Peningkatan Keamanan Protokol Message Queuing Telemetry Transport Menggunakan Transport Layer Security dan Hash-Based Message Authentication Code

Enhancing the Security of the Message Queuing Telemetry Transport Protocol Using Transport Layer Security and Hash-Based Message Authentication Code

Huda Daulan[1*], Firmansyah Maulana Sugiarta Nursuwars [2], Rohmat Gunawan[3]
Program Studi Informatika Universitas Siliwangi
Jl. Mugarsari, Kelurahan Mugarsari, Kecamatan Tamansari, Kota Tasikmalaya, Jawa Barat
[1]207006048@student.unsil.ac.id
[2]firmansyah@unsil.ac.id
[3]rohmatgunawan@unsil.ac.id

[*]Corresponding Author*:* 207006048@student.unsil.ac.id[1]

Riwayat Artikel
Naskah diterima: 31/07/2026 | Naskah direvisi: 19/10/2026 | Naskah diterbitkan: 31/10/2026

Creative Commons License 4.0 (CC BY –NC)

Abstract*—However, MQTT communication without adequate security mechanisms remains vulnerable to eavesdropping and message manipulation. This study evaluates the implementation of Transport Layer Security (TLS) and Hash-Based Message Authentication Code using SHA-256 (HMAC-SHA-256), both separately and in combination, in an MQTT-based IoT system. The experiments involve four scenarios: MQTT without additional security, MQTT with TLS, MQTT with HMAC-SHA-256, and MQTT with both mechanisms. Security is evaluated in terms of confidentiality, message integrity, and authentication, while performance is assessed using delay, throughput, jitter, packet loss, memory usage, and CPU usage. The results show that the combined TLS and HMAC-SHA-256 scenario achieves the lowest delay of 3.088 ms and the highest throughput of 406.15 Byte/s. However, this scenario also records a packet loss of 1.35%, memory usage of 144,204.1 bytes, and CPU usage of 0.21%. TLS produces the lowest jitter at 3.420 ms, whereas HMAC-SHA-256 records the lowest memory usage at 99,762.4 bytes. These findings indicate that the security mechanisms provide complementary protection but exhibit different performance characteristics. The selection of a security configuration should therefore consider application security requirements and device resource constraints.

Keywords*—MQTT, IoT, TLS, HMAC-SHA-256, security, communication.

Intisari—Message Queuing Telemetry Transport (MQTT) banyak digunakan pada aplikasi Internet of Things (IoT) karena memiliki mekanisme komunikasi yang ringan. Namun, komunikasi MQTT tanpa mekanisme keamanan yang memadai rentan terhadap penyadapan dan manipulasi pesan. Penelitian ini mengevaluasi penerapan Transport Layer Security (TLS) dan Hash-Based Message Authentication Code berbasis SHA-256 (HMAC-SHA-256), baik secara terpisah maupun kombinasi, pada sistem IoT berbasis MQTT. Pengujian dilakukan melalui empat skenario, yaitu MQTT tanpa mekanisme keamanan tambahan, MQTT dengan TLS, MQTT dengan HMAC-SHA-256, serta MQTT dengan kombinasi kedua mekanisme tersebut. Evaluasi keamanan mencakup kerahasiaan, integritas pesan, dan autentikasi, sedangkan evaluasi performa meliputi delay, throughput, jitter, packet loss, penggunaan memori, dan penggunaan CPU. Hasil pengujian menunjukkan bahwa skenario kombinasi TLS dan HMAC-SHA-256 menghasilkan delay terendah sebesar 3,088 ms dan throughput tertinggi sebesar 406,15 Byte/s. Namun, skenario tersebut juga mencatat packet loss sebesar 1,35%, penggunaan memori sebesar 144.204,1 Byte, dan penggunaan CPU sebesar 0,21%. TLS menghasilkan jitter terendah sebesar 3,420 ms, sedangkan HMAC-SHA-256 menghasilkan penggunaan memori terendah sebesar 99.762,4 Byte. Temuan ini menunjukkan bahwa kedua mekanisme memberikan perlindungan yang saling melengkapi, tetapi memiliki karakteristik performa yang berbeda. Oleh karena itu, pemilihan konfigurasi keamanan perlu mempertimbangkan kebutuhan perlindungan aplikasi dan keterbatasan sumber daya perangkat.

***Kata Kunci*—MQTT, IoT, TLS, HMAC-SHA-256, keamanan, komunikasi.**

I. PENDAHULUAN

Internet of Things (IoT) telah berkembang menjadi salah satu teknologi penting yang mendukung berbagai bidang aplikasi, termasuk *smart home*, *smart healthcare*, *smart agriculture*, dan *smart industry* [1]. Implementasi IoT melibatkan pertukaran data secara terus-menerus antar perangkat melalui jaringan komunikasi, sehingga diperlukan protokol yang ringan, efisien, dan mampu beroperasi pada perangkat dengan sumber daya komputasi terbatas [2]. Salah satu protokol komunikasi yang banyak digunakan dalam lingkungan IoT adalah *Message Queuing Telemetry Transport* (MQTT). Protokol ini menerapkan pola komunikasi *publish-subscribe* yang memisahkan pengirim dan penerima melalui broker, memiliki overhead protokol yang relatif rendah, serta dirancang untuk mendukung komunikasi pada jaringan dengan bandwidth terbatas atau kondisi konektivitas yang tidak stabil [3]. Karakteristik tersebut menjadikan MQTT sesuai untuk berbagai aplikasi IoT yang membutuhkan pertukaran data secara efisien.

Meskipun menawarkan efisiensi komunikasi, penggunaan MQTT tetap menghadapi tantangan keamanan, khususnya ketika komunikasi berlangsung tanpa perlindungan kriptografi yang memadai. MQTT tidak secara mandiri menyediakan enkripsi payload pada lapisan protokol aplikasinya, sedangkan autentikasi menggunakan *username* dan *password* saja tidak cukup untuk menjamin kerahasiaan dan integritas data selama transmisi. Dalam konfigurasi tanpa perlindungan transport yang memadai, komunikasi MQTT berpotensi mengalami penyadapan, manipulasi pesan, pencurian kredensial, serta serangan *Man-in-the-Middle* (MITM) [4][5]. Risiko tersebut menjadi semakin penting dalam lingkungan IoT karena perangkat sering ditempatkan pada lokasi yang tidak sepenuhnya terkontrol dan memiliki keterbatasan sumber daya komputasi, memori, serta energi. Oleh karena itu, mekanisme keamanan yang diterapkan perlu mempertimbangkan keseimbangan antara tingkat perlindungan dan efisiensi operasional perangkat.

Salah satu pendekatan untuk meningkatkan keamanan komunikasi MQTT adalah penerapan *Transport Layer Security* (TLS). TLS menyediakan perlindungan komunikasi melalui enkripsi, mekanisme autentikasi berdasarkan sertifikat atau metode lain yang didukung, serta perlindungan integritas data selama transmisi [6]. Namun, penerapan TLS dapat menimbulkan overhead tambahan, antara lain waktu pembentukan koneksi, kebutuhan komputasi untuk operasi kriptografi, penggunaan memori, dan konsumsi energi. Besarnya dampak tersebut bergantung pada kemampuan perangkat, konfigurasi TLS, implementasi protokol, serta kondisi jaringan [7] [8]. Penelitian [9] menunjukkan bahwa penerapan TLS pada perangkat IoT berdaya rendah tetap memungkinkan, termasuk melalui pemanfaatan akselerasi perangkat keras. Meskipun demikian, evaluasi overhead dan performa tetap diperlukan untuk menentukan kesesuaian penerapan TLS pada perangkat dengan sumber daya terbatas.

