

ANALISIS KOMPARATIF KEAMANAN PROTOKOL MQTT MENGGUNAKAN ENKRIPSI AES-256 DAN TRANSPORT LAYER SECURITY (TLS) BERBASIS SOFTWARE-DEFINED SIMULATION

Ahmad Iwan Setiawan, Yohanes Dwi Cahyono, Heri Setiawan

Rekayasa Keamanan Siber, Polteknik Angkatan Darat

Kesatrian Poltekad Jl. Raya Anggrek Desa Pendem, Kec. Junrejo, Kota Batu, Jawa Timur

mhs.cyber8.ahmadiwans@poltekad.ac.id

ABSTRAK

Perkembangan ekosistem *Internet of Things* (IoT) memicu peningkatan risiko keamanan pada jalur transmisi data, khususnya yang mengandalkan protokol *Message Queuing Telemetry Transport* (MQTT). Sifat bawaan MQTT yang mengirimkan data tanpa enkripsi menyebabkan paket informasi rentan terhadap serangan penyadapan (*packet sniffing*). Penelitian ini menguji model pengamanan data IoT secara virtual menggunakan algoritma *Advanced Encryption Standard* (AES-256) mode *Cipher Block Chaining* (CBC) melalui pendekatan *software-defined simulation* di lingkungan Google Colab. Eksperimen dievaluasi secara komparatif melalui tiga skenario utama: tanpa enkripsi (*no_enc*), enkripsi level aplikasi (*aes_only*), serta enkripsi kombinasi jaringan (*aes_tls*). Hasil simulasi menunjukkan bahwa integrasi AES-256 CBC mampu melindungi data secara mutlak dari intersepsi pihak ketiga. Implementasi ini memicu konsekuensi teknis berupa peningkatan ukuran paket data (*size overhead*) sebesar 8,47% hingga 8,59% sebagai dampak dari mekanisme pembulatan blok *padding*. Namun, beban komputasi waktu eksekusi yang dibutuhkan sangat minimal, yakni berada pada rentang 0,012 ms hingga 0,016 ms, sehingga sangat layak diaplikasikan pada infrastruktur IoT.

Kata kunci : AES-256, MQTT, Overhead, Simulasi, TLS.

1. PENDAHULUAN

Pertumbuhan perangkat IoT yang masif membawa konsekuensi serius pada aspek keamanan data. Protokol MQTT, yang menjadi tulang punggung komunikasi IoT karena sifatnya yang ringan dan efisien, ternyata tidak dirancang dengan mekanisme enkripsi bawaan yang memadai [1]. Akibatnya, data sensor yang dikirim antar perangkat berpotensi disadap, dimodifikasi, atau bahkan dimanfaatkan oleh pihak tidak bertanggung jawab melalui teknik seperti *packet sniffing* dan serangan *man-in-the-middle* (MITM)[2].

Risiko ini bukan sekadar teoretis. Penelitian Azwar dan Gary secara empiris membuktikan bahwa tanpa mekanisme pengamanan, seluruh isi pesan MQTT termasuk kredensial autentikasi dan data sensor dapat dibaca secara langsung menggunakan Wireshark. Sementara itu, penelitian terdahulu yang mengkaji keamanan MQTT umumnya berfokus pada efisiensi algoritma enkripsi atau simulasi perangkat lunak semata, tanpa mengevaluasi pengaruh nyata terhadap performa perangkat IoT secara langsung[3]. Di sisi lain, penelitian yang membandingkan tiga skenario keamanan secara sistematis, yaitu tanpa enkripsi, enkripsi AES tanpa TLS, dan enkripsi AES dengan TLS, dalam satu kerangka eksperimen yang terstruktur masih sangat terbatas.

