JURNAL ILMIAH TEKNOLOGI DAN REKAYASA
Vol. 31, No. 2, August 2026, 153 – 166
p-ISSN: 1410-9093, e-ISSN: 2089-8088
https://ejournal.gunadarma.ac.id/index.php/tekno | tekrek@gunadarma.ac.id

Perancangan sistem keamanan audio streaming menggunakan *adaptive bitrate* dengan enkripsi AES-128

[1*]Firman Al Islami, [2]Ega Hegarini

[1,2]Manajemen Sistem Informasi, Universitas Gunadarma
[1,2]Jl. Margonda Raya No. 100, Depok 16424, Jawa Barat, Indonesia
[1]firmanalislami@gmail.com, [2]hegarini@gmail.com

Abstract

The growing demand for fast, stable, and secure audio streaming services has encouraged the development of systems that can adjust audio quality while maintaining content confidentiality. This study develops an HTTP Live Streaming (HLS)-based system that integrates adaptive bitrate streaming (ABR) with Advanced Encryption Standard 128-bit (AES-128) encryption. The system provides four bitrate levels: 32, 64, 128, and 256 kbps. Each audio segment is encrypted before distribution and decrypted by the player using a valid key. Testing was conducted using two MP3 files, while audio quality was evaluated using the Mean Opinion Score (MOS) method involving 30 respondents. The results show that the system successfully adjusted the bitrate according to network conditions, and the encrypted segments could not be played without a valid decryption key. The average MOS scores were 1.60 at 32 kbps, 3.00 at 64 kbps, 4.70 at 128 kbps, and 4.93 at 256 kbps. Therefore, the integration of ABR and AES-128 provides an adaptive audio streaming service while maintaining data confidentiality during distribution.

Keywords: adaptive bitrate, audio streaming, data security, encryption.

Abstrak

Peningkatan kebutuhan layanan *audio streaming* yang cepat, stabil, dan aman mendorong pengembangan sistem yang mampu menyesuaikan kualitas audio sekaligus menjaga kerahasiaan konten. Penelitian ini merancang sistem berbasis HTTP *Live Streaming* (HLS) yang mengintegrasikan *adaptive bitrate streaming* (ABR) dan enkripsi *Advanced Encryption Standard* 128-*bit* (AES-128). Sistem menyediakan empat variasi *bitrate*, yaitu 32, 64, 128, dan 256 kbps. Setiap segmen audio dienkripsi sebelum didistribusikan dan didekripsi oleh pemutar menggunakan kunci yang valid. Pengujian dilakukan terhadap dua *file* MP3, sedangkan kualitas audio dievaluasi menggunakan metode *Mean Opinion Score* (MOS) yang melibatkan 30 responden. Hasil pengujian menunjukkan bahwa sistem berhasil menyesuaikan *bitrate* berdasarkan kondisi jaringan dan segmen terenkripsi tidak dapat diputar tanpa kunci dekripsi yang valid. Nilai rata-rata MOS yang diperoleh adalah 1,60 pada 32 kbps, 3,00 pada 64 kbps, 4,70 pada 128 kbps, dan 4,93 pada 256 kbps. Dengan demikian, integrasi ABR dan AES-128 mampu menghasilkan layanan *audio streaming* yang adaptif sekaligus menjaga kerahasiaan data selama distribusi.

Kata Kunci: *adaptive bitrate*, audio *streaming*, enkripsi, keamanan data.

1. Pendahuluan

Perkembangan teknologi informasi dan komunikasi telah mendorong transformasi besar dalam segmen industri media *streaming*. Salah satu jenis konten yang semakin populer adalah audio *streaming*, yang banyak digunakan pada layanan *podcast*, musik, dan siaran radio

* Corresponding Author

digital. Peningkatan penggunaan layanan tersebut menuntut sistem yang mampu memberikan kualitas audio secara stabil sekaligus menjaga kerahasiaan data selama proses distribusi.

Teknologi *adaptive bitrate streaming* (ABR) memungkinkan kualitas audio disesuaikan secara dinamis berdasarkan kondisi jaringan pengguna sehingga dapat meningkatkan pengalaman pengguna [1]. Namun, proses distribusi audio melalui jaringan masih menghadapi risiko akses langsung oleh pihak yang tidak memiliki kunci dekripsi. Oleh karena itu, diperlukan mekanisme enkripsi untuk menjaga kerahasiaan data audio selama proses distribusi.

Algoritma *Advanced Encryption Standard* 128-*bit* (AES-128) dipilih karena menggunakan kunci sepanjang 128 bit serta memiliki proses enkripsi dan dekripsi yang efisien. AES-128 menjalankan 10 putaran enkripsi sehingga memiliki beban komputasi yang relatif lebih ringan dibandingkan AES-192 dan AES-256 yang masing-masing menjalankan 12 dan 14 putaran [2].

Sejumlah penelitian sebelumnya telah menerapkan AES-128 untuk mengamankan berbagai jenis data. Penelitian pada SMP Yapipa Skanika menunjukkan bahwa AES-128 berhasil diterapkan untuk mengenkripsi *file* berbasis *web*. *File* yang telah dienkripsi hanya dapat dikembalikan ke bentuk semula melalui proses dekripsi menggunakan kunci yang sesuai [3].

Penelitian lain menerapkan AES-128 pada aplikasi registrasi pasien “Si-Periksa”. Hasil penelitian tersebut menunjukkan bahwa proses enkripsi dan dekripsi berjalan sesuai dengan yang diharapkan. Data terenkripsi dapat dikembalikan ke bentuk aslinya tanpa kehilangan informasi sehingga membuktikan bahwa algoritma tersebut berhasil diterapkan untuk melindungi data pada aplikasi registrasi pasien [4].

Penerapan AES-128 pada data audio juga telah dilakukan melalui pengujian proses enkripsi dan dekripsi suara. Hasil penelitian tersebut menunjukkan bahwa informasi suara berhasil disamarkan melalui proses enkripsi dan dapat dipulihkan kembali melalui proses dekripsi. Evaluasi dilakukan berdasarkan perubahan kualitas suara menggunakan nilai *Signal-to-Noise Ratio* (SNR) serta perubahan ukuran *file* audio sebelum dan setelah proses enkripsi dan dekripsi [5].

Penelitian lainnya menggunakan AES-128 dan bahasa Python untuk mengembangkan aplikasi enkripsi dan dekripsi *file* rekam medis berformat PDF. Hasil penelitian menunjukkan bahwa proses enkripsi dan dekripsi berhasil dilakukan. Isi *file* hasil dekripsi, termasuk teks, gambar, dan elemen lainnya, dapat dipulihkan sesuai dengan *file* asli tanpa mengalami kerusakan atau perubahan data [6].