Selain kerahasiaan komunikasi, integritas dan autentikasi pesan merupakan aspek penting dalam melindungi pertukaran data pada sistem IoT. Salah satu mekanisme yang dapat digunakan untuk memenuhi kebutuhan tersebut adalah *Hash-Based Message Authentication Code* (HMAC). HMAC mengombinasikan fungsi hash kriptografi dengan kunci rahasia bersama untuk menghasilkan kode autentikasi yang dapat diverifikasi oleh penerima. Dengan membandingkan kode yang diterima dengan hasil perhitungan ulang, penerima dapat mendeteksi perubahan pada pesan dan memverifikasi bahwa pesan dibuat oleh pihak yang mengetahui kunci rahasia tersebut. Landasan keamanan HMAC dibahas dalam penelitian mengenai konstruksi dan analisis fungsi autentikasi berbasis kunci [10], sedangkan standar kriptografi terkait menetapkan penggunaan HMAC dengan fungsi hash yang sesuai, termasuk SHA-256 [11]. HMAC-SHA-256 berpotensi menjadi mekanisme pelengkap untuk menjaga integritas dan keaslian pesan dengan biaya komputasi yang perlu diukur sesuai karakteristik perangkat dan beban komunikasi [12]. HMAC tidak mengenkripsi payload dan tidak dengan sendirinya mencegah penyadapan maupun serangan *replay*.

Penelitian terdahulu telah membahas keamanan MQTT melalui pendekatan yang beragam. Penelitian [7] dan [8] mengevaluasi dampak penerapan TLS terhadap keamanan dan performa komunikasi, sedangkan penelitian [13] mengembangkan mekanisme autentikasi berbasis HMAC melalui skema MARAS. Studi lainnya mengkaji pemanfaatan HMAC untuk mendukung autentikasi dan integritas data dalam berbagai domain aplikasi [14], [15]. Meskipun penelitian tersebut memberikan landasan penting bagi pengembangan komunikasi IoT yang aman, evaluasi terhadap penerapan TLS dan HMAC-SHA-256 secara terintegrasi pada komunikasi MQTT masih perlu diperdalam, khususnya melalui perbandingan eksperimental yang mengukur efektivitas perlindungan keamanan sekaligus overhead performa pada perangkat IoT. Dengan demikian, diperlukan evaluasi komparatif yang dapat mengidentifikasi kontribusi masing-masing mekanisme keamanan serta dampak ketika keduanya diterapkan secara bersamaan.

Berdasarkan kesenjangan tersebut, penelitian ini bertujuan untuk menganalisis pengaruh penerapan TLS dan HMAC-SHA-256, baik secara terpisah maupun kombinasi, terhadap keamanan dan performa komunikasi MQTT pada sistem IoT. Evaluasi dilakukan melalui empat skenario, yaitu MQTT tanpa mekanisme keamanan tambahan, MQTT dengan TLS, MQTT dengan HMAC-SHA-256, serta MQTT dengan kombinasi TLS dan HMAC-SHA-256. Pengujian keamanan difokuskan pada kemampuan mekanisme yang diterapkan dalam melindungi kerahasiaan komunikasi, mendeteksi manipulasi pesan, dan menghadapi skenario serangan MITM. Sementara itu, evaluasi performa mencakup parameter *latency*, *throughput*, *jitter*, dan *packet loss*, serta penggunaan CPU dan memori pada perangkat yang diuji. Hasil pengujian antarskenario dibandingkan untuk mengidentifikasi kontribusi masing-masing mekanisme keamanan dan kompromi antara perlindungan

JUTEI Edisi Volume. 10 No.2 Oktober 2026
ISSN 2579-3675, e-ISSN-5538
DOI 10.21460/jutei.2026.102.526

keamanan dengan overhead sistem. Penelitian ini diharapkan dapat memberikan dasar empiris untuk menentukan konfigurasi keamanan MQTT yang sesuai bagi implementasi IoT dengan sumber daya terbatas.

II. LANDASAN TEORI

A. MQTT

Message Queuing Telemetry Transport (MQTT) adalah protokol komunikasi pesan pada lapisan aplikasi yang dirancang untuk pertukaran data pada lingkungan Internet of Things (IoT), menggunakan mekanisme publish-subscribe melalui perantara yang disebut broker [16]. MQTT memiliki karakteristik ringan (lightweight), kompleksitas protokol yang relatif rendah, dan mendukung komunikasi asinkron antara perangkat pengirim dan penerima. Karakteristik ini membuatnya sesuai untuk berbagai aplikasi IoT, khususnya pemantauan sensor dan pengiriman data telemetri [17].

Mekanisme publish-subscribe pada MQTT memungkinkan perangkat mengirimkan pesan ke topik tertentu tanpa harus berkomunikasi secara langsung dengan perangkat penerima. Broker bertugas mengelola langganan (subscription) dan meneruskan pesan kepada klien yang berlangganan pada topik terkait. Pemisahan antara pengirim (publisher) dan penerima (subscriber) ini mendukung komunikasi yang fleksibel dan skalabel dalam sistem IoT [18].

Selain itu, MQTT mendukung tiga tingkat Quality of Service (QoS), yaitu QoS 0 (at most once), QoS 1 (at least once), dan QoS 2 (exactly once), yang memberikan pilihan tingkat jaminan pengiriman pesan sesuai kebutuhan aplikasi. MQTT umumnya beroperasi di atas TCP, sedangkan perlindungan komunikasi dapat diterapkan menggunakan TLS dan mekanisme autentikasi. Namun, penggunaan MQTT saja tidak otomatis menjamin keamanan komunikasi; konfigurasi keamanan tetap diperlukan untuk melindungi data dari penyadapan, manipulasi, dan akses yang tidak sah [19].

B. TLS

Transport Layer Security (TLS) merupakan protokol keamanan berbasis kriptografi yang dirancang untuk melindungi pertukaran informasi antara dua pihak yang berkomunikasi melalui jaringan komputer. Protokol ini berperan dalam membentuk kanal komunikasi yang aman dengan menyediakan mekanisme perlindungan terhadap kerahasiaan (*confidentiality*), integritas (*integrity*), dan autentikasi (*authentication*) data. Kerahasiaan memastikan bahwa isi pesan yang dikirimkan tidak dapat dibaca oleh pihak yang tidak memiliki otorisasi, sedangkan integritas memungkinkan penerima mendeteksi perubahan data yang terjadi selama proses transmisi. Sementara itu, autentikasi digunakan untuk memverifikasi identitas pihak yang berkomunikasi sesuai dengan mekanisme autentikasi yang diterapkan. TLS telah menjadi salah satu protokol kriptografi yang banyak digunakan dalam komunikasi internet, termasuk untuk melindungi pertukaran data pada layanan web dan aplikasi jaringan.

