan secara utuh. Pengujian *hashing* SHA-256 dilakukan pada 25 dan 50 *frame* dari video untuk mengecek keaslian data berjalan dengan baik tanpa ada perubahan pada data *frame* yang dikirimkan.

Kata Kunci: *Rijndael; Video Streaming; Enkripsi; Dekripsi; RTP; RTSP.*

A B S T R A C T

Security is one of the most important aspects in the current process of exchanging information. Not all information exchanged can be allowed to become public consumption, especially if it falls into the hands of irresponsible parties who can change, manipulate and

**Peneliti Korespondensi:*
email: euscheyso62@gmail.com

disseminate the information without permission. One form of information that requires security measures is video streaming data. The encryption method is the most appropriate choice to secure the confidentiality of the data. Encryption is a science to convert data into an incomprehensible form in order to hide information that can only be accessed by authorized parties. This study builds a prototype to secure streaming video data and its transmission path by utilizing the AES-Rijndael encryption algorithm which is applied in the client-server architecture by utilizing the RTP transmission protocol, RTSP in media flow control, and SHA-256 hashing to validate each video frame. AES-Rijndael encryption will be used to convert data from video file format with Mjpeg extension. The AES-Rijndael algorithm is able to secure video data and information well, when the client enters a different decryption key, the video frame sent from the server cannot be decrypted and displayed in its entirety. SHA-256 hashing tests were carried out on 25 and 50 frames of video to check the authenticity of the data running well without any changes to the data frames sent.

Keywords: *Rijndael; Video Streaming; Encryption; Decryption; RTP; RTSP.*

1. PENDAHULUAN

Seiring dengan perkembangan jaman yang begitu pesat dalam dunia teknologi informasi mengakibatkan adanya peningkatan resiko keamanan. Banyak data dan informasi membutuhkan sebuah metode pengamanan untuk menjaga kerahasiaan data dalam proses penyimpanan dan pendistribusiannya. Terutama dalam pengamanan data video, karena tidak semua data video bersifat umum. Ada juga data video yang sangat rahasia yang tidak bisa dengan sembarangan dipublikasikan, misalnya data rekaman kamera pengawas dalam *surveillance system* CCTV (*Closed Circuit Television*), dimana data tersebut hanya boleh diakses oleh orang atau perusahaan tertentu dengan keamanan, kerahasiaan, dan keaslian data video yang tetap terjaga. Proses transmisi data video rahasia juga sangat rentan terhadap penyerangan atau pencurian oleh pihak ketiga yang tidak memiliki wewenang untuk mengakses video tersebut seperti menggunakan perangkat lunak EagleEyes yang mampu melihat perekaman video CCTV secara Real-Time hanya dengan terhubung kejaringan lokal yang sama dan mengetahui IP kamera CCTV tersebut. Dalam menjaga kerahasiaan data, terdapat sebuah metode yang dikenal sebagai kriptografi. Kriptografi merupakan seni dalam menyembunyikan informasi dengan merubah informasi tersebut menjadi bentuk yang tidak mudah dipahami oleh siapapun kecuali pihak yang memiliki kunci akses untuk mengembalikan atau menterjemahkan informasi tersebut ke bentuk semula yang dapat dengan mudah dipahami.

Algoritma *rijndael* merupakan salah satu dari sekian banyak algoritma pengamanan data yang diciptakan oleh Vincent Rijmen dan Joan Daemen dari Belgia. Algoritma *riijndael* merupakan algoritma yang memenangkan kontes algoritma yang lebih kuat untuk mengganti algoritma sebelumnya, yaitu DES (*Data Encryption Standard*) [1].

Penelitian ini diterapkan dalam jaringan WLAN (*Wireless Local Area Network*) Wi-Fi dengan arsitektur *Client-Server*, dimana pengguna yang bertindak sebagai *server* akan menjadi *host* yang melakukan *broadcast file* video berekstensi Mjpeg dan melakukan proses enkripsi pada *frame* video yang akan dikirim. *Client* akan bertindak sebagai pengguna yang melakukan *request* pengiriman data *file* video dari *server* yang akan didekripsi dengan kunci dekripsi yang dimasukan. Program aplikasi yang dikembangkan dibagi menjadi 2 program terpisah, yaitu *client* dan *server* yang akan dioperasikan oleh pengguna melalui Laptop atau PC yang sudah

terinstal JDK 8 (*Java Development Kit*). Sebelum menjalankan program aplikasi, baik *client* maupun *server* harus terhubung terlebih dahulu ke jaringan Wi-Fi untuk mempermudah proses transmisi pengiriman *file* media. Protokol RTP dan protokol RTSP yang dirancang khusus untuk mengatur arus media agar lebih efisien. Penelitian ini juga merancang sebuah *interface* khusus di sisi *client* untuk mempermudah mengatur arus data *streaming* dari sisi *server*, sedangkan disisi *server* hanya *interface* berupa jendela *counter* per *frame* yang menampilkan jumlah *frame* yang telah dikirim melalui protokol RTP dan menambahkan fungsi validasi *hashing* SHA-256 dengan tujuan mengecek keaslian dari tiap *frame* yang dikirim antara *client* dan *server*.