Kesenjangan inilah yang menjadi motivasi penelitian ini. Sebagian besar studi hanya membandingkan ada atau tidaknya enkripsi, tanpa memisahkan secara eksplisit kontribusi masing-masing lapisan keamanan. Padahal perbedaan antara skenario *aes_only* dan *aes_tls* bersifat substansial: AES melindungi isi payload, sedangkan TLS

melindungi seluruh jalur komunikasi termasuk metadata seperti nama topik, alamat broker, dan kredensial koneksi[4]. Tanpa memahami perbedaan ini, praktisi IoT berisiko menganggap enkripsi payload sudah cukup, padahal kanal komunikasinya masih rentan.

Penelitian ini mengimplementasikan dan mengevaluasi mekanisme keamanan protokol MQTT menggunakan algoritma AES-256 *Cipher Block Chaining* (CBC) berbasis simulasi perangkat lunak pada platform Google Colaboratory. Tiga skenario pengujian dibandingkan secara sistematis: (1) tanpa enkripsi sebagai kondisi *baseline*, (2) enkripsi AES-256 CBC tanpa TLS, dan (3) enkripsi AES-256 CBC dikombinasikan dengan simulasi TLS 1.3. Parameter yang diukur meliputi waktu enkripsi, latensi publikasi, total waktu pemrosesan, dan persentase penambahan ukuran data.

Kontribusi utama penelitian ini ada tiga. Pertama, penelitian ini menyediakan perbandingan empiris tiga skenario keamanan MQTT secara sistematis dalam satu kerangka eksperimen yang terkontrol. Kedua, penelitian ini mengklarifikasi perbedaan proteksi antara lapisan payload dan lapisan transport secara eksplisit, yang sering diabaikan dalam literatur. Ketiga, hasil penelitian ini memberikan landasan bagi praktisi IoT yang ingin mengimplementasikan keamanan MQTT pada lingkungan berbasis Python tanpa ketergantungan perangkat keras fisik.

2. TINJAUAN PUSTAKA

2.1. Advanced Encryption Standard (AES).

AES ditetapkan oleh NIST sebagai standar enkripsi resmi yang menggantikan DES, dengan

dukungan tiga panjang kunci: 128-bit, 192-bit, dan 256-bit [5][6][7]. Untuk kunci 256-bit, proses enkripsi berlangsung dalam 14 putaran yang masing-masing terdiri dari empat transformasi: *AddRoundKey*, *SubBytes*, *ShiftRows*, dan *MixColumns*[6].

Persamaan Enkripsi:

$$C_i = E_k(P_i \oplus C_{i-1})$$

(Untuk blok pertama ($i = 1$), gunakan $C_0 = IV$)

Persamaan Dekripsi:

$$P_i = D_k(C_i) \oplus C_{i-1}$$

Keterangan Variabel: 4, 6, 8

- C_i : Ciphertext (blok data tersandi) ke- i .
- P_i : Plaintext (blok data asli) ke- i .
- E_k : Fungsi enkripsi AES menggunakan kunci rahasia k .
- D_k : Fungsi dekripsi AES menggunakan kunci rahasia k .
- $\oplus$: Operasi logika XOR biner.
- IV : Initialization Vector (128-bit) yang digunakan untuk memastikan keunikan blok pertama.

Gambar 1. Rumus enkripsi dan dekripsi AES 256

AES unggul dibanding alternatif lain dalam konteks IoT. AES memiliki waktu enkripsi rata-rata 12 ms, jauh lebih cepat dibanding RSA (98 ms) dan Triple DES (55 ms), dengan konsumsi sumber daya yang lebih rendah[5]. Temuan ini konsisten dengan sifat desain AES yang mempertimbangkan efisiensi pada perangkat keras terbatas, dari kartu pintar 8-bit hingga prosesor modern[3].