Dalam TLS 1.3, pembentukan kanal komunikasi yang aman melibatkan dua komponen utama, yaitu *handshake protocol* dan *record layer protocol*. Proses *handshake* memungkinkan client dan server menyepakati parameter kriptografi, melakukan autentikasi sesuai konfigurasi, serta membentuk rahasia bersama yang digunakan untuk menurunkan kunci sesi. Pada mekanisme autentikasi berbasis sertifikat, identitas server dapat diverifikasi melalui sertifikat digital dan bukti kriptografis yang terkait. Setelah parameter dan kunci sesi terbentuk, *record layer* digunakan untuk melindungi data aplikasi yang dipertukarkan dengan mekanisme enkripsi terautentikasi, sehingga kerahasiaan dan integritas pesan dapat dipertahankan selama komunikasi berlangsung. Analisis keamanan TLS 1.3 menunjukkan bahwa keamanan mekanisme tersebut bergantung pada pemenuhan asumsi kriptografi dan penggunaan protokol sesuai rancangan yang ditetapkan [20]. Oleh karena itu, penerapan TLS perlu disertai konfigurasi yang tepat, termasuk pemilihan versi protokol, algoritma kriptografi, serta validasi sertifikat, agar perlindungan komunikasi dapat berfungsi sebagaimana mestinya.

C. HMAC

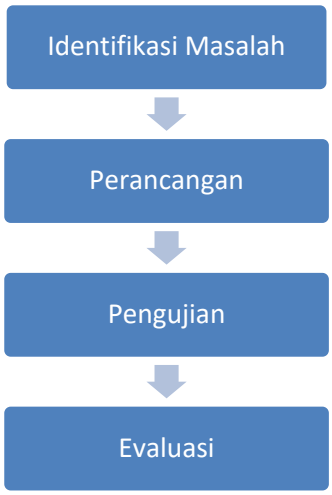
Hash-Based Message Authentication Code (HMAC) merupakan mekanisme autentikasi pesan berbasis kriptografi yang mengombinasikan fungsi hash kriptografi dengan kunci rahasia (secret key) untuk memverifikasi integritas data dan keaslian pesan. Berbeda dengan fungsi hash biasa yang dapat dihitung oleh siapa pun tanpa memerlukan kunci rahasia, HMAC menggunakan kunci yang hanya diketahui oleh pihak-pihak yang berwenang dalam proses komunikasi [13]. Pengirim menghitung nilai autentikasi atau message authentication code berdasarkan isi pesan dan kunci rahasia tersebut, kemudian mengirimkan pesan beserta nilai HMAC kepada penerima. Setelah menerima pesan, penerima menghitung kembali nilai HMAC menggunakan pesan yang diterima dan kunci rahasia yang sama, lalu membandingkan hasilnya dengan nilai autentikasi yang dikirimkan. Apabila kedua nilai tersebut cocok, pesan dapat dinyatakan lolos verifikasi integritas dan autentikasi berdasarkan kunci yang digunakan. Sebaliknya, ketidakcocokan nilai HMAC menunjukkan bahwa pesan mungkin telah mengalami perubahan atau dibuat oleh pihak yang tidak memiliki kunci rahasia yang sesuai. Dengan demikian, HMAC dapat membantu mendeteksi manipulasi data selama transmisi sekaligus memberikan bukti kriptografis bahwa pesan berasal dari pihak yang memiliki kunci rahasia yang benar. Mekanisme ini telah digunakan dalam berbagai skema keamanan komunikasi dan autentikasi data, termasuk pada lingkungan jaringan dan Internet of Things (IoT) [21].

III. METODOLOGI

A. Tahapan Penelitian

Penelitian ini dilaksanakan melalui empat tahapan utama yang disusun secara sistematis untuk memastikan bahwa setiap proses penelitian dapat berjalan sesuai dengan tujuan yang telah ditetapkan. Tahapan tersebut meliputi identifikasi masalah, perancangan sistem dan skenario eksperimen,

pengujian, serta evaluasi hasil penelitian. Tahap identifikasi masalah dilakukan untuk menentukan fokus permasalahan dan aspek yang akan dikaji, sedangkan tahap perancangan mencakup penyusunan arsitektur sistem, pemilihan perangkat keras dan perangkat lunak, serta penetapan skenario pengujian. Selanjutnya, tahap pengujian dilaksanakan untuk memperoleh data empiris mengenai keamanan dan performa komunikasi MQTT dengan menerapkan beberapa konfigurasi mekanisme keamanan. Tahap terakhir adalah evaluasi, yang bertujuan untuk menganalisis dan membandingkan hasil pengujian guna mengetahui pengaruh penerapan TLS dan HMAC-SHA-256, baik secara terpisah maupun dalam kombinasi, terhadap keamanan komunikasi, performa jaringan, dan penggunaan sumber daya perangkat. Hubungan antara keempat tahapan tersebut beserta urutan pelaksanaannya disajikan secara sistematis pada Gambar 1.



Gambar 1. Tahapan Penelitian

B. *Identifikasi Masalah*

Tahap identifikasi masalah dilakukan untuk mengkaji kebutuhan penerapan mekanisme keamanan pada komunikasi *Message Queuing Telemetry Transport* (MQTT), khususnya dalam sistem Internet of Things (IoT) yang melibatkan pertukaran data antara perangkat sensor dan broker MQTT melalui jaringan. Meskipun *Transport Layer Security* (TLS) dan *Hash-Based Message Authentication Code* (HMAC) memiliki fungsi keamanan yang berbeda dan berpotensi saling melengkapi, pengaruh penerapan kedua mekanisme tersebut, baik secara individual maupun secara kombinasi, terhadap keamanan dan performa komunikasi MQTT perlu dianalisis melalui pengujian empiris. TLS berfungsi melindungi kanal komunikasi melalui mekanisme enkripsi dan perlindungan integritas data, sedangkan HMAC-SHA-256 digunakan untuk memverifikasi integritas dan keaslian pesan berdasarkan kunci rahasia bersama. Namun, penerapan mekanisme keamanan dapat menimbulkan tambahan beban pemrosesan dan penggunaan sumber daya perangkat yang berpotensi memengaruhi karakteristik komunikasi, terutama pada perangkat IoT dengan keterbatasan kapasitas komputasi dan memori. Oleh karena itu, diperlukan penelitian eksperimental untuk mengidentifikasi kontribusi masing-masing mekanisme keamanan serta pengaruh kombinasi keduanya terhadap perlindungan data dan performa komunikasi MQTT. Analisis dilakukan dengan membandingkan kondisi komunikasi tanpa mekanisme keamanan, dengan penerapan TLS saja, dengan penerapan HMAC-SHA-256 saja, serta dengan kombinasi TLS dan HMAC-SHA-256. Melalui perbandingan tersebut, penelitian ini diharapkan dapat memberikan gambaran yang terukur mengenai efektivitas penerapan mekanisme keamanan dalam melindungi komunikasi MQTT, sekaligus mengidentifikasi perubahan performa jaringan dan penggunaan sumber daya perangkat yang terjadi pada setiap skenario pengujian.

C. *Perancangan*

Tahap perancangan sistem dilakukan untuk menentukan arsitektur, perangkat keras, perangkat lunak, serta mekanisme keamanan yang digunakan dalam implementasi dan pengujian komunikasi *Message Queuing Telemetry Transport* (MQTT). Perancangan ini disusun untuk mendukung pelaksanaan eksperimen secara sistematis, sehingga pengaruh penerapan *Transport Layer Security* (TLS) dan *Hash-Based Message Authentication Code* (HMAC) terhadap keamanan dan performa komunikasi dapat diamati dan dibandingkan berdasarkan skenario pengujian yang telah ditentukan.