Penerapan AES juga telah digunakan pada berbagai jenis data dan lingkungan sistem lainnya. AES telah diterapkan untuk melindungi data selama proses transmisi [7], pengamanan *file* teks [8], dan basis data [9]. Pada konteks audio, penelitian [10] menganalisis penggunaan variasi *bitrate* dalam proses pengkodean audio, sedangkan penerapan AES pada berbagai jenis dokumen digital telah dilakukan pada penelitian [11]–[14]. Hasil penelitian tersebut menunjukkan luasnya penerapan AES dan pentingnya *bitrate* dalam pengolahan audio. Namun, penelitian-penelitian tersebut belum membahas integrasi ABR berbasis HLS dengan enkripsi AES-128 pada konten audio.

Berdasarkan penelitian terdahulu, penerapan AES-128 telah banyak digunakan untuk pengamanan *file*, basis data, dokumen, dan data audio [3]–[6]. Di sisi lain, penelitian [15] menunjukkan penerapan *adaptive streaming* berbasis DASH dalam menyesuaikan kualitas media terhadap kondisi jaringan, sedangkan penelitian [16] menerapkan ABR menggunakan HLS untuk menyesuaikan kualitas media secara dinamis berdasarkan kondisi jaringan pengguna. Penelitian terbaru juga menunjukkan penerapan AES-128 pada segmen HLS untuk menjaga kerahasiaan konten selama proses distribusi [17], sedangkan penelitian lain telah menerapkan algoritma AES pada pengamanan konten *video streaming* [18]. Meskipun

demikian, masih terbatas penelitian yang secara khusus mengintegrasikan HLS-*based* ABR pada konten audio dengan enkripsi AES-128 serta mengevaluasi secara bersamaan kemampuan adaptasi *bitrate*, akses terhadap segmen audio terenkripsi, dan kualitas audio pada beberapa tingkat *bitrate*. Berdasarkan celah tersebut, penelitian ini mengembangkan sistem *audio streaming* berbasis HLS yang menggabungkan mekanisme *adaptive bitrate* dan AES-128 untuk menjaga kerahasiaan segmen audio selama distribusi.

Penelitian ini bertujuan untuk merancang, mengimplementasikan, dan menguji sistem audio *streaming* berbasis HLS yang mengintegrasikan ABR dengan enkripsi AES-128. Sistem menyediakan empat variasi *bitrate*, yaitu 32, 64, 128, dan 256 kbps, serta mengenkripsi setiap segmen audio sebelum didistribusikan.

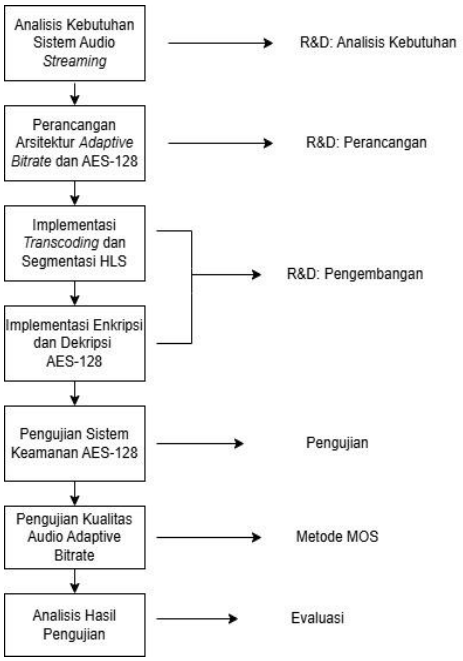
Kontribusi penelitian ini adalah menghasilkan implementasi sistem yang menggabungkan mekanisme adaptasi kualitas audio dan perlindungan kerahasiaan segmen audio dalam layanan *streaming*. Hasil penelitian ini diharapkan dapat menjadi acuan dalam pengembangan layanan distribusi audio yang adaptif dengan perlindungan kerahasiaan konten selama proses distribusi

Kebaruan penelitian ini terletak pada integrasi ABR berbasis HLS dengan enkripsi AES-128 yang secara khusus diterapkan pada sistem audio *streaming*. Penelitian terdahulu mengenai *adaptive streaming* dan HLS telah menunjukkan kemampuan penyesuaian kualitas media secara dinamis berdasarkan kondisi jaringan [15], [16], sedangkan penelitian mengenai AES-128 pada HLS menunjukkan penerapannya untuk menjaga kerahasiaan segmen media selama distribusi [17]. Namun, masih terbatas penelitian yang secara khusus mengintegrasikan kedua mekanisme tersebut pada konten audio sekaligus mengevaluasi kemampuan adaptasi *bitrate*, akses terhadap segmen audio terenkripsi, dan kualitas audio yang dirasakan pengguna. Oleh karena itu, aspek spesifik yang menjadi kebaruan penelitian ini adalah implementasi dan evaluasi terintegrasi HLS-*based* ABR dan AES-128 pada audio dengan empat tingkat *bitrate*, yaitu 32, 64, 128, dan 256 kbps, serta evaluasi kualitas menggunakan metode *Mean Opinion Score* (MOS).

2. Metode Penelitian

Penelitian ini menggunakan metode *Research and Development* (R&D) sebagai pendekatan penelitian untuk menghasilkan dan mengevaluasi produk berupa sistem *audio streaming*. R&D diterapkan pada tahap analisis kebutuhan, perancangan arsitektur, dan pengembangan sistem [19]. Dalam proses pengembangan perangkat lunak, digunakan model *Waterfall* yang mengatur tahapan pengembangan secara sistematis dan berurutan, meliputi analisis kebutuhan, perancangan, implementasi, dan pengujian sistem [20]. Model *Waterfall* dipilih karena kebutuhan sistem telah ditentukan sejak awal, yaitu mengintegrasikan ABR berbasis HTTP *Live Streaming* (HLS) dengan enkripsi AES-128. Setelah sistem dikembangkan, dilakukan pengujian keamanan AES-128 untuk mengetahui kerahasiaan segmen audio, sedangkan kualitas audio pada *bitrate* 32, 64, 128, dan 256 kbps dievaluasi menggunakan metode MOS. Seluruh hasil pengujian kemudian dianalisis dan dievaluasi. Tahapan penelitian ditunjukkan pada Gambar 1.

Data penelitian terdiri atas dua *file* audio berformat MP3 dengan ukuran awal masing-masing 5,7 MB dan 8,9 MB. Durasi masing-masing *file* audio adalah 6 menit 21 detik dan 4 menit 2 detik. Setiap *file* kemudian diproses melalui tahap *transcoding* menjadi empat variasi *bitrate*, yaitu 32, 64, 128, dan 256 kbps.



Gambar 1. Tahapan penelitian

Berdasarkan Gambar 1, pendekatan R&D mencakup tahap analisis kebutuhan, perancangan arsitektur, dan pengembangan sistem. Pengembangan sistem dilakukan menggunakan model *Waterfall*, sedangkan metode MOS digunakan secara khusus pada tahap pengujian kualitas audio. Pengujian keamanan AES-128 dilakukan setelah sistem dikembangkan untuk memastikan bahwa segmen audio terenkripsi tidak dapat diputar tanpa kunci dekripsi yang valid. Hasil pengujian keamanan dan kualitas audio selanjutnya dibahas pada tahap analisis dan evaluasi.