Adapun tujuan yang ingin dicapai dalam penelitian ini, yaitu: 1.)Membangun sebuah program aplikasi pengamanan data video *streaming* menggunakan algoritma enkripsi AES-Rijndael berbasis JAVA sebagai bentuk prototipe yang mampu melakukan kontrol alur media dengan protokol RTSP dan RTP sebagai protokol transmisinya dengan menambahkan validasi *hashing* untuk tiap *frame* yang dikirim dan diterima dalam proses *streaming* video, 2.) Mengetahui unjuk kerja dari program aplikasi yang telah dibuat apakah mampu mengamankan *file* video dalam proses *streaming* bersama jalur transmisinya dan menjamin data yang dikirimkan tidak terjadi perubahan serta fungsi fitur-fitur yang ada telah berjalan dengan baik secara keseluruhan.

2. TINJAUAN PUSTAKA

Banyak penelitian mengenai metode pengamanan *file* video dengan memanfaatka algoritma AES-Rijndael. Penelitian-penelitian yang telah dilakukan sebelumnya dimanfaatkan menjadi perbandingan untuk mempermudah proses analisa dan memperkaya pembahasan dalam penelitian yang sedang dilakukan.

Penelitian yang berjudul "Implementasi Algoritma Rijndael untuk Pengamanan pada File Video" oleh Wirman Anugerah Pratama pada tahun 2019, membangun sebuah aplikasi enkripsi untuk mengamankan data visual yang penting khususnya pada *file* video yang berekstensi .MP4 dengan menerapkan algoritma AES-Rijndael yang proses enkripsi dan dekripsinya berfokus pada 16-Byte pertama yang mengandung digital *signature* pada *file* video .MP4 [1]. Yang membedakan dengan program aplikasi yang nantinya akan dibuat adalah proses enkripsi yang dilakukan mencakup keseluruhan *frame* video yang disiarkan oleh *server* dengan ekstensi *file* video yang digunakan adalah .MJPEG. Penelitian sebelumnya menguji video yang sudah di enkripsi maupun yang belum di enkripsi menggunakan aplikasi pemutar video seperti VLC Player, RealOne Player, Media Player, dan GOM Player, sedangkan dalam penelitian ini proses pemutaran video sudah dibangun *interface* khusus di sisi *client* untuk menampilkan video tersebut dan mengetahui apakah video telah mengalami proses enkripsi atau belum.

Penelitian kedua yang berjudul "Penggunaan Algorithma AES-Rijndael pada Sistem Enkripsi dan Dekripsi untuk Komunikasi Data". Penelitian ini bertujuan untuk membangun sebuah program aplikasi yang berfungsi untuk mengenkripsi beberapa jenis file komputer seperti file notepad ".TXT", word document ".DOC", image ".PNG", sound ".MP3" dan video ".FLV" dengan Bahasa pemrograman Visual Basic, penelitian ini sukses dalam proses pengamanan pada *file* yang telah disebutkan sebelumnya, dimana proses dekripsi hanya akan sukses jika *key* dekripsi yang digunakan sama dengan *key* enkripsi [8]. Yang membedakan penelitian ini dengan yang sebelumnya adalah jenis *file* video yang digunakan adalah .MJPEG dan *file* yang dilakukan proses enkripsi dan dekripsi dikirim melalui jaringan nirkabel Wi-Fi dengan protokol

RTSP yang dikhususkan untuk mengatur arus *streaming* media namun tetap memiliki *state* untuk memudahkan ketika video di-*pause* dan *play* dalam proses penyiaran oleh *server* dan menggunakan Bahasa pemrograman JAVA. Penelitian yang sedang dilakukan juga menampilkan hasil dari enkripsi setiap *frame* pada *command line* yang mana potongan *frame* diambil 16-*Byte* pertama untuk dilakukan enkripsi dan menampilkan hasil enkripsi tersebut dalam format ASCII yang menampilkan karakter-karakter yang tidak bisa diterjemahkan dalam bahasa manusia yang dapat dengan mudah dimengerti.

3. METODOLOGI PENELITIAN

Metodologi penelitian berisi tahapan atau gambaran dari penelitian yang telah dilakukan. Berikut adalah gambaran metodologi penelitian yang dibuat dalam bentuk *flowchart* atau diagram alur.

Gambar 1. *Flowchart* Metodologi Penelitian

3.1. Analisa

Analisa penting dilakukan untuk membuat rancangan dalam program aplikasi yang dibangun, dalam hal ini dilakukan perangkuman dari penelitian-penelitian sebelumnya yang berkaitan dengan implementasi algoritma enkripsi AES-Rijndael dan kebutuhan dalam perancangan seperti dari segi *software* dan *hardware* yang akan digunakan.