Perangkat keras yang digunakan dalam penelitian ini terdiri atas ESP32 sebagai mikrokontroler, sensor DHT22 sebagai perangkat untuk mengukur suhu dan kelembapan lingkungan, serta komputer yang menjalankan layanan broker MQTT dan berperan sebagai *subscriber*. ESP32 dirancang untuk membaca data dari sensor DHT22, kemudian mengirimkan data hasil pengukuran melalui jaringan Wi-Fi menggunakan mekanisme komunikasi MQTT. Komputer yang menjalankan broker Mosquitto berfungsi menerima pesan dari ESP32 dan meneruskannya kepada *subscriber* untuk diproses lebih lanjut. Konfigurasi tersebut memungkinkan proses pengiriman, penerimaan, dan pemantauan pesan dilakukan dalam satu lingkungan eksperimen yang terkontrol.

Dari sisi perangkat lunak, Arduino IDE digunakan untuk mengembangkan, mengompilasi, dan mengunggah program ke ESP32, termasuk program pembacaan sensor dan pengiriman data melalui MQTT. Mosquitto digunakan sebagai perangkat lunak broker MQTT yang mengelola pertukaran pesan antara *publisher* dan *subscriber*. Sementara itu, OpenSSL dimanfaatkan untuk mendukung pengelolaan sertifikat digital dan konfigurasi komunikasi berbasis TLS. Penerapan TLS dirancang untuk melindungi kanal komunikasi melalui mekanisme enkripsi serta perlindungan integritas data, sehingga informasi yang dikirimkan tidak dapat dibaca secara langsung oleh pihak yang tidak memiliki akses yang sah ketika transmisi berlangsung. Validasi sertifikat juga diperhatikan untuk membantu memastikan bahwa perangkat terhubung ke server yang identitasnya dapat diverifikasi.

Selain perlindungan pada lapisan transport, penelitian ini merancang mekanisme verifikasi integritas dan

JUTEI Edisi Volume. 10 No.2 Oktober 2026
ISSN 2579-3675, e-ISSN-5538
DOI 10.21460/jutei.2026.102.526

autentikasi pesan menggunakan HMAC-SHA-256. Implementasi mekanisme tersebut dilakukan dengan bahasa pemrograman Python melalui pustaka hmac dan hashlib. Pustaka hashlib digunakan untuk mendukung fungsi hash SHA-256, sedangkan pustaka hmac digunakan untuk menghasilkan dan memverifikasi kode autentikasi pesan berdasarkan kunci rahasia bersama. Dalam implementasinya, pengirim menghitung nilai HMAC dari pesan yang akan dikirim, sedangkan penerima menghitung kembali nilai autentikasi berdasarkan pesan yang diterima dan kunci yang sama. Hasil perhitungan tersebut kemudian diverifikasi untuk menentukan apakah pesan memenuhi persyaratan integritas dan autentikasi. Dengan demikian, mekanisme HMAC-SHA-256 dirancang untuk mendeteksi perubahan pesan dan membantu memverifikasi bahwa pesan dibuat oleh pihak yang memiliki kunci rahasia yang sesuai. Karena HMAC tidak menyediakan enkripsi, penerapan mekanisme ini secara terpisah dari TLS tidak memberikan perlindungan kerahasiaan terhadap isi pesan.

Untuk mendukung observasi dan pengumpulan data, MQTT Explorer digunakan sebagai perangkat bantu untuk memantau topik, pesan, dan aktivitas komunikasi MQTT. Wireshark digunakan untuk menangkap serta menganalisis paket jaringan guna mengamati karakteristik pertukaran data dan mendukung pengujian keamanan dalam lingkungan laboratorium yang terkontrol. Data hasil eksperimen, termasuk catatan waktu pengiriman dan penerimaan pesan, hasil verifikasi HMAC, hasil pengujian keamanan, serta parameter performa komunikasi, disimpan dalam basis data SQLite atau berkas CSV. Penyimpanan tersebut dirancang agar data dapat dikelompokkan berdasarkan skenario pengujian dan diolah lebih lanjut pada tahap evaluasi.

Secara keseluruhan, rancangan sistem memungkinkan dilakukannya pengujian terhadap empat konfigurasi komunikasi, yaitu MQTT tanpa mekanisme keamanan, MQTT dengan TLS, MQTT dengan HMAC-SHA-256, serta MQTT dengan kombinasi TLS dan HMAC-SHA-256. Seluruh konfigurasi dirancang menggunakan parameter eksperimen yang konsisten, seperti ukuran *payload*, interval pengiriman pesan, tingkat *Quality of Service* (QoS), dan jumlah pesan yang dikirimkan. Konsistensi parameter tersebut diperlukan agar perbedaan hasil pengujian dapat dianalisis berdasarkan mekanisme keamanan yang diterapkan, bukan semata-mata akibat perubahan kondisi eksperimen. Hasil dari tahap perancangan ini selanjutnya menjadi dasar bagi pelaksanaan pengujian keamanan, pengukuran performa komunikasi, dan evaluasi penggunaan sumber daya perangkat.

D. Pengujian

Pengujian dilakukan menggunakan empat skenario komunikasi MQTT sebagaimana disajikan pada Tabel I. Keempat skenario tersebut dirancang untuk mengevaluasi pengaruh penerapan TLS (*Transport Layer Security*) dan HMAC-SHA-256 (*Hash-Based Message Authentication Code*) terhadap keamanan dan performa komunikasi, baik ketika diterapkan secara terpisah maupun secara kombinasi. Pada setiap skenario, ESP32 yang terhubung dengan sensor DHT22 berfungsi sebagai *publisher* yang mengirimkan data suhu dan kelembapan melalui jaringan Wi-Fi ke broker MQTT Mosquitto. Broker kemudian meneruskan pesan kepada *subscriber* untuk diterima dan diproses lebih lanjut. Pengujian dilakukan dengan mempertahankan parameter eksperimen, seperti ukuran payload, interval pengiriman, tingkat QoS (*Quality of Service*), jumlah pesan, dan kondisi jaringan, agar hasil antar skenario dapat dibandingkan secara objektif.

Tabel I
SKENARIO PENGUJIAN

Skenario Pengujian	Mekanisme Keamanan	Tujuan Pengujian
Skenario 1	Tanpa keamanan	Menetapkan kondisi dasar (baseline) untuk evaluasi performa keamanan
Skenario 2	TLS	Mengevaluasi efektivitas TLS dalam menjaga kerahasiaan data selama transmisi serta menganalisis dampaknya terhadap performa komunikasi MQTT.
Skenario 3	HMAC-SHA256	Mengevaluasi kemampuan HMAC-SHA-256 dalam memverifikasi integritas dan keaslian pesan serta menganalisis dampaknya terhadap performa komunikasi MQTT.
Skenario 4	TLS dan HMAC-SHA256	Mengevaluasi efektivitas kombinasi TLS dan HMAC-SHA-256 dalam meningkatkan keamanan komunikasi serta menganalisis dampak gabungan keduanya terhadap performa sistem.