2.1 Analisis Kebutuhan

Analisis kebutuhan dilakukan untuk mengidentifikasi apa saja yang dibutuhkan dalam membangun sistem keamanan audio *streaming* berbasis web. Kebutuhan sistem dibagi menjadi dua, yaitu kebutuhan fungsional dan nonfungsional.

Kebutuhan fungsional meliputi:

1. *streaming* audio dengan *adaptive bitrate*, sistem harus mampu menyesuaikan kualitas audio secara otomatis berdasarkan kecepatan koneksi pengguna menggunakan teknologi HLS.
2. enkripsi dan dekripsi audio, sistem harus mengenkripsi setiap segmen audio di sisi *server* menggunakan AES-128 sebelum dikirim untuk *streaming*.
3. manajemen *file* audio, sistem harus mampu menangani *file* audio dalam beberapa kualitas (*bitrate*) dan mengatur *playlist* .m3u8 yang sesuai.
4. pemutar audio berbasis web, pengguna dapat memutar audio langsung melalui browser dengan antarmuka interaktif.

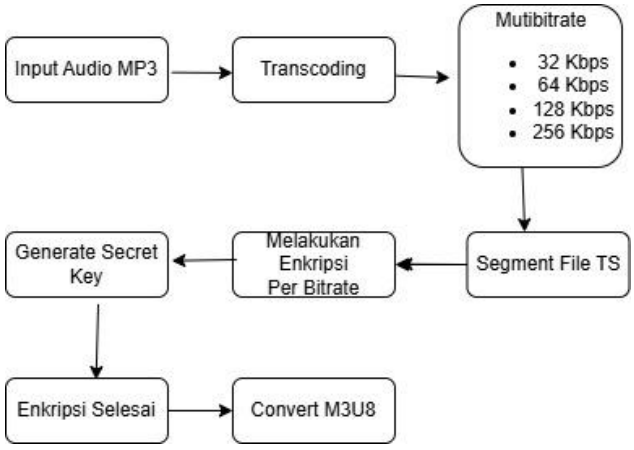
Kebutuhan nonfungsional terdiri dari:

1. keamanan *file* audio yang dikirim ke klien harus dienkripsi dengan standar AES-128.
2. sistem harus mampu menyesuaikan kualitas audio saat terjadi perubahan *bandwidth* dan pemutaran harus tetap lancar tanpa *buffering* berlebihan.
3. aplikasi dapat diakses melalui *browser* modern seperti Chrome, Firefox, dan Edge, baik di desktop maupun perangkat *mobile*.
4. sistem dapat dikembangkan lebih lanjut untuk mendukung lebih banyak pengguna atau integrasi dengan layanan *cloud storage*.

2.2 Perancangan Sistem

Perancangan sistem keamanan *streaming* audio ini dilakukan dengan pendekatan *client-server* berbasis web. Sistem dirancang agar pengguna dapat memutar audio secara *streaming* melalui *website*, dengan perlindungan kerahasiaan konten menggunakan enkripsi AES-128 dan penyesuaian *bitrate* secara otomatis dalam beberapa tahapan berikut:

1. Mekanisme pembagian *bitrate*.
2. Integrasi modul enkripsi dan *streaming.*
3. Membuat tampilan pemutar audio yang akan digunakan untuk *streaming.*
4. Merancang *transcoding* audio.



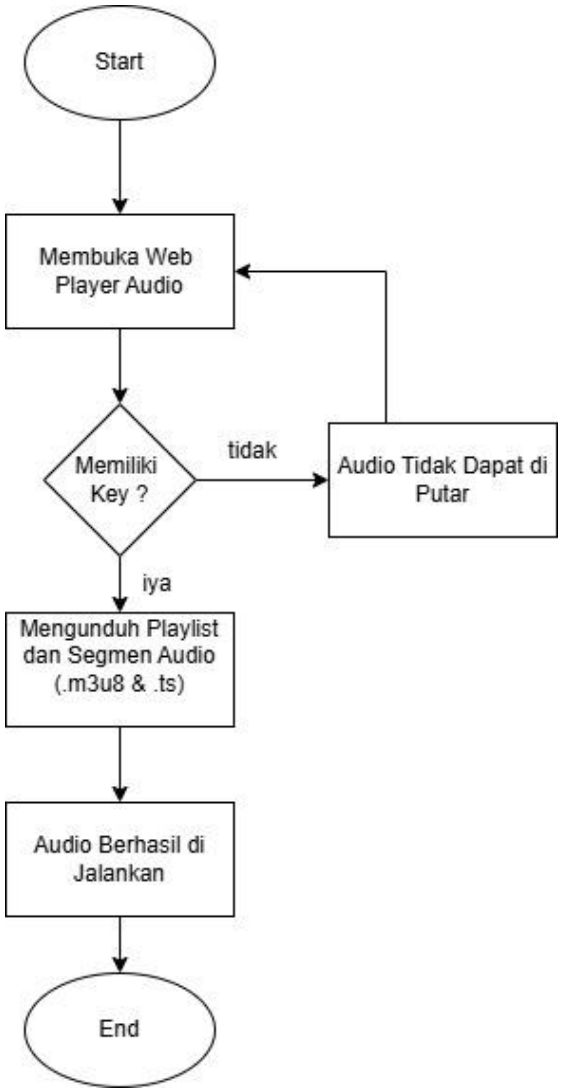
Gambar 2. Alur proses enkripsi audio

Alur proses ditunjukkan pada Gambar 2. Sistem mengintegrasikan mekanisme ABR dan enkripsi AES-128 pada layanan audio *streaming* berbasis HLS. Proses diawali dengan unggah *file* audio berformat MP3 ke server untuk diproses menggunakan FFmpeg. Pada tahap ini, audio diproses melalui *transcoding* menjadi beberapa variasi *bitrate*, yaitu 32 kbps, 64 kbps, 128 kbps, dan 256 kbps, untuk mendukung mekanisme *adaptive bitrate* sesuai kondisi jaringan pengguna.

Setelah proses *transcoding* selesai, sistem melakukan segmentasi audio menjadi beberapa *file* dengan format file (.ts) dan menghasilkan *playlist* (.m3u8) sebagai media pengelolaan distribusi konten. Setiap segmen audio kemudian dienkripsi menggunakan algoritma AES-128 untuk meningkatkan kerahasiaan data selama proses transmisi. Informasi lokasi kunci dekripsi dicantumkan pada *playlist* HLS melalui tag EXT-X-KEY sehingga *player* dapat memperoleh kunci yang sesuai untuk mendekripsi segmen audio terenkripsi.

Pada sisi pengguna, HLS.js secara otomatis melakukan estimasi *bandwidth*, memantau kondisi *buffer*, serta mengevaluasi stabilitas koneksi jaringan untuk menentukan *bitrate* yang paling sesuai. Mekanisme *adaptive bitrate* memungkinkan sistem menyesuaikan kualitas audio secara dinamis sehingga kontinuitas layanan tetap terjaga pada berbagai kondisi jaringan.