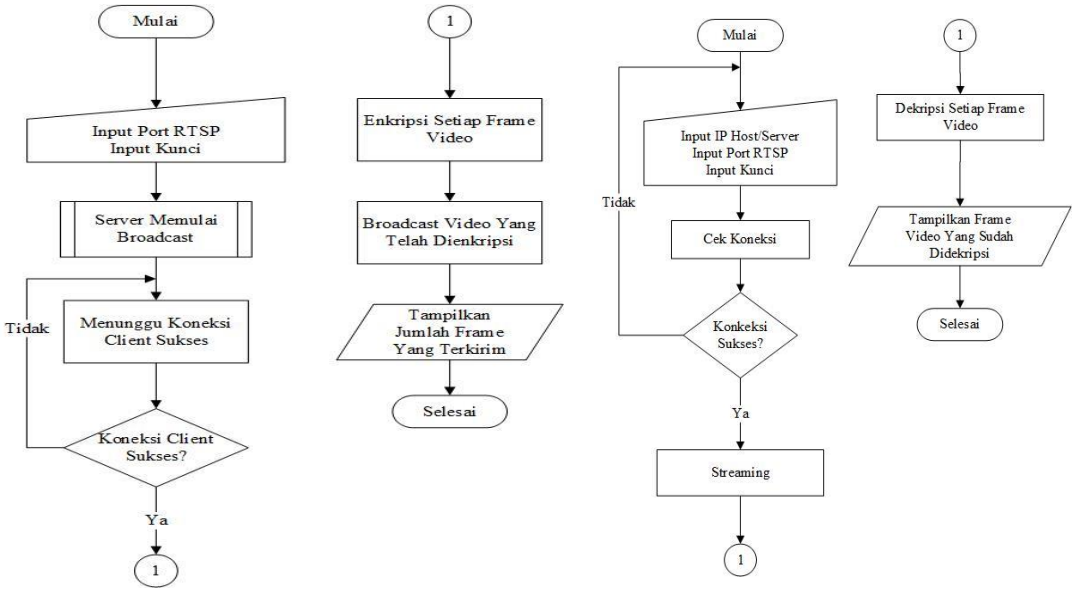
3.2. Perancangan

Tahapan perancangan menentukan bagaimana penggambaran dari program aplikasi yang akan dibuat berbasis *Java* dengan software Visual Studio Code dalam bentuk awal yaitu *mockup*.

(a) (b)

Gambar 2. Rancangan (a) Program Aplikasi *Server* dan (b) Program Aplikasi *Client*

Perancangan aplikasi *server* terdapat 2 jendela utama yaitu jendela yang menampilkan *counter friame* yang dikirimkan dan jendela command line untuk mengeksekusi program, menampilkan *debugging*, menampilkan *frame* yang dikirimkan dan cuplikan Byte data yang dienkripsi. Untuk disisi *client* menampilkan 2 jendela utama juga dimana jendela pertama client berfungsi untuk menampilkan atau memutar video yang diterima dan didekripsi juga memiliki tombol fungsi yang berkaitan dengan kontrol alur media dari protocol RTSP seperti *Setup*, *Play*, *Pause*, dan *Teardown*. Untuk bagian *command line* memiliki fungsi yang sama seperti *command line* pada *server*.

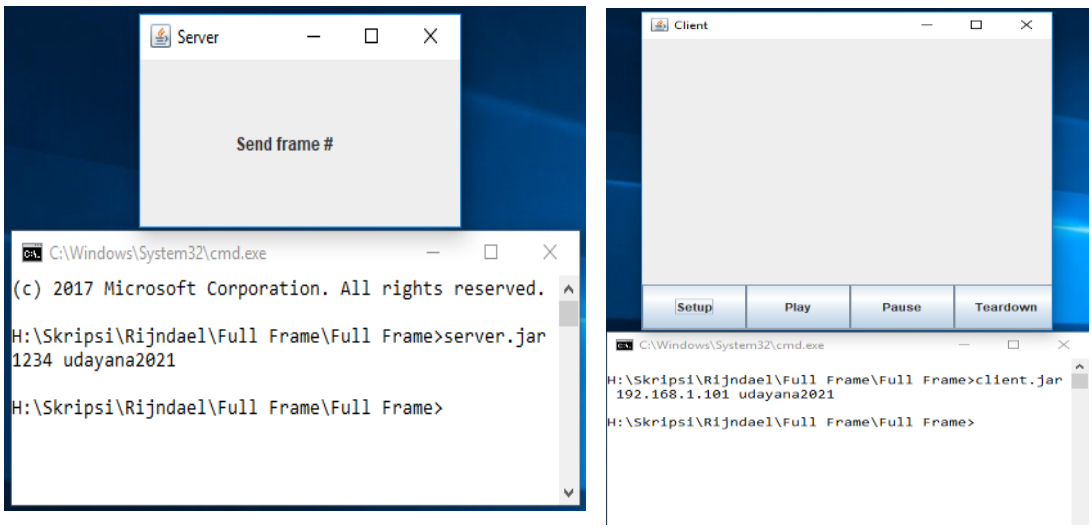


(a) (b)

Gambar 3. Diagram Alir (a) Program Aplikasi *Server* dan (b) Program Aplikasi *Client*

3.3. Implementasi

Dari perancangan yang sudah dilakukan, menghasilkan sebuah program aplikasi client dan server untuk enkripsi dan dekripsi byte frame data video, berikut ini adalah hasil dari rancangan program aplikasi yang dibuat.