Selanjutnya, dilakukan pengujian keamanan melalui simulasi serangan *Man-in-the-Middle* (MITM) dalam lingkungan laboratorium yang terkontrol. Pengujian ini bertujuan untuk mengamati kemampuan masing-masing skenario dalam menghadapi penyadapan dan manipulasi pesan selama transmisi. Pada skenario TLS, pengujian difokuskan pada perlindungan kerahasiaan data dan validasi sertifikat. Pada skenario HMAC-SHA-256, pengujian difokuskan pada kemampuan mendeteksi perubahan pesan melalui proses verifikasi kode autentikasi. Sementara itu, pada skenario kombinasi TLS dan HMAC-SHA-256, pengujian dilakukan untuk mengetahui kontribusi kedua mekanisme dalam memberikan perlindungan berlapis terhadap komunikasi MQTT. Seluruh hasil pengujian dicatat dan digunakan sebagai dasar untuk mengevaluasi efektivitas mekanisme keamanan serta membandingkan performa komunikasi berdasarkan parameter yang telah ditetapkan pada tahap evaluasi.

E. Evaluasi

Evaluasi dilakukan terhadap dua aspek utama, yaitu keamanan dan performa komunikasi. Evaluasi keamanan bertujuan untuk menganalisis efektivitas penerapan TLS (*Transport Layer Security*) dan HMAC-SHA-256 dalam meningkatkan keamanan komunikasi MQTT, yang mencakup perlindungan terhadap serangan *Man-in-the-Middle* (MITM), kerahasiaan data, integritas pesan, serta autentikasi dan verifikasi keaslian pesan. Evaluasi performa

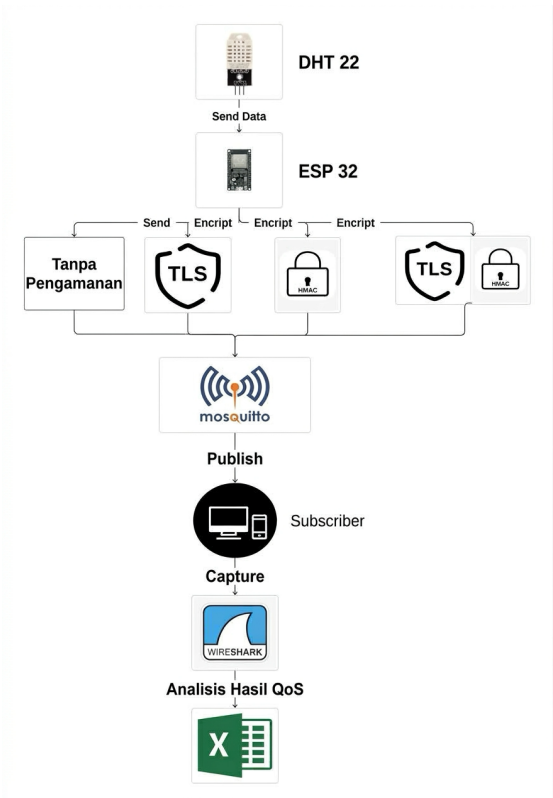
dilakukan berdasarkan parameter *Quality of Service* (QoS), yang meliputi latensi (*latency*), *throughput*, *jitter*, dan *packet loss*. Selain itu, penggunaan sumber daya perangkat, khususnya CPU dan memori pada ESP32, diukur untuk mengetahui overhead komputasi yang ditimbulkan oleh penerapan masing-masing mekanisme keamanan.

Data hasil pengujian selanjutnya dianalisis secara komparatif berdasarkan skenario eksperimen untuk mengidentifikasi pengaruh penerapan TLS dan HMAC-SHA-256, baik secara terpisah maupun secara kombinasi, terhadap keamanan, performa komunikasi, dan penggunaan sumber daya perangkat. Hasil analisis tersebut digunakan untuk menentukan konfigurasi yang memberikan keseimbangan terbaik antara tingkat keamanan dan efisiensi komunikasi pada sistem Internet of Things (IoT) berbasis MQTT.

IV. Hasil Dan Analisis

A. Implementasi Sistem

Sistem komunikasi MQTT diimplementasikan menggunakan ESP32 sebagai publisher yang terhubung dengan sensor DHT22 untuk memperoleh data suhu dan kelembapan. Data sensor dikirimkan melalui jaringan Wi-Fi ke broker Mosquitto, kemudian diteruskan kepada subscriber yang berjalan pada komputer seperti ditampilkan pada gambar 2.



Gambar 2. Implementasi Sistem

Pengujian dilakukan menggunakan empat skenario, yaitu komunikasi MQTT tanpa mekanisme keamanan tambahan, MQTT dengan TLS, MQTT dengan HMAC-SHA-256, serta MQTT dengan kombinasi TLS dan HMAC-SHA-256. Setiap skenario digunakan untuk mengevaluasi karakteristik keamanan dan performa komunikasi. Analisis keamanan dilakukan berdasarkan kemampuan masing-masing mekanisme dalam menjaga kerahasiaan data, integritas pesan, dan autentikasi. Sementara itu, evaluasi performa mencakup pengukuran delay, throughput, jitter, packet loss, penggunaan memori, dan penggunaan CPU. Data hasil pengujian selanjutnya dibandingkan untuk mengidentifikasi pengaruh penerapan mekanisme keamanan terhadap komunikasi MQTT pada perangkat IoT.

B. Analisis Keamanan

Pengujian keamanan dilakukan menggunakan skenario serangan *Man-in-the-Middle* (MITM) pada setiap mekanisme komunikasi. Hasil pengujian menunjukkan bahwa setiap mekanisme memberikan tingkat perlindungan yang berbeda sebagaimana ditunjukkan pada Tabel II.

Tabel II
Analisis Keamanan

Skenario	Kerahasiaan	Integritas	Autentikasi
Tanpa Keamanan	X	X	X
TLS	✓	✓	X
HMAC-SHA256	X	✓	✓
TLS + HMAC-SHA256	✓	✓	✓

Hasil evaluasi keamanan menunjukkan adanya perbedaan karakteristik perlindungan pada keempat skenario pengujian. Pada skenario tanpa mekanisme keamanan tambahan, data MQTT dikirimkan tanpa perlindungan enkripsi pada lapisan transport. Akibatnya, payload berpotensi dibaca oleh pihak yang dapat mengakses lalu lintas jaringan. Selain itu, tanpa mekanisme verifikasi integritas dan autentikasi pesan yang memadai, perubahan pesan selama transmisi tidak secara otomatis terdeteksi oleh aplikasi penerima.

Pada skenario TLS, komunikasi memperoleh perlindungan kerahasiaan dan integritas melalui mekanisme kriptografi pada lapisan transport. Perlindungan tersebut membantu mencegah pihak yang tidak berwenang membaca atau mengubah data selama transmisi. Namun, autentikasi identitas pengirim aplikasi bergantung pada konfigurasi autentikasi TLS yang digunakan. Apabila hanya autentikasi server yang diterapkan, TLS tidak dengan sendirinya membuktikan identitas setiap publisher.