Pada Gambar 3 ditunjukkan alur penggunaan kunci dekripsi pada proses pemutaran audio terenkripsi. Segmen audio dapat didekripsi dan diputar ketika *player* memperoleh kunci dekripsi yang sesuai. Mekanisme tersebut berfokus pada perlindungan kerahasiaan konten dan tidak merepresentasikan mekanisme autentikasi atau otorisasi pengguna. Skema tersebut dirancang agar tetap terintegrasi dengan *adaptive bitrate* berbasis HLS, sehingga perlindungan kerahasiaan konten dan mekanisme adaptasi kualitas audio dapat diterapkan secara bersamaan.



Gambar 3. Alur proses memutar audio

2.3 Implementasi

Implementasi sistem dilakukan menggunakan *framework* Laravel berbasis PHP pada sisi server. Proses *transcoding* audio menggunakan FFmpeg untuk menghasilkan beberapa variasi *bitrate*, yaitu 32 kbps, 64 kbps, 128 kbps, dan 256 kbps. Audio dipecah menjadi beberapa segmen HLS (.ts) dan *playlist* (.m3u8). Setiap segmen audio dienkripsi menggunakan AES-128 dengan kunci enkripsi yang dihasilkan oleh server sebelum didistribusikan kepada pengguna. Informasi lokasi kunci enkripsi disimpan pada *playlist* melalui *tag* EXT-X-KEY sehingga HLS.js dapat mengambil kunci tersebut dan melakukan proses dekripsi selama pemutaran audio berlangsung. Integrasi antara AES-128 dan HLS memungkinkan sistem menjaga kerahasiaan data audio dan mempertahankan mekanisme *adaptive bitrate* dalam menyesuaikan kualitas audio berdasarkan kondisi jaringan pengguna. Lingkungan implementasi dan pengujian sistem yang digunakan dalam penelitian ini terdiri atas:

1. PHP 8.3
2. Sistem Operasi macOS
3. *Framework*: Laravel 12
4. *Transcoding*: FFmpeg 8.1.2
5. *Player*: HLS.js 1.6.16
6. *Browser* Pengujian: Google Chrome dan Mozilla Firefox

2.4 Pengujian

Beberapa parameter yang diuji pada penelitian ini meliputi pengujian fungsional sistem dan pengujian kualitas audio.

2.4.1 Pengujian Fungsionalitas

Pengujian fungsional dilakukan untuk memastikan bahwa sistem *streaming* audio berbasis *adaptive bitrate* dapat berjalan sesuai dengan kebutuhan sistem. Adapun aspek yang diuji pada pengujian fungsional meliputi:

1. Memastikan setiap segmen dan *playlist* dapat dimuat serta diputar dengan baik oleh media *player*.
2. Memastikan sistem mampu melakukan adaptasi *bitrate* secara otomatis ketika terjadi perubahan kualitas jaringan pengguna.
3. Memastikan segmen audio yang telah dienkripsi tidak dapat diputar tanpa menggunakan kunci dekripsi yang sesuai.

2.4.2 Pengujian Kualitas Audio

Pengujian kualitas audio menggunakan metode *Mean Opinion Score* (MOS) [21] dengan melibatkan 30 responden yang terdiri atas 70% laki-laki dan 30% perempuan berusia 22–38 tahun. Responden diminta mengakses *website* yang telah disediakan, mendengarkan audio menggunakan *headset* melalui laptop atau *smartphone* kemudian memberikan penilaian terhadap kualitas audio pada variasi bitrate 32 kbps, 64 kbps, 128 kbps, dan 256 kbps melalui kuesioner Google Form. Penilaian menggunakan skala MOS 1–5. Adapun skala penilaian yang digunakan pada pengujian MOS adalah sebagai berikut:

a. Nilai 5: Tidak terasa adanya penurunan kualitas audio (sama dengan audio asli).
b. Nilai 4: Terasa terdapat penurunan kualitas audio, namun tidak mengganggu.
c. Nilai 3: Terasa terdapat penurunan kualitas audio dan sedikit mengganggu.
d. Nilai 2: Terasa terdapat penurunan kualitas audio dan mengganggu proses mendengarkan.
e. Nilai 1: Kualitas audio sangat mengganggu untuk didengarkan.

Nilai MOS dihitung dengan mengacu pada metode MOS [21] menggunakan Persamaan (1).

$$MOS = \frac{\sum x}{N} \quad (1)$$

dengan $\sum x$ adalah total nilai seluruh responden sedangkan N adalah jumlah responden.

Hasil perhitungan MOS kemudian digunakan untuk menganalisis pengaruh *bitrate* terhadap kualitas audio pada sistem *streaming* audio berbasis *adaptive bitrate* menggunakan metode HLS. Pengujian dalam penelitian ini tidak mencakup pengukuran kuantitatif terhadap latensi, *processing overhead*, *throughput*, dan *startup delay*.

2.5 Indikator Keberhasilan

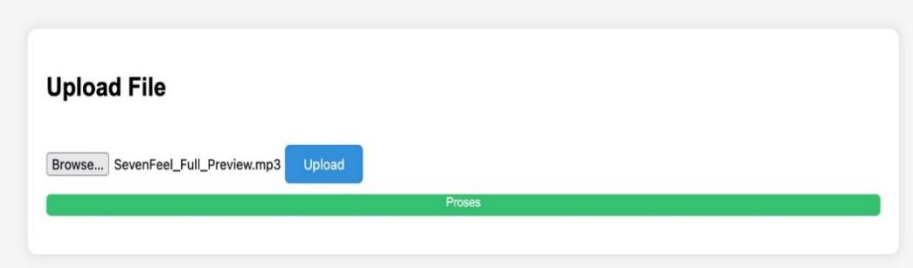
Untuk menilai keberhasilan sistem yang dikembangkan, digunakan beberapa indikator utama yang mencakup aspek teknis, keamanan, dan kualitas pengalaman pengguna. Indikator-indikator tersebut meliputi *file* audio tidak dapat diputar tanpa kunci, dan audio dapat berubah *bitrate* sesuai dengan kecepatan jaringan dari pengguna.

3. Hasil dan Pembahasan

Pada tahap implementasi, proses *transcoding* audio dilakukan pada sisi *server* menggunakan FFmpeg terhadap file audio berformat MP3 yang diunggah ke sistem, seperti ditunjukkan pada Gambar 4. Audio kemudian dikonversi menjadi beberapa variasi *bitrate*,

yaitu 32 kbps, 64 kbps, 128 kbps, dan 256 kbps untuk mendukung mekanisme *adaptive bitrate* pada layanan audio *streaming* berbasis HLS. Audio dipecah menjadi beberapa segmen (.ts) dan menghasilkan *playlist* (.m3u8) yang digunakan untuk mendistribusikan konten audio kepada pengguna.

Setiap segmen audio dienkripsi menggunakan AES-128 sehingga konten audio pada segmen tersebut tidak dapat diputar secara langsung tanpa proses dekripsi menggunakan kunci yang sesuai. Hasil implementasi menunjukkan bahwa integrasi *adaptive bitrate* dan AES-128 dapat berjalan secara bersamaan, sehingga sistem mampu menjaga kerahasiaan distribusi audio serta mempertahankan kemampuan kualitas audio sesuai kondisi jaringan pengguna.