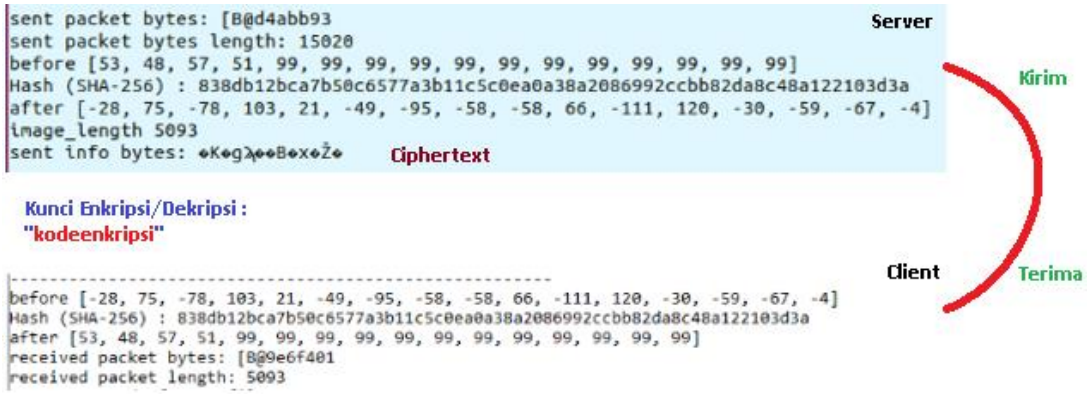


(a) (b)

Gambar 4. Hasil Rancangan (a) Program Aplikasi *Server* dan (b) Program Aplikasi *Client*

3.4. Pengujian

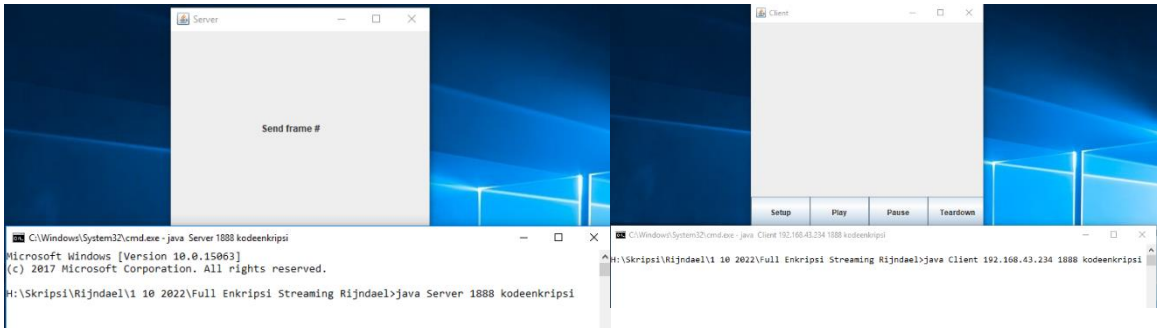
Pengujian program aplikasi dilakukan dengan 2 skenario utama, dimulai dengan *server* yang mengirimkan *file* video Mjpeg yang telah dienkripsi dengan kunci khusus. *Client* akan memasukkan kunci dekripsi yang sama dan berbeda dari *server* lalu melakukan *streaming*. Proses streaming video terenkripsi ditambahkan sebuah fungsi *hashing* untuk menguji adanya perubahan data yang dikirim dan yang diterima antara *client* dan *server* sebagai skenario tambahan.



Gambar 5. Proses Enkripsi dan Dekripsi Server dan Client

Berikut ini merupakan tahapan pengujian berdasarkan scenario yang diuji:

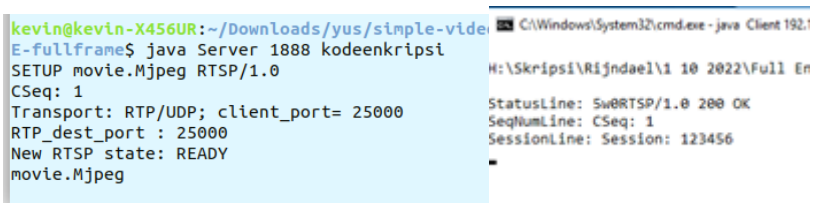
1) Skenario pengujian pengamanan data dengan kunci enkripsi yang sama antara *server* dan *client* dalam proses *streaming* dan fitur-fiturnya.