2.2. Mode CBC dan Implikasi pada Ukuran Data

Mode *Cipher Block Chaining* (CBC) dipilih dalam penelitian ini karena kemampuannya mengatasi kelemahan mode ECB, di mana blok plaintext yang identik menghasilkan ciphertext yang identik pula. Pada CBC, setiap blok plaintext di-XOR dengan blok ciphertext sebelumnya sebelum dienkripsi, sehingga pola yang berulang dalam data tidak menghasilkan pola yang berulang pada ciphertext. Konsekuensi dari mode CBC adalah penambahan ukuran data akibat mekanisme *padding* PKCS#7. Widodo dan Purnomo [8] melaporkan penambahan ukuran sebesar 25% dari ukuran dokumen asli pada implementasi AES-256, yang disebabkan oleh pembentukan *key schedule* pada proses *key expansion*. Penambahan ini bersifat deterministik dan dapat dihitung, sehingga dapat dipertimbangkan dalam perancangan sistem IoT yang sensitif terhadap bandwidth[6].

2.3. Transport Layer Security (TLS).

TLS bekerja pada dua lapisan: *Handshake Protocol* untuk negosiasi algoritma dan pertukaran kunci menggunakan kriptografi asimetris, serta *Record Protocol* untuk memverifikasi integritas dan kerahasiaan data selama sesi berlangsung. Keduanya bekerja bersama untuk membangun apa yang disebut sebagai *secure tunnel* antara klien dan server[3][8][9].

Kombinasi TLS dan *Digital Signature* dalam konteks aplikasi finansial berbasis *cloud* dan menemukan bahwa keduanya membentuk model keamanan berlapis yang saling melengkapi: TLS

mengamankan jalur komunikasi, sementara enkripsi pada lapisan data mengamankan isi pesan itu sendiri. Analisis ini relevan langsung dengan penelitian MQTT, di mana skenario *aes_only* hanya melindungi payload tetapi membiarkan metadata koneksi terekspos, sedangkan skenario *aes_tls* melindungi kedua lapisan secara bersamaan[4][7][9][10].

2.4. Keamanan MQTT.

MQTT dirancang sebagai protokol komunikasi ringan untuk perangkat dengan keterbatasan sumber daya, menggunakan pola *publish-subscribe* di mana broker berperan sebagai perantara antara *publisher* dan *subscriber* [11]. Keunggulan ini menjadikannya protokol dominan dalam aplikasi IoT, namun juga menghadirkan masalah mendasar: MQTT secara bawaan hanya menyediakan autentikasi berbasis *username* dan *password* tanpa enkripsi lapisan aplikasi [12].

2.5. Penelitian terdahulu.

Riset keamanan data pada protokol MQTT dan ekosistem IoT telah didekati dengan berbagai metode. Suharti dkk. [1] menerapkan stream cipher Grain-128 pada MQTT-SN untuk menekan beban memori pada node sensor medis. Di sisi lain, Akbar dkk. [2] menguji algoritma RSA pada ESP8266 untuk komunikasi *publish-subscribe*, namun karakteristik komputasi asimetris menuntut waktu dekripsi yang lebih panjang. Untuk enkripsi berbasis blok, Rizky dkk. [7] membuktikan keandalan AES mode CBC pada perangkat keras nirkabel, meski menghasilkan penambahan ukuran data akibat padding blok.

Pada level transport, Azwar dan Gary [3] membuktikan bahwa implementasi TLS pada broker MQTT mampu menutupi kerentanan sniffing kredensial dan isi data sensor. Namun, penerapan TLS pada kanal jaringan memiliki tantangan konfigurasi cipher suite dan risiko serangan seperti yang diidentifikasi oleh Dharmawan dkk. [8], serta kebutuhan mitigasi replay attack pada koneksi lanjutan yang dikaji oleh Abdelhafez dkk. [9]. Selain itu, karakteristik pengiriman data MQTT secara periodik juga terbukti sensitif terhadap latensi transmisi jaringan, sebagaimana ditunjukkan pada studi Sadewo dkk. [11] serta Mahfuza dkk. [12].