Pada skenario HMAC-SHA-256, penerima memverifikasi kode autentikasi pesan menggunakan kunci rahasia bersama. Apabila pesan mengalami perubahan, hasil perhitungan HMAC tidak akan sesuai dengan kode autentikasi yang diterima sehingga manipulasi dapat terdeteksi. Meskipun demikian, HMAC-SHA-256 tidak mengenkripsi payload. Oleh karena itu, data masih dapat terbaca selama transmisi apabila tidak dilindungi oleh mekanisme enkripsi lain.

Skenario kombinasi TLS dan HMAC-SHA-256 mengintegrasikan perlindungan kerahasiaan komunikasi dari TLS dengan verifikasi integritas dan keaslian pesan melalui

JUTEI Edisi Volume. 10 No.2 Oktober 2026
ISSN 2579-3675, e-ISSN-5538
DOI 10.21460/jutei.2026.102.526

HMAC-SHA-256. Kombinasi ini memberikan perlindungan pada dua aspek yang saling melengkapi. Namun, efektivitasnya tetap bergantung pada konfigurasi TLS, pengelolaan kunci rahasia, serta implementasi verifikasi HMAC pada sisi penerima.

Secara keseluruhan, hasil tersebut memperlihatkan bahwa TLS dan HMAC-SHA-256 memiliki fungsi yang berbeda dan saling melengkapi. TLS sesuai untuk melindungi komunikasi selama transmisi, sedangkan HMAC-SHA-256 memungkinkan aplikasi memverifikasi integritas dan keaslian pesan. Meskipun demikian, klaim keberhasilan menghadapi serangan *Man-in-the-Middle* (MITM) perlu didasarkan pada bukti pengujian, seperti hasil validasi sertifikat, keberhasilan atau kegagalan koneksi, keterbacaan payload dalam hasil penangkapan paket, serta kemampuan sistem mendeteksi pesan yang telah dimodifikasi.

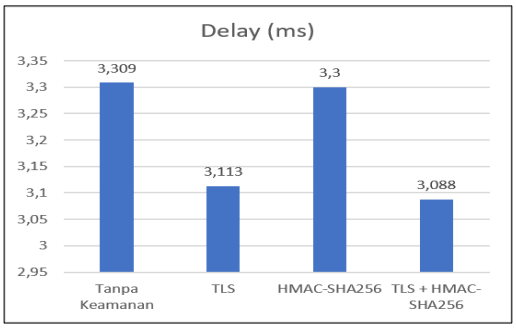
C. Analisis Kinerja Sistem

Evaluasi performa dilakukan dengan membandingkan nilai *delay*, *throughput*, *jitter*, *packet loss*, penggunaan memori, dan penggunaan CPU pada setiap skenario. Hasil pengukuran yang disajikan pada Tabel III menjadi dasar untuk menganalisis perubahan performa setelah penerapan TLS dan HMAC-SHA-256.

TABEL III
HASIL PENGUJIAN QOS DAN PENGGUNAAN SUMBER DAYA

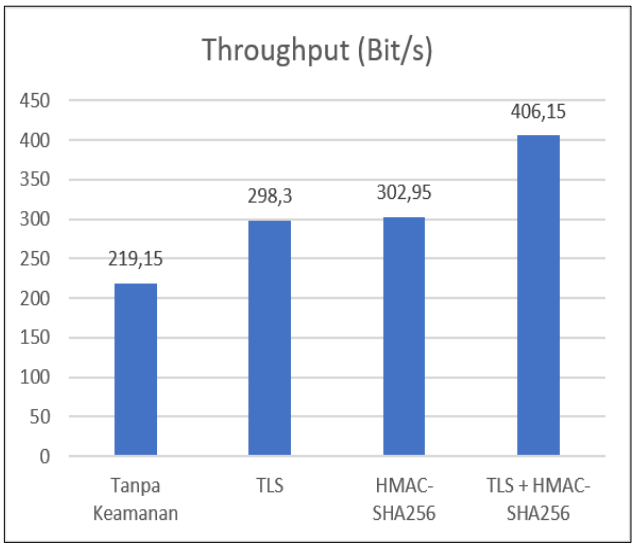
Skenario	Delay (ms)	Throughput (Bit/s)	Jitter (ms)	Packet Loss (%)	Penggunaan Memori (Byte)	Penggunaan CPU (%)
Tanpa Keamanan	3,309	219,15	3,491	0	100075,5	0,12%
TLS	3,113	298,3	3,42	0	148682,9	0,13%
HMAC-SHA256	3,3	302,95	3,43	0	99762,4	0,21%
TLS + HMAC-SHA256	3,088	406,15	3,444	1,35	144204,1	0,21%

1) Analisis Delay dan Throughput



Gambar 3. Hasil Pengukuran Delay

Berdasarkan gambar 3, hasil pengujian menunjukkan bahwa skenario tanpa mekanisme keamanan tambahan menghasilkan delay sebesar 3,309 ms. Pada skenario TLS, nilai tersebut turun menjadi 3,113 ms, sedangkan skenario HMAC-SHA-256 menghasilkan delay sebesar 3,300 ms. Nilai delay terendah diperoleh pada skenario kombinasi TLS dan HMAC-SHA-256, yaitu 3,088 ms. Dibandingkan dengan kondisi dasar, nilai delay pada skenario TLS turun sekitar 5,92%, sedangkan pada skenario HMAC-SHA-256 turun sekitar 0,27%. Pada skenario kombinasi, penurunan delay mencapai sekitar 6,68%. Hasil ini menunjukkan bahwa pada kondisi eksperimen yang dilakukan, penerapan mekanisme keamanan tidak disertai peningkatan delay berdasarkan nilai yang tercatat. Namun, hasil tersebut belum cukup untuk menyimpulkan bahwa mekanisme keamanan secara langsung mempercepat komunikasi, karena pengaruh variasi jaringan, jumlah pengulangan, dan ketidakpastian pengukuran perlu diperhitungkan.



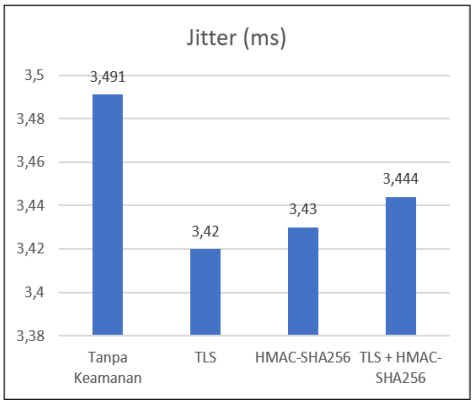
Gambar 4. Hasil Pengukuran Throughput

Berdasarkan gambar 4 hasil pengukuran throughput, skenario tanpa mekanisme keamanan menghasilkan 219,15 Byte/s. Nilai tersebut meningkat menjadi 298,30 Byte/s pada skenario TLS dan 302,95 Byte/s pada skenario HMAC-SHA-256. Skenario kombinasi TLS dan HMAC-SHA-256 menghasilkan throughput tertinggi, yaitu 406,15 Byte/s. Nilai ini sekitar 85,34% lebih tinggi daripada kondisi dasar. Peningkatan throughput tersebut menunjukkan bahwa skenario dengan mekanisme keamanan menghasilkan nilai laju transfer data yang lebih tinggi pada pengukuran yang dilakukan. Akan tetapi, karena mekanisme keamanan umumnya menambahkan pemrosesan kriptografi, peningkatan ini tidak dapat langsung diatribusikan kepada TLS atau HMAC. Pengujian berulang dengan kondisi jaringan yang terkontrol diperlukan untuk memastikan apakah perbedaan tersebut konsisten atau dipengaruhi oleh variasi pengukuran.