(a) (b)

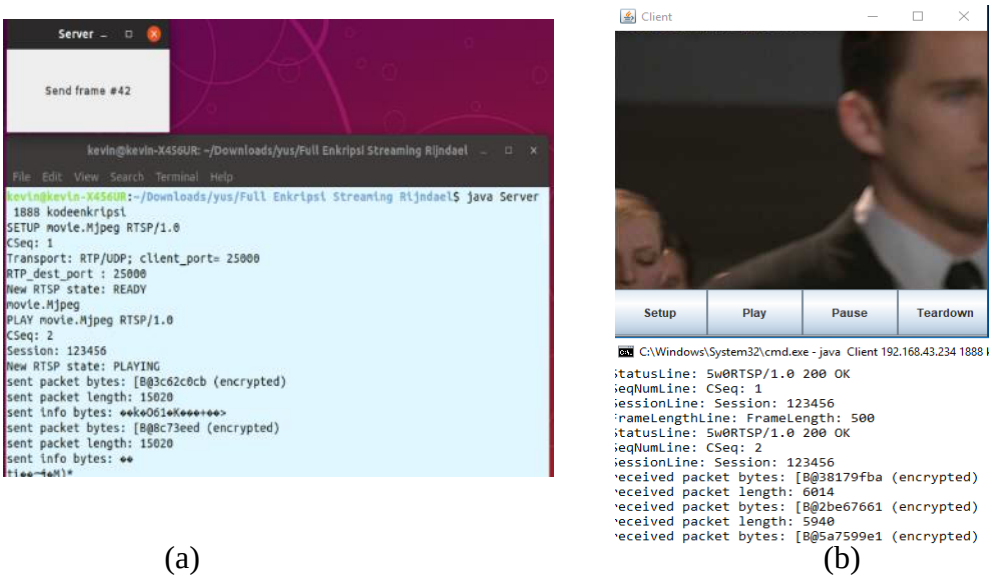
Gambar 6. Memulai (a) Program Aplikasi *Server* dan (b) Program Aplikasi *Client*

Program aplikasi *client* dan *server* dijalankan dari CMD. *Server* akan dimulai dengan sintaks "java Server [PORT] [Encryption Key]" dan *client* akan dijalankan dengan sintaks *Java Client* "[IP Server] [Port] [Decryption Key]", dimana kunci enkripsi dan dekripsi akan disamakan.



(a) (b)

Gambar 7. *Setup* (a) Program Aplikasi *Server* dan (b) Program Aplikasi *Client*



(a) (b)

Gambar 8. Proses *Streaming* (a) Program Aplikasi *Server* dan (b) Program Aplikasi *Client*

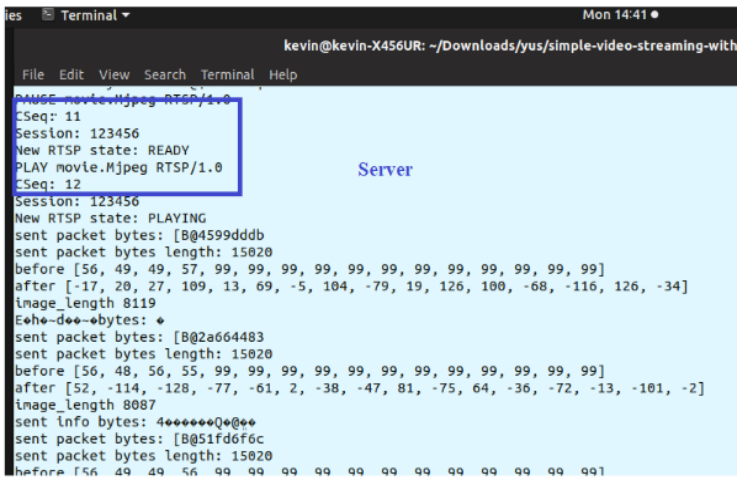
Gambar 8 menampilkan proses *streaming* yang terjadi antara *server* dan client, dimana *client* dapat mendekripsi setiap *frame* video karena kunci dekripsinya sama dengan *server*.

Tabel 1. Informasi Variabel dalam Proses *Streaming*

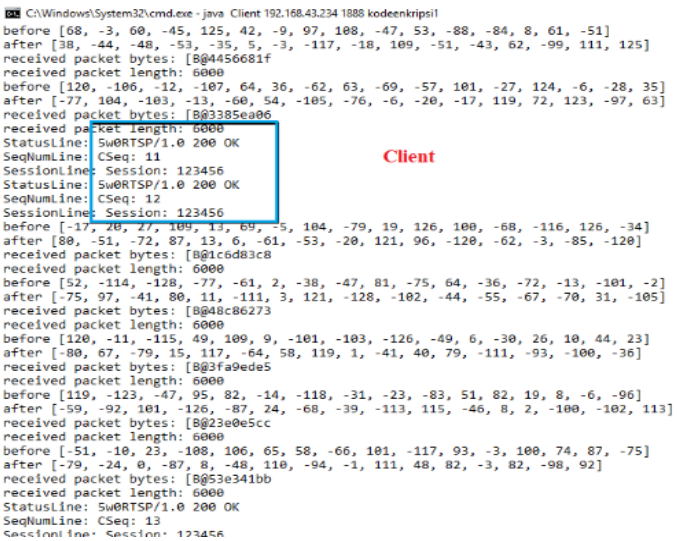
Variabel	Keterangan
Sent packet bytes	Merupakan informasi mengenai *array byte* data *bitstream* (*frame image* video MJPEG) yang dikirimkan oleh *server* dalam bentuk pengkodean ASCII yang sudah dienkripsi.
Sent packet length	Merupakan informasi Panjang dari paket yang telah dikirimkan oleh *server*.
Session	Merupakan informasi sesi yang sedang berjalan saat terjadinya proses *streaming*.
Cseq	Merupakan informasi sekuens dari paket data yang dikirimkan, dimana sekuens ini menampilkan nomor sekuens yang otomatis di-*generate* oleh *client* guna mempertahankan *state* yang sedang aktif dan mempermudah penanganan dalam saat ada permintaan *play/pause* dari *client* agar paket data yang dikirimkan tidak hilang atau kacau saat sampai ke sisi *client*.
RTSP state	Merupakan indikator yang menampilkan status RTSP dalam proses *streaming* seperti *Playing*, *Pause*, dan *Ready*.
Sent info bytes	Merupakan informasi cuplikan *bytes* dari *frame* video yang akan di transfer, Adapun format data yang ditampilkan menggunakan pengkodean ASCII, oleh karena itu ada beberapa simbol yang seperti tanda '?', yang fungsinya adalah *debugging*/membandingkan data yang dikirim *server* dan yang diterima *client*.