Sebagian besar riset di atas menguji skenario keamanan secara parsial, yaitu hanya pada lapisan muatan data atau lapisan transport jaringan saja. Perbedaan penelitian ini terletak pada pengujian komparatif tiga skenario sekaligus dalam satu sistem simulasi, yakni tanpa enkripsi, enkripsi payload AES-256 CBC, dan kombinasi payload-transport TLS 1.3. Pendekatan ini memungkinkan analisis batasan proteksi dan trade-off overhead komputasi antar lapisan dilakukan secara lebih terukur.

3. METODE PENELITIAN

Penelitian ini dilaksanakan dalam lima fase secara sistematis. Fase pertama adalah perencanaan,

yang dimulai dengan identifikasi masalah terkait kerentanan keamanan pada protokol MQTT dalam ekosistem IoT, di mana data sensor dikirimkan tanpa mekanisme enkripsi yang memadai sehingga rentan terhadap serangan sniffing. Selanjutnya dilakukan studi pustaka secara komprehensif mencakup algoritma AES-256, protokol MQTT, mekanisme TLS 1.3, serta penelitian terdahulu di bidang keamanan IoT. Fase perencanaan diakhiri dengan perancangan sistem simulasi, meliputi pendefinisian struktur data, alur komunikasi, dan penetapan tiga skenario pengujian yang akan digunakan.

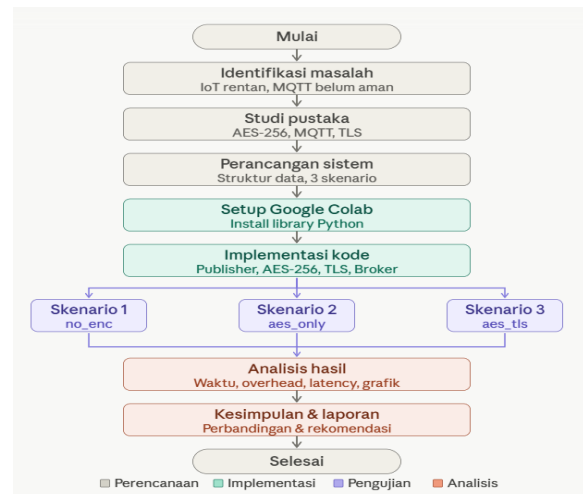
Fase kedua adalah persiapan lingkungan simulasi. Karena penelitian ini tidak menggunakan perangkat keras fisik, seluruh pengujian dilaksanakan secara virtual menggunakan Google Colab sebagai platform komputasi. Pada tahap ini dilakukan instalasi seluruh dependensi yang diperlukan, yaitu pustaka *pycryptodome* untuk implementasi kriptografi AES-256, *paho-mqtt* untuk simulasi komunikasi MQTT, serta *pandas* dan *matplotlib* untuk pengolahan dan visualisasi data hasil eksperimen.

Fase ketiga adalah implementasi kode. Komponen yang dibangun meliputi virtual publisher yang mensimulasikan perangkat ESP32 dengan menghasilkan data suhu dan kelembaban secara acak, modul enkripsi AES-256 mode CBC dengan 14 putaran pemrosesan yang terdiri dari operasi *AddRoundKey*, *SubBytes*, *ShiftRows*, dan *MixColumns*, simulasi TLS *handshake* yang meniru proses pertukaran kunci antara klien dan server, serta simulator broker MQTT berbasis *in-memory* beserta komponen penyerang (*sniffer*) untuk keperluan demonstrasi keamanan.

Fase keempat adalah pengujian, yang dilaksanakan dalam tiga skenario berbeda. Skenario pertama (*no_enc*) menguji pengiriman data tanpa enkripsi sebagai kondisi dasar (*baseline*), di mana seluruh isi pesan dapat dibaca secara langsung oleh pihak yang tidak berwenang. Skenario kedua (*aes_only*) menerapkan enkripsi AES-256 CBC pada payload tanpa perlindungan TLS pada lapisan koneksi, sehingga isi pesan terenkripsi namun metadata MQTT seperti topik dan alamat broker masih dapat terekspos. Skenario ketiga (*aes_tls*) menerapkan perlindungan berlapis dengan mengombinasikan enkripsi AES-256 CBC pada payload dan simulasi TLS 1.3 pada lapisan koneksi, sehingga baik isi pesan maupun metadata komunikasi terlindungi sepenuhnya.