Gambar 4. Proses *transcoding*

3.1 *Transcoding Audio Adaptive Bitrate*

Hasil proses *transcoding* terhadap dua *file* audio sampel menunjukkan bahwa sistem mampu menghasilkan segmen HLS (.ts) dengan empat variasi *bitrate*, yaitu 32 kbps, 64 kbps, 128 kbps, dan 256 kbps seperti pada Tabel 1. Setiap hasil *transcoding* menghasilkan *playlist* (.m3u8) yang digunakan oleh HLS.js untuk mendukung mekanisme *adaptive bitrate*.

Berdasarkan Tabel 1, ukuran *file* hasil *transcoding* cenderung meningkat seiring dengan bertambahnya *bitrate* yang digunakan. Hal ini menunjukkan bahwa *bitrate* yang lebih tinggi menghasilkan jumlah data yang lebih besar sehingga berpengaruh terhadap ukuran *file* yang didistribusikan. Penerapan HLS membagi audio menjadi beberapa segmen (.ts) dan menghasilkan *playlist* (.m3u8), yang turut menambah ukuran data secara keseluruhan.

Mekanisme ini memungkinkan sistem menyesuaikan kualitas audio berdasarkan kondisi jaringan pengguna. Pada kondisi *bandwidth* rendah, sistem dapat memilih *bitrate* yang lebih kecil untuk menjaga kontinuitas layanan, sedangkan pada kondisi jaringan yang lebih baik sistem dapat menggunakan *bitrate* yang lebih tinggi untuk menghasilkan kualitas audio yang lebih optimal.

Tabel 1. Ukuran *file* audio

Nama *file*	*Bitrate*	Ukuran *file* (MB)
69ed6fd24ca90_32k	32 kbps	1.2
69ed6fd24ca90_64k	64 kbps	2.2
69ed6fd24ca90_128k	128 kbps	4.4
69ed6fd24ca90_256k	256 kbps	8.7
69ed70944d33c_32k	32 kbps	1.9
69ed70944d33c_64k	64 kbps	3.4
69ed70944d33c_128k	128 kbps	6.8
69ed70944d33c_256k	256 kbps	13.5

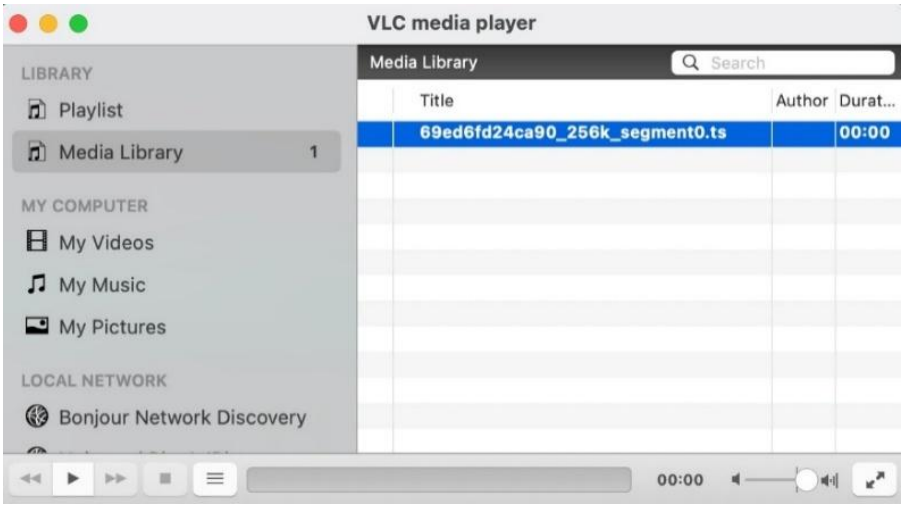
3.2 Enkripsi AES-128

Pada proses ini, sistem menghasilkan *file* enc.key sebagai kunci enkripsi dan enc.keyinfo sebagai konfigurasi yang digunakan oleh FFmpeg untuk mengenkripsi setiap

segmen HLS (.ts) menggunakan AES-128. Lokasi kunci enkripsi disimpan pada *playlist* (.m3u8) melalui *tag* EXT-X-KEY sehingga *player* yang memiliki kunci yang sesuai dapat mendekripsi dan memutar segmen audio yang terenkripsi.

Proses enkripsi menggunakan *Initialization Vector* (IV) 128 bit yang dihasilkan secara acak untuk menginisialisasi proses enkripsi AES-128 pada segmen audio. Seluruh kunci enkripsi disimpan di sisi server.

Hasil pengujian menunjukkan bahwa *file playlist* (.m3u8) dan segmen audio (.ts) masih dapat diakses melalui proses pengunduhan, sebagaimana ditunjukkan pada Gambar 5. Namun, konten audio terenkripsi tidak dapat diputar tanpa kunci dekripsi yang valid. Hasil tersebut menunjukkan bahwa penerapan AES-128 mampu menjaga kerahasiaan konten audio selama proses distribusi dengan mengenkripsi segmen audio sehingga konten tidak dapat langsung diputar tanpa proses dekripsi menggunakan kunci yang sesuai. Pengujian ini terbatas pada aspek kerahasiaan (*confidentiality*) dan tidak mencakup mekanisme autentikasi maupun otorisasi pengguna.

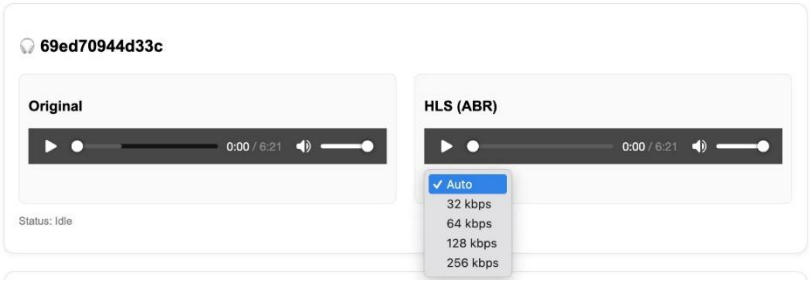


Gambar 5. Pengujian *software* pemutar audio

Temuan ini sejalan dengan penelitian sebelumnya [4]–[6] yang menunjukkan bahwa AES-128 dapat diterapkan untuk mengenkripsi dan melindungi kerahasiaan data digital. Secara lebih luas, penelitian mengenai keamanan multimedia juga menunjukkan bahwa enkripsi merupakan salah satu mekanisme yang digunakan untuk melindungi kerahasiaan konten multimedia dalam lingkungan digital [22]. Berbeda dengan penelitian tersebut, penelitian ini menerapkan AES-128 pada segmen audio dalam layanan *streaming* berbasis HLS yang menggunakan mekanisme *adaptive bitrate*. Hasil pengujian menunjukkan bahwa segmen audio terenkripsi tidak dapat diputar tanpa kunci dekripsi yang sesuai, sementara mekanisme *adaptive bitrate* tetap dapat menyesuaikan kualitas audio berdasarkan kondisi jaringan. Pengujian ini berfokus pada aspek kerahasiaan segmen audio dan tidak mengukur dampak enkripsi terhadap latensi, *processing overhead*, *throughput*, maupun *startup delay*.