(a) (b)

Gambar 9. *Teardown* (a) Program Aplikasi *Server* dan (b) Program Aplikasi *Client*



(a)

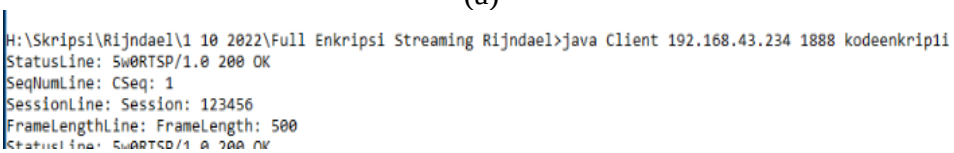


(b)

Gambar 10. Kontrol Media *Play* dan *Pause* (a) Program Aplikasi *Server* dan (b) Program Aplikasi *Client*

2) Skenario pengujian pengamanan data dengan kunci enkripsi yang berbeda antara *server* dan *client* dalam proses *streaming*. Dalam tahap ini, langsung ditampilkan pada proses *streaming* disisi *client* karena dari tahap memulai sampai berakhirnya tetap sama seperti pengujian sebelumnya.

(a)



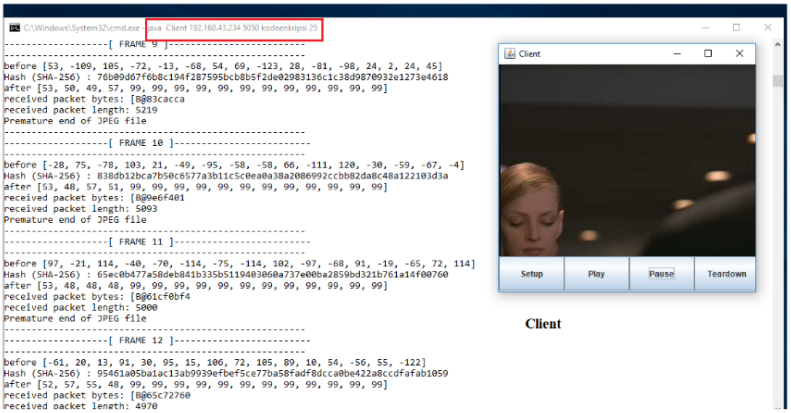
(b)

Gambar 11. *Streaming* (a) Program Aplikasi *Server* dan (b) Program Aplikasi *Client* Dengan Kunci Enkripsi/Dekripsi Berbeda

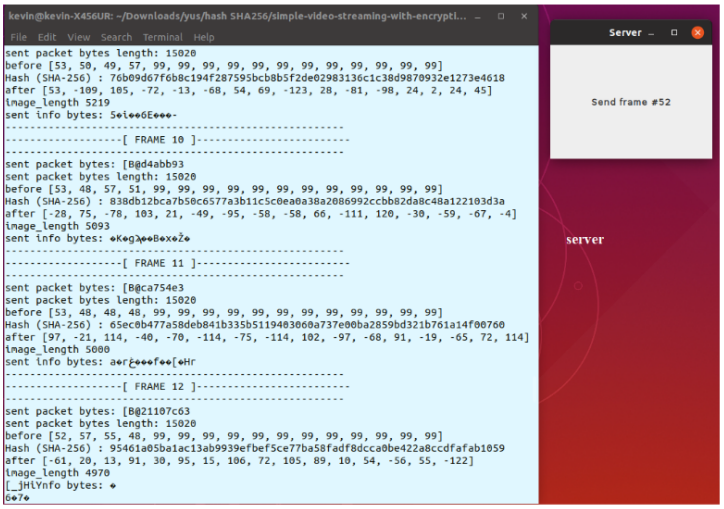
Hasil dari pengujian dengan kunci enkripsi dan dekripsi yang berbeda pada sisi *client* tidak dapat menampilkan gambar atau frame apapun karena proses dekripsi dengan kunci yang

salah disisi *client* juga membuat signature dari *file* .Mjpeg rusak ketika dicoba untuk ditampilkan, namun tampilan command line baik sekuens, *before*/*after* enkripsi dekripsi tetap berjalan seperti saat menggunakan kunci yang benar pada skenario ke-1 diatas.