Fase kelima adalah analisis hasil. Parameter yang diukur dan dibandingkan antar skenario meliputi waktu enkripsi (*encrypt_time_ms*), latensi publikasi ke broker MQTT (*publish_time_ms*), total waktu pemrosesan (*total_time_ms*), serta persentase penambahan ukuran data akibat enkripsi (*size_overhead_pct*). Seluruh hasil pengukuran disimpan dalam format CSV dan divisualisasikan menggunakan grafik batang dan grafik garis untuk memudahkan perbandingan antar skenario. Penelitian diakhiri dengan penyusunan kesimpulan dan

rekomendasi berdasarkan perbandingan ketiga skenario tersebut.



Gambar 2. Alur penelitian

4. HASIL DAN PEMBAHASAN

4.1. Hasil Pengujian

Pengujian dilaksanakan sebanyak lima iterasi pada masing-masing skenario dengan total 15 baris data hasil eksperimen. Data sensor yang dihasilkan oleh *virtual publisher* mencakup nilai suhu dalam rentang 25,80°C hingga 34,77°C dan kelembaban dalam rentang 55,47% hingga 79,71%, yang merepresentasikan kondisi pembacaan sensor DHT11 secara realistis. Ukuran payload plaintext yang dihasilkan berkisar antara 176 hingga 177 bytes, bergantung pada panjang karakter nilai suhu dan kelembaban yang dibangkitkan secara acak. Ringkasan statistik rata-rata hasil pengujian seluruh skenario disajikan pada Tabel I dan II.

Tabel 1. Rata-rata waktu pemrosesan per skenario

Skenario	Rata-rata waktu enkripsi (ms)	Waktu Publikasi Rata-rata (ms)	Rata-rata Total Waktu (ms)
Tanpa Enkripsi	0,0000	0,0213	0,0638
AES 256	0,0287	0,0872	0,2289
AES+TLS	0,0091	0,0747	0,1896

Tabel 2. Rata-rata ukuran data per skenario

Skenario	Ukuran Rata-rata Overhead (%)	Plain (bytes)	Cipher (bytes)
Tanpa Enkripsi	0,00	176,8	176,8
AES 256	8,59	176,8	192,0
AES+TLS	8,59	176,8	192,0

4.2. Analisis Waktu Enkripsi

Berdasarkan Tabel I, skenario *no_enc* mencatat waktu enkripsi sebesar 0,0000 ms karena tidak melibatkan operasi kriptografi apapun. Skenario *aes_only* mencatat rata-rata waktu enkripsi tertinggi

sebesar 0,0287 ms, dengan variasi antar iterasi yang cukup besar mulai dari 0,0061 ms hingga 0,0959 ms. Variasi ini disebabkan oleh karakteristik *cold start* pada eksekusi pertama, di mana pustaka *pycryptodome* melakukan inisialisasi konteks kriptografi sehingga iterasi pertama cenderung membutuhkan waktu lebih lama dibandingkan iterasi berikutnya. Skenario *aes_tls* mencatat rata-rata waktu enkripsi sebesar 0,0091 ms, lebih rendah dibandingkan *aes_only* karena pada saat pengujian *aes_tls* dijalankan, konteks kriptografi AES telah ter-cache oleh *runtime* Python sehingga eksekusi berlangsung lebih efisien. Secara keseluruhan, waktu enkripsi AES-256 CBC pada kedua skenario terenkripsi berada di bawah 0,1 ms, yang menunjukkan bahwa overhead komputasi algoritma ini sangat rendah dan layak diterapkan pada sistem IoT berbasis perangkat lunak.