2) Analisis Jitter dan Packet Loss

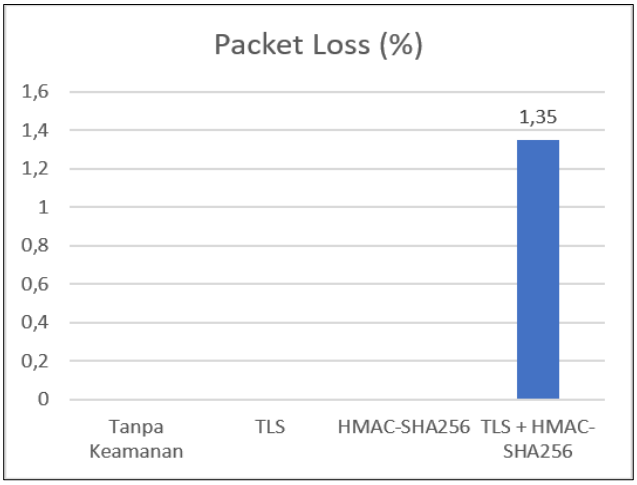
Hasil pengukuran jitter menunjukkan nilai sebesar 3,491 ms pada skenario tanpa mekanisme keamanan, 3,420 ms pada skenario TLS, 3,430 ms pada skenario HMAC-SHA-256, dan 3,444 ms pada skenario kombinasi. Dengan

demikian, nilai jitter terendah diperoleh pada skenario TLS, sedangkan nilai tertinggi terdapat pada skenario tanpa mekanisme keamanan seperti ditunjukkan pada gambar 5.



Gambar 5. Hasil Pengukuran Jitter

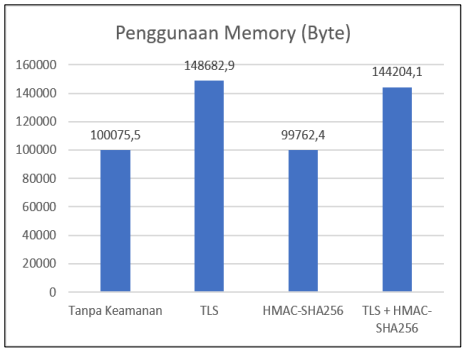
Perbedaan nilai jitter antarskenario relatif kecil. Dibandingkan dengan kondisi dasar, penurunan jitter pada skenario TLS sekitar 2,03%, pada skenario HMAC-SHA-256 sekitar 1,75%, dan pada skenario kombinasi sekitar 1,35%. Berdasarkan hasil tersebut, belum terlihat perbedaan besar dalam kestabilan waktu kedatangan paket antarskenario. Penilaian lebih lanjut memerlukan pengulangan eksperimen dan analisis variasi hasil pengukuran.



Gambar 6. Hasil Pengukuran Packet Loss

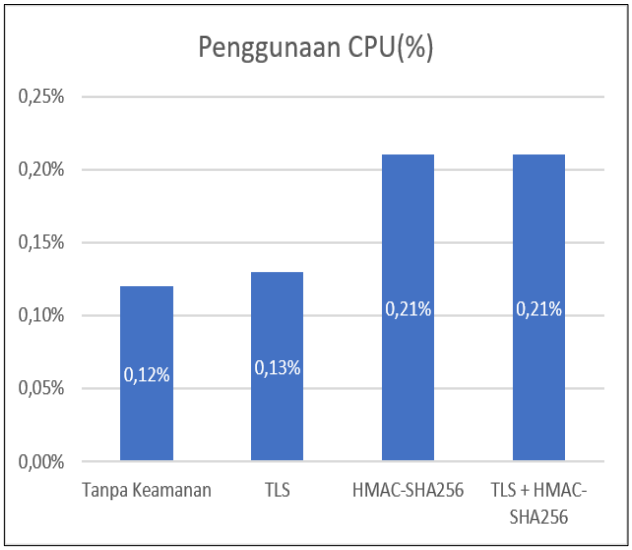
Berdasarkan gambar 6 hasil pengukuran packet loss, skenario tanpa mekanisme keamanan, TLS, dan HMAC-SHA-256 masing-masing menghasilkan nilai 0%. Sementara itu, skenario kombinasi TLS dan HMAC-SHA-256 menghasilkan packet loss sebesar 1,35%. Temuan ini menunjukkan bahwa kehilangan paket tercatat pada skenario kombinasi, tetapi belum membuktikan bahwa penerapan kedua mekanisme tersebut merupakan penyebab langsung kehilangan paket. Untuk menguji hubungan sebab-akibat, diperlukan pengulangan pengujian dengan kondisi jaringan yang setara serta pemeriksaan jumlah paket yang dikirim dan diterima.

3) Analisis Penggunaan Memori dan CPU



Gambar 7. Hasil Pengukuran Penggunaan Memory

Berdasarkan gambar 7 hasil pengukuran penggunaan memori menunjukkan nilai sebesar 100.075,5 Byte pada skenario tanpa mekanisme keamanan tambahan. Pada skenario TLS, penggunaan memori meningkat menjadi 148.682,9 Byte, sedangkan skenario HMAC-SHA-256 menghasilkan penggunaan memori sebesar 99.762,4 Byte. Skenario kombinasi TLS dan HMAC-SHA-256 membutuhkan memori sebesar 144.204,1 Byte. Dibandingkan dengan kondisi dasar, penggunaan memori pada skenario TLS meningkat sekitar 48,58%, sedangkan pada skenario HMAC-SHA-256 turun sekitar 0,31%. Pada skenario kombinasi, penggunaan memori meningkat sekitar 44,10%. Hasil ini menunjukkan bahwa TLS dan kombinasi TLS dengan HMAC-SHA-256 memerlukan lebih banyak memori pada implementasi yang diuji dibandingkan dengan kondisi dasar. Sebaliknya, penggunaan memori pada skenario HMAC-SHA-256 tercatat sedikit lebih rendah daripada kondisi dasar. Perbedaan tersebut perlu ditafsirkan sesuai metode pengukuran, kondisi awal memori, dan implementasi perangkat lunak yang digunakan.



Gambar 8. Hasil Pengukuran Penggunaan CPU

Berdasarkan gambar 8, hasil penggunaan CPU menunjukkan nilai sebesar 0,12% pada skenario tanpa mekanisme keamanan, 0,13% pada skenario TLS, serta 0,21% pada skenario HMAC-SHA-256 dan kombinasi TLS

JUTEI Edisi Volume. 10 No.2 Oktober 2026
ISSN 2579-3675, e-ISSN-5538
DOI 10.21460/jutei.2026.102.526

dengan HMAC-SHA-256. Dengan demikian, penggunaan CPU pada skenario TLS meningkat sekitar 8,33% relatif terhadap kondisi dasar, sedangkan kedua skenario yang menggunakan HMAC-SHA-256 meningkat sekitar 75%.