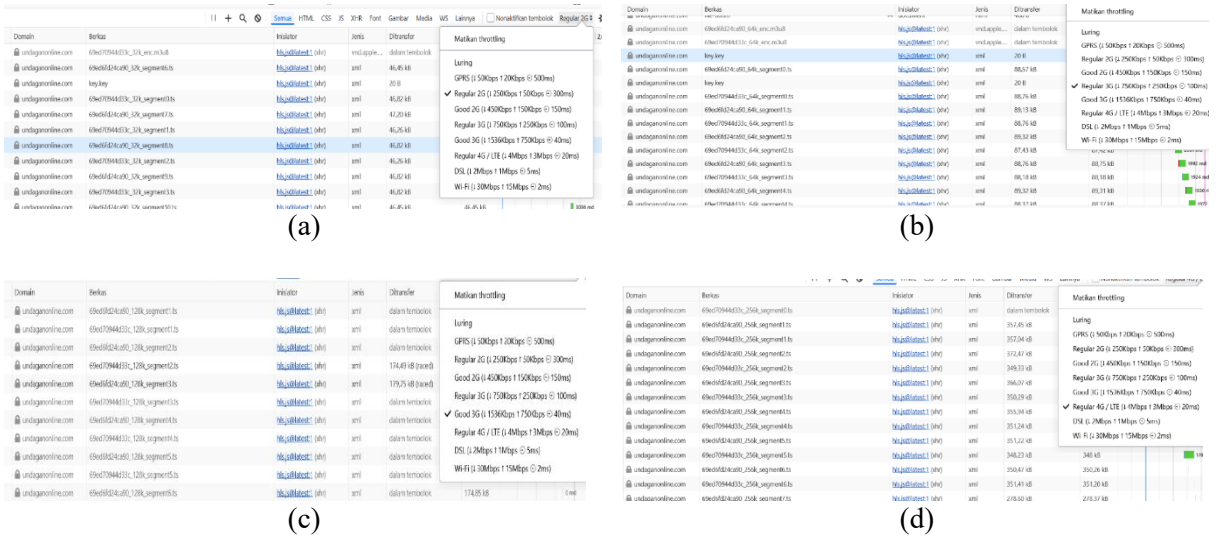
3.3 *Streaming Adaptive Bitrate*

Hasil implementasi menunjukkan bahwa audio *streaming* dapat diputar menggunakan HLS.js yang telah terintegrasi dengan mekanisme validasi kunci enkripsi, seperti ditunjukkan pada Gambar 6. Sistem menyediakan empat variasi *bitrate*, yaitu 32 kbps, 64 kbps, 128 kbps, dan 256 kbps yang dipilih secara dinamis berdasarkan kondisi jaringan pengguna.



Gambar 6. Pemutar audio HLS.js

Pengujian dilakukan menggunakan simulasi *network throttling* pada Google Chrome untuk merepresentasikan beberapa kondisi jaringan, yaitu *Regular* 2G, *Regular* 3G, *Good* 3G, dan 4G, seperti ditunjukkan pada Gambar 7. Hasil pengujian menunjukkan bahwa mekanisme *adaptive bitrate* mampu menyesuaikan kualitas audio dengan kondisi *bandwidth* yang tersedia. Pada kondisi jaringan rendah *Regular* 2G, *player* memilih segmen audio 32 kbps. Ketika *bandwidth* meningkat pada jaringan *Regular* 3G dan *Good* 3G, sistem secara bertahap menggunakan bitrate 64 kbps dan 128 kbps. Pada jaringan 4G, sistem dapat menggunakan *bitrate* tertinggi, yaitu 256 kbps.



Gambar 7. Pengujian jaringan: (a) 2G, (b) 3G, (c) *Good* 3G, (d) 4G

Mekanisme *adaptive bitrate* dilakukan secara otomatis oleh HLS.js berdasarkan estimasi *bandwidth*, stabilitas koneksi, dan kondisi *buffer playback*. Pengguna tidak perlu melakukan pengaturan kualitas audio secara manual karena sistem dapat menyesuaikan kualitas *streaming* secara dinamis. Penelitian mengenai HLS juga menunjukkan bahwa protokol tersebut memiliki kemampuan adaptasi terhadap perubahan kondisi jaringan dalam proses distribusi media [23].

Hasil pengujian menunjukkan bahwa sistem ABR yang dikembangkan mampu memilih kualitas audio berdasarkan kondisi jaringan pengguna. Ketika kondisi jaringan mengalami perubahan, sistem menyesuaikan tingkat *bitrate* yang digunakan agar proses pemutaran audio tetap berlangsung. Hasil ini sejalan dengan penelitian [15] dan [16] yang menunjukkan bahwa mekanisme *adaptive streaming* dapat menyesuaikan kualitas media berdasarkan kondisi jaringan. Penelitian terbaru juga menunjukkan bahwa konfigurasi HLS dan kondisi jaringan berpengaruh terhadap kualitas layanan *live streaming* [24], sedangkan penelitian [25] menunjukkan bahwa algoritma *adaptive bitrate* pada HTTP *Adaptive Streaming* melakukan penyesuaian kualitas media terhadap perubahan kondisi jaringan. Berbeda dengan penelitian

tersebut, penelitian ini menerapkan mekanisme ABR berbasis HLS secara khusus pada konten audio dengan empat variasi *bitrate*, yaitu 32, 64, 128, dan 256 kbps, serta mengintegrasikan enkripsi AES-128 untuk menjaga kerahasiaan segmen audio. Penelitian ini juga mengevaluasi kualitas audio menggunakan metode MOS.

3.4 Hasil Kualitas Audio

Sebanyak 30 responden diminta untuk melakukan penilaian terhadap kualitas audio pada setiap variasi *bitrate* yang tersedia melalui pengisian kuesioner. Pada kuesioner tersebut disediakan daftar URL *website* yang berisi beberapa variasi *bitrate* audio *streaming*. Berdasarkan hasil penilaian yang diberikan oleh responden, selanjutnya dilakukan perhitungan nilai MOS pada masing-masing *bitrate*. Perhitungan dilakukan dengan menghitung rata-rata nilai yang diberikan responden untuk setiap variasi *bitrate* sehingga diperoleh nilai MOS yang digunakan untuk mengetahui tingkat kualitas audio pada setiap *bitrate* audio *streaming*, seperti ditunjukkan pada Tabel 2.