4.3. Analisis Latensi Publikasi

Rata-rata latensi publikasi ke broker MQTT pada skenario *no_enc* adalah 0,0213 ms, sedangkan pada skenario *aes_only* meningkat menjadi 0,0872 ms dan pada skenario *aes_tls* sebesar 0,0747 ms. Peningkatan latensi publikasi pada kedua skenario terenkripsi terjadi karena ukuran payload yang dikirimkan lebih besar akibat proses enkripsi, sehingga membutuhkan waktu routing yang lebih lama pada simulator broker MQTT. Perbedaan latensi antara *aes_only* dan *aes_tls* tidak signifikan secara praktis, dengan selisih hanya sebesar 0,0125 ms. Hal ini menunjukkan bahwa penambahan lapisan TLS tidak memberikan beban latensi yang berarti pada tahap pengiriman pesan dalam lingkungan simulasi.

4.4. Analisis Total Waktu Pemrosesan

Total waktu pemrosesan yang mencakup keseluruhan operasi dari enkripsi hingga penerimaan pesan oleh subscriber menunjukkan perbedaan yang signifikan antar skenario. Skenario *no_enc* mencatat rata-rata total waktu paling singkat sebesar 0,0638 ms. Skenario *aes_tls* mencatat total waktu rata-rata sebesar 0,1896 ms, lebih singkat dibandingkan skenario *aes_only* yang mencatat 0,2289 ms. Perbandingan ini menunjukkan bahwa total overhead waktu akibat penerapan AES-256 CBC dibandingkan tanpa enkripsi adalah sekitar 0,1258 ms pada skenario *aes_tls*, atau setara dengan peningkatan sebesar 197,2% terhadap baseline *no_enc*. Meskipun demikian, nilai absolut overhead tersebut tetap berada dalam orde sub-milidetik, sehingga tidak berdampak signifikan terhadap performa sistem IoT secara keseluruhan.

4.5. Analisis Overhead Ukuran Data

Penerapan enkripsi AES-256 CBC dengan mekanisme padding PKCS#7 menghasilkan peningkatan ukuran payload secara konsisten pada kedua skenario terenkripsi. Ukuran ciphertext yang dihasilkan adalah 192 bytes untuk seluruh pesan, terlepas dari variasi kecil ukuran plaintext antara 176 dan 177 bytes. Hal ini disebabkan oleh mekanisme

padding PKCS#7 yang menambahkan byte tambahan hingga ukuran pesan menjadi kelipatan 128-bit (16 bytes) sesuai dengan ukuran blok AES. Rata-rata overhead ukuran data pada kedua skenario terenkripsi adalah 8,59%, dengan nilai terendah 8,47% dan tertinggi 9,09%. Overhead sebesar ini dinilai wajar dan dapat diterima mengingat tingkat keamanan yang diberikan oleh enkripsi AES-256, terlebih pada sistem IoT yang tidak memiliki batasan ketat terhadap ukuran bandwidth.

4.6. Analisis Keamanan Data

Demonstrasi pengujian sniffing membuktikan perbedaan tingkat keamanan yang nyata antar ketiga skenario. Pada skenario *no_enc*, penyerang yang berlangganan pada topik yang sama dapat membaca seluruh isi pesan secara langsung, termasuk nilai suhu (34,77°C), kelembaban (66,59%), *device_id*, *payload_id*, dan *timestamp* tanpa hambatan apapun. Pada skenario *aes_only*, penyerang hanya berhasil menangkap ciphertext dalam format yang tidak bermakna, seperti `yd1qxPFgiFgxfcu2EEhl8AbvIw3yqr7+KQmbpMuFgIq...`, sehingga isi data sensor tidak dapat diketahui. Namun demikian, metadata koneksi MQTT seperti nama topik sensor/temperature dan alamat broker masih dapat terekspos karena tidak adanya perlindungan pada lapisan transport. Pada skenario *aes_tls*, perlindungan berlapis berhasil diterapkan secara penuh: payload terenkripsi AES-256 dan seluruh kanal komunikasi dilindungi oleh TLS 1.3 dengan cipher suite TLS_AES_256_GCM_SHA384 serta pertukaran kunci sesi berbasis ECDH, sehingga penyerang tidak dapat mengakses baik isi pesan maupun metadata komunikasi. Subscriber yang memiliki kunci AES yang sah berhasil mendekripsi seluruh pesan pada kedua skenario terenkripsi tanpa kesalahan.