3) Pengujian pengamanan data dengan validasi *hashing* SHA-256 pada 25 dan 50 *frame* dari video ter enkripsi saat dikirim dari *server* dan diterima disisi *client*.

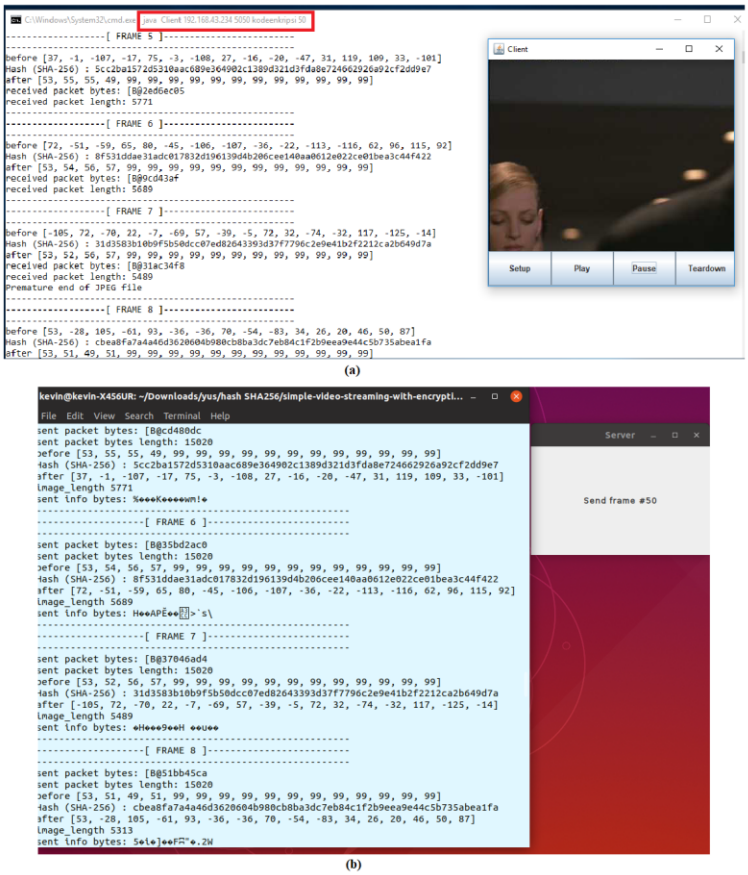


(a)



(b)

Gambar 12. *Hashing 25 Frame* (a) Program Aplikasi *Client* dan (b) Program Aplikasi *Server*



Gambar 13. *Hashing 50 Frame* (a) Program Aplikasi *Client* dan (b) Program Aplikasi *Server*

Validasi *hashing* yang ditambahkan memastikan tidak adanya perubahan data yang dikirimkan *server* dan yang diterima oleh *client*.

4. HASIL DAN PEMBAHASAN

Hasil pengujian yang didapatkan berasal dari pengoperasian dan pengetesan fungsi program yang telah dibangun.

Tabel 2. Hasil Pengujian

Program Aplikasi	Deskripsi	Hasil
Server	Fungsi untuk terhubung kedalam jaringan lokal Wi-Fi melalui *socket* RTP.	*Server* berhasil terhubung ke jaringan Wi-Fi yang sama dengan *client*
	Fungsi menangani setiap *request* atau permintaan dari *client* saat melakukan koneksi dan status khusus seperti fungsi protokol RTSP baik SETUP, PLAY,	*Server* berhasil menangani *request* dari *client* baik *Setup* dengan hasil 200 OK, menangani *request play/pause* dilihat dari jumlah sekuens yang bertambah dan aksi yang dijalankan, *request teardown* berhasil ditangani dengan berakhirnya koneksi antara *server* dan *client*

	PAUSE, dan TEARDOWN.	
	Fungsi melakukan enkripsi pada setiap paket *frame* dari media video Mjpeg.	Fungsi enkripsi berhasil dijalankan dengan dilihat dari *ciphertext* yang ditampilkan pada *commandline* dengan informasi *byte* sebelum dan sesudah enkripsi yang berjumlah 16-Byte (129-bit)
	Fungsi membaca *file* media video yang disediakan.	*Server* mampu membaca file video dengan baik dilihat dari proses setup yang menampilkan nama *file* yang akan dienkripsi dalam kasus ini (Movie.Mjpeg)
Client	Terhubung ke jaringan lokal Wi-Fi melalui *socket* RTP.	*Client* berhasil terhubung ke server melalui jaringan Wi-Fi yang sama dan melakukan koneksi melalui fungsi fitur *setup* dengan hasil 200 OK
	Fungsi *request* atau permintaan ke *server* saat melakukan koneksi dengan protokol RTSP.	Client sukses melakukan request kontrol media seperti *setup*, *play*, *setup*, dan *teardown* dan aksi yang terjadi telah sesuai
	Fungsi membaca setiap *frame* video yang dikirimkan oleh *server*.	*Client* mampu membaca setiap *byte frame* yang dikirimkan *server* dengan melihat variabel *before* dan *after* pada *comand line*
	Fungsi melakukan dekripsi pada setiap paket *frame* dari media video Mjpeg.	*Client* mampu melakukan dekripsi *byte frame* data video yang dikirimkan *server* dengan kunci enkripsi/dekripsi yang dimasukan dilihat dari *command line* yang menampilkan *before* dan *after* setelah *byte frame* terdekripsi.
Client & Server	Fungsi *Hashing* paket *frame* yang dikirim dan diterima	Fungsi hashing SHA-256 yang ditambahkan sebagai validasi tiap *byte frame* berhasil dijalankan sebelum enkripsi dan setelah dekripsi paket yang dikirim (*server*) dan diterima (*client*)