Tabel 2. MOS *bitrate*

Bitrate	Nilai 1	Nilai 2	Nilai 3	Nilai 4	Nilai 5
32 kbps	20	6	2	0	2
64 kbps	0	12	11	2	5
128 kbps	0	0	1	7	22
256 kbps	0	0	0	2	28

Berdasarkan Tabel 2, terlihat adanya pergeseran pola penilaian responden seiring dengan peningkatan *bitrate*. Pada *bitrate* 32 kbps, sebanyak 20 responden (66,7%) memberikan nilai 1 dan 6 responden (20%) memberikan nilai 2, sehingga 86,7% responden menilai kualitas audio pada kategori rendah. Pada 64 kbps, penilaian mulai bergeser ke nilai yang lebih tinggi, dengan 12 responden (40%) memberikan nilai 2 dan 11 responden (36,7%) memberikan nilai 3. Pada 128 kbps, sebanyak 29 dari 30 responden (96,7%) memberikan nilai 4 atau 5, yang menunjukkan bahwa mayoritas responden telah menilai kualitas audio pada kategori baik hingga sangat baik. Sementara itu, pada 256 kbps seluruh responden memberikan nilai 4 atau 5, dengan 28 responden (93,3%) memberikan nilai 5. Pola tersebut menunjukkan bahwa peningkatan *bitrate* menyebabkan distribusi penilaian bergeser dari skor rendah menuju skor tinggi, yang mengindikasikan peningkatan kualitas audio yang dirasakan oleh responden. Berdasarkan distribusi penilaian responden tersebut, selanjutnya dihitung nilai rata-rata MOS untuk setiap tingkat *bitrate*. Hasil perhitungan MOS ditunjukkan pada Tabel 3.

Tabel 3. Skor MOS

Bitrate	**Skor MOS**
32 kbps	1.6
64 kbps	3
128 kbps	4.7
256 kbps	4.93

Nilai MOS pada Tabel 3 menunjukkan pola yang konsisten dengan distribusi penilaian responden pada Tabel 2. Pada *bitrate* 32 kbps, dominasi nilai 1 dan 2 menghasilkan MOS sebesar 1,60. Pada 64 kbps, distribusi penilaian bergeser ke nilai 2 dan 3 dengan MOS sebesar 3,00. Sementara itu, pada 128 kbps dan 256 kbps, dominasi nilai 4 dan 5 menghasilkan MOS masing-masing sebesar 4,70 dan 4,93. Hasil ini menunjukkan bahwa peningkatan *bitrate* diikuti oleh peningkatan kualitas audio yang dirasakan responden.

Selisih MOS antara 128 dan 256 kbps hanya sebesar 0,23, meskipun *bitrate* 256 kbps dua kali lebih tinggi. Berdasarkan ukuran data hasil *transcoding* pada Tabel 1, 128 kbps dapat menjadi pilihan yang lebih efisien dalam konteks penelitian ini karena memberikan kualitas audio yang tinggi dengan kebutuhan data yang lebih rendah. Hal tersebut menunjukkan adanya *trade-off* antara kualitas audio dan kebutuhan data, sehingga mekanisme ABR dapat digunakan untuk menyesuaikan kualitas audio berdasarkan kondisi jaringan pengguna.

4. Kesimpulan

Hasil penelitian menunjukkan bahwa sistem *audio streaming* yang mengintegrasikan ABR berbasis HLS dengan enkripsi AES-128 mampu menyesuaikan *bitrate* audio pada tingkat 32, 64, 128, dan 256 kbps berdasarkan kondisi jaringan pengguna. Enkripsi AES-128 berhasil menjaga kerahasiaan segmen audio karena segmen terenkripsi tidak dapat diputar tanpa kunci dekripsi yang valid. Berdasarkan pengujian MOS terhadap 30 responden, diperoleh nilai rata-rata 1,60 pada 32 kbps, 3,00 pada 64 kbps, 4,70 pada 128 kbps, dan 4,93 pada 256 kbps. Hasil tersebut menunjukkan bahwa peningkatan *bitrate* menghasilkan kualitas audio yang lebih baik, sedangkan ABR memungkinkan sistem menyeimbangkan kualitas audio dan penggunaan *bandwidth*.

Penelitian ini belum mengevaluasi *latency*, *processing overhead*, *throughput,* dan *startup delay*, sehingga pengaruh enkripsi AES-128 terhadap performa *streaming* belum dapat disimpulkan secara menyeluruh. Sistem juga belum dilengkapi mekanisme perlindungan URL. Penelitian selanjutnya dapat mencakup pengujian pada variasi jaringan dan jumlah pengguna yang lebih beragam, evaluasi performa secara kuantitatif, serta penerapan *dynamic key rotation*, autentikasi berbasis token, dan URL dengan masa berlaku terbatas untuk mengurangi risiko akses dan distribusi ulang konten oleh pihak yang tidak berwenang.

Ucapan Terima Kasih

Penulis mengucapkan terima kasih kepada Universitas Gunadarma atas dukungan dan fasilitas yang diberikan selama pelaksanaan penelitian ini.

Pernyataan

Kontribusi Penulis. F.A.I.: konseptualisasi, perancangan metode, implementasi sistem, pengujian, analisis data, dan penulisan draf awal. E.H.: validasi hasil, peninjauan, penyuntingan naskah, serta supervisi penelitian. Seluruh penulis telah membaca dan menyetujui versi akhir naskah.

Pendanaan. Penelitian ini tidak menerima pendanaan khusus dari lembaga pendanaan publik, komersial, maupun nirlaba.

Konflik Kepentingan. Penulis menyatakan tidak terdapat konflik kepentingan terkait publikasi artikel ini.

Ketersediaan Data. Data yang mendukung hasil penelitian ini tersedia dari penulis korespondensi dan dapat diberikan atas permintaan yang wajar.

Penggunaan Kecerdasan Buatan (AI). Penulis menyatakan bahwa alat berbasis kecerdasan buatan (AI) digunakan sebagai bantuan dalam penyuntingan bahasa, pemeriksaan tata tulis, dan dukungan pemrograman selama proses penyusunan naskah. Seluruh isi, analisis, dan kesimpulan penelitian tetap menjadi tanggung jawab penuh penulis.

Daftar Referensi

[1] M. A. Melnyk, N. J. Stavrakos, F. Breg, and A. Penner, "Adaptive bitrate management for streaming media over packet networks," U.S. Patent Application Publication US 2012/0290739 A1, Nov. 15, 2012.

[2] S. P. Ananda, S. Lukman, and I. Irfan, "Analisa metode kriptografi modern Advance Encryption Standard (AES) 128 Bit dalam mengenkripsi dan mendekripsi file dokumen digital," *Jurnal Ilmiah Komputasi*, vol. 21, no. 3, pp. 333–344, 2022, doi: 10.32409/jikstik.21.3.2973.

[3] I. Priambudi and M. Mufti, "Implementasi kriptografi dengan metode AES-128 untuk pengamanan file berbasis web pada SMP Yapipa," *SKANIKA: Sistem Komputer dan Teknik Informatika*, vol. 6, no. 1, pp. 22–31, 2023, doi: 10.36080/skanika.v6i1.2997.