4.7. Pembahasan Umum

Hasil pengujian secara keseluruhan menunjukkan bahwa penerapan algoritma AES-256 CBC pada protokol MQTT memberikan peningkatan keamanan yang signifikan dengan overhead kinerja yang sangat minimal. Waktu enkripsi rata-rata di bawah 0,03 ms dan overhead ukuran data sebesar 8,59% merupakan tradeoff yang sangat menguntungkan mengingat tingkat perlindungan kriptografis yang diberikan. Skenario *aes_tls* terbukti memberikan perlindungan paling komprehensif dengan mengamankan baik lapisan payload maupun lapisan transport secara bersamaan, dan justru mencatat total waktu pemrosesan yang lebih rendah dibandingkan *aes_only* berkat efisiensi eksekusi *runtime* Python. Temuan ini mengonfirmasi bahwa implementasi AES-256 CBC dikombinasikan dengan TLS 1.3 merupakan solusi keamanan yang feasible dan efisien untuk diterapkan pada sistem komunikasi IoT berbasis protokol MQTT.

5. HASIL DAN PEMBAHASAN

Penelitian ini berhasil mengimplementasikan dan mengevaluasi mekanisme keamanan protokol MQTT menggunakan algoritma AES-256 *Cipher Block Chaining* (CBC) berbasis simulasi perangkat lunak pada platform Google Colaboratory dalam tiga skenario pengujian, yaitu *no_enc*, *aes_only*, dan *aes_tls*. Hasil pengujian menunjukkan bahwa penerapan enkripsi AES-256 CBC terbukti efektif melindungi kerahasiaan data sensor IoT dengan overhead kinerja yang sangat rendah, di mana rata-rata waktu enkripsi hanya sebesar 0,0287 ms pada skenario *aes_only* dan 0,0091 ms pada skenario *aes_tls*, serta overhead ukuran data yang konsisten sebesar 8,59% akibat mekanisme padding PKCS#7 dengan ukuran ciphertext 192 bytes dari plaintext rata-rata 176,8 bytes. Kombinasi AES-256 CBC dan TLS 1.3 pada skenario *aes_tls* terbukti memberikan perlindungan paling komprehensif dengan mengamankan lapisan payload sekaligus lapisan transport menggunakan cipher suite TLS_AES_256_GCM_SHA384, sehingga baik isi pesan maupun metadata koneksi MQTT sepenuhnya terlindungi dari upaya penyadapan, berbeda dengan skenario *aes_only* yang masih menyisakan kerentanan pada lapisan transport berupa eksposur metadata topik dan alamat broker. Untuk penelitian selanjutnya, disarankan agar pengujian dilaksanakan pada perangkat keras fisik ESP32 guna memperoleh data kinerja yang lebih representatif, serta memperluas perbandingan dengan mode operasi AES-GCM yang menyediakan enkripsi dan autentikasi pesan secara bersamaan, mengintegrasikan autentikasi *mutual* berbasis sertifikat digital X.509, serta meningkatkan jumlah iterasi pengujian untuk memperoleh hasil statistik yang lebih kokoh.