Secara singkat, hasil pengujian yang dilakukan dengan skenario kunci benar dan salah, menunjukan video dapat diputar jika kedua kunci enkripsi/dekripsi dari server dan client sama, dan video tidak dapat diputar jika kunci enkripsi/dekripsi berbeda.

Hashsing SHA-256 yang ditambahkan menjadi fungsi validasi berjalan dengan baik disisi *server* dan *client*, hasil dari *hashing* 25 dan 50 paket *frame* menunjukan keseluruhan tiap *frame* yang dienkripsi dan dekripsi tidak terjadi adanya perubahan data *frame* atau paket yang rusak.

Fungsi kontrol alur media baik *Setup, Play, Pause* dan *Teardown* berjalan dengan baik, bisa dilihat dari jumlah *sequence* yang bertambah ketika fitur-fitur tersebut dijalankan atau diklik.

5. KESIMPULAN

Dari pembahasan dan pengujian dari penelitian ini, dapat disimpulkan bahwa, program aplikasi *server* mampu melakukan proses pengamanan dengan enkripsi AES-Rijndael pada tiap-tiap frame video yang dikirimkan ke *client* dengan kunci enkripsi yang sudah ditetapkan, walaupun *client* atau siapapun yang ingin membaca data tersebut dengan metode yang sama

tetap tidak akan berhasil karena harus memiliki kunci dekripsi yang sama dengan *server*. *Server* dan *client* sukses terhubung melalui protokol transmisi RTP dan melakukan kontrol alur media dengan baik melalui protokol RTSP. *Client* mampu melakukan dekripsi dan *streaming* video Mjpeg yang dikirim oleh *server*. Program aplikasi yang dibangun mampu menjaga keaslian data dengan pengujian *hashing* pada *frame* yang dikirim dan diterima dengan baik. Sistem yang dirancang sudah sesuai dan berjalan dengan baik secara keseluruhan berdasarkan hasil pengujian setiap fitur yang terdapat pada program aplikasi.

REFERENSI

[1] Wirman Anugerah Pratama, “Implementasi Algoritma Rijndael Untuk Pengamanan Pada File Video”, Pelita Informatika: Informasi Dan Informatika, 2019, Vol. 7, No. 4.
[2] Munir And Renaldi, “Kriptografi”, Diktat Kuliah IF5054, Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung, Bandung, 2006.
[3] Eko Kurniawan, ‘Analisis Kualitas Real Time Video Streaming Terhadap Bandwidth Jaringan Yang Tersedia’. Singuda ENSIKOM, 2014, Vol. 9, No. 2.
[4] B. A Prasetiya, ‘Pengaruh Video Bit-rate dan Background Traffic Terhadap Kinerja Video Streaming pada Jaringan Wireless LAN’, Skripsi S. Kom.Institut Pertanian Bogor, 2008.
[5] Afaq and Iqbal, “Analyzing Impact of Video Codec, Encapsulation Methods and Streaming Protocols on the Quality of Video Streaming”. 978-1-4799-0615-4/13/$31.00 ©2013 IEEE, April 2013.
[6] Hartanto, K. W., Gideon, S. S & Handoko, ‘Implementasi Real Time Streaming Protocol Untuk Aplikasi Radio Internet’, Techné Jurnal Ilmiah Elektroteknika, Vol. 8, No.1, 2009.
[7] D. T. Vo and T. Q. Nguyen, "Quality Enhancement for Motion JPEG Using Temporal Redundancies," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 18, no. 5, pp. 609-619, May 2008, doi: 10.1109/TCSVT.2008.918807.
[8] M. Amarullah, A. & Suprianto, A, " Penggunaan Algorithma AES-Rijndael pada Sistem Enkripsi dan Dekripsi untuk Komunikasi Data," Jurnal Penelitian dan Pengkajian Sains dan Teknologi. Vol. 25, No. 2, 2015
[9] D. E. Kurniawan and S. Fani, ‘Perancangan Sistem Kamera Pengawas Berbasis Perangkat Bergerak Menggunakan Rasberry Pi’, Jurnal Ilmiah Teknologi Informasi Terapan, vol. III, no. 2, April 2017.
[10] D. E. Kurniawan and Narupi, ‘Teknik Penyembunyian Data Menggunakan Kombinasi Kriptografi Rijndael dan Steganografi Least Significant Bit (LSB)’, Jurnal Teknik Informatika dan Sistem Informasi, vol. 2, no. 3, Desember 2016.