[4] T. A. Ramadhani, A. F. Cobantoro, and S. Sugianti, "Implementasi algoritma Advanced Encryption Standard 128 untuk pengamanan database sistem registrasi pasien," *Jurnal Informatika Polinema*, vol. 10, no. 4, pp. 521–526, 2024, doi: 10.33795/jip.v10i4.5619.

[5] F. Nugraha and T. Arifin, "Enkripsi dan dekripsi suara menggunakan metode AES 128b dengan secret key," *Sistemasi: Jurnal Sistem Informasi*, vol. 11, no. 1, pp. 97–107, Jan. 2022, doi: 10.32520/stmsi.v11i1.1627.

[6] Z. Tamin and B. Hendrik, "Penerapan algoritme Advanced Encryption Standard (AES-128) untuk mengamankan file rekam medis pasien," *Jurnal KomtekInfo*, vol. 12, no. 1, pp. 22–30, 2025, doi: 10.35134/komtekinfo.v12i1.592.

[7] I. Asrowardi, S. D. Putra, E. Subyantoro, and B. P. Zen, "Enhancing IoT data security: AES encryption for protecting data in transit—A case study in smart agriculture," *J. INFOTEL*, vol. 17, no. 4, pp. 839–853, Nov. 2025, doi: 10.20895/INFOTEL.V17I4.1361.

[8] A. Kautsar and M. Ikhsan, "Implementasi algoritma Advanced Encryption Standard (AES) dan teknik steganografi Bit Plane Complexity Segmentation (BPCS) dalam eskalasi keamanan file teks," *Sistemasi: Jurnal Sistem Informasi*, vol. 14, no. 2, pp. 956–968, 2025, doi: 10.32520/stmsi.v14i2.5097.

[9] M. A. Ruswandi and W. Windarto, "Enkripsi database sistem informasi helpdesk dengan algoritme kriptografi AES-128 dan Vigenere chiper," *SKANIKA: Sistem Komputer dan Teknik Informatika*, vol. 5, no. 2, pp. 240–254, Jul. 2022, doi: 10.36080/skanika.v5i2.2957.

[10] A. Ridwan, V. JR, H. Satria, and I. Elfitri, "Analisis kinerja skema MPEG Surround pada pengkodean audio 22 kanal menggunakan bitrate 1000–2000 Kbps," *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, vol. 5, no. 5, pp. 843–852, Oct. 2021, doi: 10.29207/resti.v5i5.3372.

[11] D. D. Rusnadi and N. Juliasari, "Implementasi algoritme AES 128 untuk aplikasi serah terima dokumen project pada PT Telkomsigma," *SKANIKA: Sistem Komputer dan Teknik Informatika*, vol. 4, no. 2, pp. 99–104, Jul. 2021, doi: 10.36080/skanika.v4i2.2205.

[12] A. Ignasius and D. V. S. Y. Sakti, "Penerapan algoritma AES (Advance Encryption Standart) 128 untuk enkripsi dokumen di PT. Gunung Geulis Elok Abadi," *SKANIKA: Sistem Komputer dan Teknik Informatika*, vol. 5, no. 1, pp. 1–10, Jan. 2022, doi: 10.36080/skanika.v5i1.2118.

[13] M. B. Aryanto, M. Tahir, S. I. Devita, Z. N. Mustofa, Q. Ainiyah, and S. Sundoro, "Implementasi enkrip dan dekrip file menggunakan metode Advance Encryption Standard (AES-128)," *Jurnal Ilmiah Sistem Informasi dan Ilmu Komputer*, vol. 3, no. 1, pp. 89–104, Mar. 2023, doi: 10.55606/juisik.v3i1.434.

[14] Y. A. P. Tarigan, R. Aulia, and A. M. Elhanafi, "Algoritma AES 128 dalam mengenkripsikan berkas bansos Kecamatan Tigabinanga berbasis web," *JURNAL UNITEK*, vol. 17, no. 2, pp. 193–204, Jul.–Dec. 2024, doi: 10.52072/unitek.v17i2.943.
[15] D. Kristiadi and Marwiyati, "Adaptive streaming server dengan FFMPEG dan Golang," *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, vol. 5, no. 3, pp. 413–420, 2021, doi: 10.29207/resti.v5i3.2998.
[16] M. R. B. Jaya, W. Andriyani, D. Kristomo, and M. A. Nugroho, "Dynamic bitrate adjustment in web-based video streaming applications using HTTP Live Streaming (HLS)," *Journal of Intelligent Software Systems*, vol. 3, no. 1, pp. 13–21, 2024, doi: 10.26798/jiss.v3i1.1344.
[17] B. S. Sabir and A. A. Mohammed, "Evaluating AES-128 segment encryption in live HTTP streaming under content tampering and packet loss," *Network*, vol. 6, no. 1, art. no. 4, 2026, doi: 10.3390/network6010004.
[18] Z. Cheyso, I. M. A. D. Suarjaya, and G. M. A. Sasmita, "Pengamanan data video streaming menggunakan algoritma AES-Rijndael," *CESS (Journal of Computer Engineering, System and Science)*, vol. 7, no. 2, pp. 355–367, 2022, doi: 10.24114/cess.v7i2.32989.
[19] A. Perdana, S. Dewi, N. A. Farhana, and D. Febrian, "Comparative analysis of SDLC and R&D methods in system development: A case study of integrity zone management system," *Sinkron: Jurnal dan Penelitian Teknik Informatika*, vol. 9, no. 4, pp. 3197–3209, 2025, doi: 10.33395/sinkron.v9i4.15337.
[20] A. Saravanos and M. X. Curinga, "Simulating the software development lifecycle: The waterfall model," *Applied System Innovation*, vol. 6, no. 6, art. no. 108, 2023, doi: 10.3390/asi6060108.
[21] International Telecommunication Union, "Mean Opinion Score (MOS) terminology," ITU-T Recommendation P.800.1, 2016. [Online]. Available: https://www.itu.int/rec/T-REC-P.800.1/en. (Accessed: May 9, 2026).
[22] K. M. Hosny, M. A. Zaki, N. A. Lashin, M. M. Fouda, and H. M. Hamza, "Multimedia security using encryption: A survey," *IEEE Access*, vol. 11, pp. 63027–63056, Jun. 2023, doi: 10.1109/ACCESS.2023.3287858.
[23] S. S. Saini and L. S. Sharma, "Investigation of the HTTP Live Streaming Media Protocol's (HLS) adaptability and performance," *Journal of The Institution of Engineers (India): Series B*, vol. 106, pp. 1081–1089, 2025, doi: 10.1007/s40031-024-01132-w.
[24] B. S. Sabir and A. A. Mohammad, "Improving live streaming QoE through HLS parameter tuning and load balancing to mitigate packet loss," *Kurdistan Journal of Applied Research*, vol. 10, no. 2, pp. 77–92, Aug. 2025, doi: 10.24017/science.2025.2.7.
[25] S. Uddin, M. Grega, M. Leszczuk, and W. ur Rahman, "Evaluating HAS and low-latency streaming algorithms for enhanced QoE," *Electronics*, vol. 14, no. 13, art. no. 2587, 2025, doi: 10.3390/electronics14132587.