DAFTAR PUSTAKA

- [1] I. Suharti, A. Kusyanti, and F. A. Bakhtiar, "Implementasi Algoritme Enkripsi Grain-128 pada Sistem Pemantauan Elektrokardiogram berbasis Protokol MQTT-SN," *J. Pengemb. Teknol. ...*, vol. 6, no. 6, pp. 2770–2780, 2022, [Online]. Available: <https://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/11098%0Ahttps://j-ptiik.ub.ac.id/index.php/j-ptiik/article/download/11098/4950>
- [2] R. Akbar, S. Bahri, and I. Nirmala, "Implementasi Algoritma Rsa Untuk Proses Enkripsi-Autentikasi Publish-Subscribe Pada Protokol Mqtt Menggunakan Esp8266 Berbasis Iot," *Coding J. Komput. dan Apl.*, vol. 12, no. 1, p. 23, 2024, doi: 10.26418/coding.v12i1.70151.
- [3] H. Azwar and F. Gary, "Implementasi Transport Layer Security dengan Algoritma AES Pada Jaringan IoT Menggunakan Protokol MQTT," *J. Elektro dan Mesin Terap.*, vol. 11, no. 1, pp. 60–66, 2025, doi: 10.35143/elementer.v11i1.6478.
- [4] M. Liga and I. Khaliq, "Optimalisasi Keamanan Data dengan Protokol SSL di Lingkungan Cloud Computing," *J. Ilm. FLASH*, vol. 10, no. 1, pp. 1–6, 2024.
- [5] A. Syam Aswandi, M. Nurtanzis Sutoyo, and A. Pradipta, "Analisis Performa Dan Keamanan Implementasi Kriptografi Aes Untuk Penyandian Dokumen Berbasis Web," *J. Mnemon.*, vol. 8, no. 1, pp. 24–32, 2025, doi: 10.36040/mnemonic.v8i1.12053.
- [6] B. E. Widodo and A. S. Purnomo, "Implementasi Advanced Encryption Standard Pada Enkripsi Dan the Implementation of Advanced Encryption Standard on the Encryption and Decryption of the Confidential Documents At," *J. Tek. Inform.*, vol. 1, no. 2, pp. 69–77, 2020, [Online]. Available: <https://doi.org/10.20884/1.jutif.2020.1.2.21>
- [7] P. Alifia Rizky and S. Soim, "RESISTOR Journal | 71 Implementasi Algoritma Kriptografi AES CBC Untuk Keamanan Komunikasi Data Pada Hardware," *Resistor*, vol. 3, no. 1, pp. 71–78, 2024, [Online]. Available: <https://s.id/jurnalresistor>
- [8] N. Dharmawan, Gani Indriyanta, and I Kadek Dendy Senaparth, "Analisis Keamanan Jaringan Universitas Kristen Duta Wacana Dengan Serangan Ssl/Tls," *J. Terap. Teknol. Inf.*, vol. 6, no. 2, pp. 121–130, 2022, doi: 10.21460/jutei.2022.62.214.
- [9] M. E. Abdelhafez, S. Ramadass, and M. Abdelwahab, "TLS Guard for TLS 1.3 zero round-trip time (0-RTT) in a distributed environment," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 35, no. 10, p. 101797, 2023, doi: 10.1016/j.jksuci.2023.101797.
- [10] R. Rahman, "Analisis Penerapan Ssl / Tls Dalam Menjaga Keamanan Transmisi Data Pada Aplikasi Web," vol. 10, no. 1, pp. 171–173, 2026.
- [11] D. L. Sadewo, M. Ramdhani, and D. Rahmawati, "Analisis Komunikasi data menggunakan Internet of Things dengan protokol MQTT pada alat Swimming Lap Counter," *e-Proceeding Eng.*, vol. 11, no. 5, pp. 5654–5658, 2024.
- [12] S. Mahfuza, U. Ristian, and D. M. Midyanti, "Penerapan Protokol MQTT pada Sistem Otomatisasi Fotosintesis Tanaman Aquascape Berbasis Internet of Things," *Coding J. Komput. dan Apl.*, vol. 12, no. 3, pp. 264–274, 2024, doi: 10.26418/coding.v12i3.88183.