Hasil pengukuran tersebut memperlihatkan bahwa penggunaan HMAC-SHA-256 berasosiasi dengan peningkatan penggunaan CPU pada eksperimen ini. Akan tetapi, nilai yang tercatat masih merupakan hasil pengukuran pada konfigurasi dan beban kerja tertentu, sehingga belum dapat digeneralisasi untuk semua perangkat IoT. Pengukuran berulang dan dokumentasi metode pemantauan CPU diperlukan untuk memastikan bahwa perbedaan tersebut konsisten.

D. Pembahasan

Berdasarkan hasil pengujian, setiap skenario menunjukkan karakteristik yang berbeda dalam aspek keamanan dan penggunaan sumber daya. Skenario tanpa mekanisme keamanan tambahan menjadi acuan untuk mengukur performa dasar, tetapi tidak memberikan perlindungan kriptografi yang memadai terhadap penyadapan dan manipulasi pesan. Penerapan TLS memberikan perlindungan kerahasiaan komunikasi dan menghasilkan nilai delay serta jitter yang lebih rendah daripada kondisi dasar, meskipun penggunaan memorinya meningkat.

Skenario HMAC-SHA-256 menyediakan mekanisme verifikasi integritas dan keaslian pesan tanpa mengenkripsi payload. Pada pengujian ini, skenario tersebut menghasilkan throughput sebesar 302,95 Byte/s, delay sebesar 3,300 ms, penggunaan memori sebesar 99.762,4 Byte, dan penggunaan CPU sebesar 0,21%. Hasil tersebut menunjukkan bahwa kebutuhan sumber daya HMAC perlu dipertimbangkan bersama manfaat verifikasi pesan yang diberikan.

Kombinasi TLS dan HMAC-SHA-256 menghasilkan throughput tertinggi, yaitu 406,15 Byte/s, dan delay terendah, yaitu 3,088 ms. Namun, skenario ini juga mencatat packet loss sebesar 1,35%, penggunaan memori sebesar 144.204,1 Byte, dan penggunaan CPU sebesar 0,21%. Oleh karena itu, hasil pengujian belum menunjukkan bahwa kombinasi kedua mekanisme selalu menghasilkan performa terbaik pada seluruh parameter.

Dari sisi keamanan, kombinasi TLS dan HMAC-SHA-256 menyediakan perlindungan yang saling melengkapi, selama implementasi dan konfigurasi keduanya benar. Dari sisi performa, hasil yang diperoleh menunjukkan adanya perbedaan antarskenario yang perlu dinilai berdasarkan kebutuhan aplikasi, keterbatasan perangkat, dan toleransi terhadap kehilangan paket. Dengan demikian, pemilihan mekanisme keamanan sebaiknya mempertimbangkan tujuan perlindungan, penggunaan sumber daya, serta karakteristik komunikasi yang dibutuhkan oleh aplikasi IoT.

V. KESIMPULAN

Penelitian ini mengevaluasi penerapan TLS dan HMAC-SHA-256 pada komunikasi MQTT dalam sistem IoT melalui empat skenario pengujian, yaitu tanpa mekanisme keamanan tambahan, menggunakan TLS, menggunakan HMAC-SHA-256, serta menggunakan kombinasi keduanya. Hasil evaluasi menunjukkan bahwa TLS dan HMAC-SHA-256 memiliki fungsi yang saling melengkapi. TLS memberikan perlindungan kerahasiaan komunikasi selama transmisi, sedangkan HMAC-SHA-256 memungkinkan verifikasi integritas dan keaslian pesan berdasarkan kunci rahasia bersama. Kombinasi keduanya menyediakan perlindungan yang lebih menyeluruh terhadap aspek keamanan yang diuji, dengan tetap memperhatikan konfigurasi dan implementasi sistem.

Berdasarkan pengukuran performa, skenario kombinasi TLS dan HMAC-SHA-256 menghasilkan *delay* terendah sebesar 3,088 ms dan *throughput* tertinggi sebesar 406,15 Byte/s. Namun, skenario tersebut juga mencatat *packet loss* sebesar 1,35%, penggunaan memori sebesar 144.204,1 Byte, dan penggunaan CPU sebesar 0,21%. Sementara itu, skenario TLS menghasilkan *jitter* terendah sebesar 3,420 ms, sedangkan skenario HMAC-SHA-256 mencatat penggunaan memori terendah sebesar 99.762,4 Byte. Hasil tersebut memperlihatkan bahwa keunggulan suatu mekanisme keamanan bergantung pada parameter performa yang diprioritaskan.

Secara keseluruhan, kombinasi TLS dan HMAC-SHA-256 berpotensi menjadi pilihan untuk aplikasi IoT yang membutuhkan perlindungan kerahasiaan, integritas, dan keaslian pesan secara bersamaan. Meskipun demikian, hasil penelitian ini terbatas pada konfigurasi perangkat dan kondisi pengujian yang digunakan. Penelitian selanjutnya perlu melakukan pengujian berulang dengan kondisi jaringan yang terkontrol, memperluas skenario serangan, dan mengukur variasi penggunaan sumber daya untuk memperoleh kesimpulan yang lebih kuat mengenai efektivitas dan overhead setiap mekanisme keamanan.

DEKLARASI PENGGUNAAN ALAT AI

Selama penyiapan naskah ini, penulis menggunakan ChatGPT untuk menyempurnakan tata bahasa dan pencarian referensi. Setelah menggunakan alat ini, penulis merevisi dan memverifikasi isi teks secara mandiri serta bertanggung jawab penuh atas keseluruhan isi publikasi.

DAFTAR PUSTAKA

[1] R. Chataut, A. Phoummalayvane, and R. Akl, “Unleashing the Power of IoT : A Comprehensive Review of IoT Applications and Future Prospects in Healthcare, Agriculture, Smart Homes, Smart Cities, and Industry 4.0,” *Sensosrs*, vol. 23, 2023.

[2] K. Küçük, A. Sevin, C. Bay, M. A. Ebleme, and Ü. Çavus, “A survey on communication protocols and performance evaluations for Internet of Things,” *Digit. Commun. Networks*, vol. 8, no. November 2019, pp. 1094–1104, 2022, doi: 10.1016/j.dcan.2022.03.013.

[3] M. Al-fayoumi and Q. A. Al-haija, “Capturing low-rate DDoS attack based on MQTT protocol in software Defined-IoT environment,” *Array*, vol. 19, no. June, p. 100316, 2023, doi: 10.1016/j.array.2023.100316.

[4] P. Verma, “MQTT Security Challenges and Solutions,” *Int. J. Enhanc. Res. Sci. Technol. Eng.*, vol. 11, no. 4, 2022, [Online]. Available: https://www.researchgate.net/publication/360919708

[5] M. A. Khan *et al.*, “A deep learning-based intrusion detection system for MQTT enabled IoT,” *Sensors*, vol. 21, no. 21, 2021, doi: 10.3390/s21217016.

[6] S. Chen, S. Jero, M. Jagielski, A. Boldyreva, and C. Nita-rotaru, "Secure Communication Channel Establishment : TLS 1 . 3 (over TCP Fast Open) versus QUIC ∗," *J. Cryptol.*, vol. 34, no. 3, pp. 1–41, 2021, doi: 10.1007/s00145-021-09389-w.

[7] A. R. Alkhafajee, A. M. A. Al-Muqarm, A. H. Alwan, and Z. R. Mohammed, "Security and Performance Analysis of MQTT Protocol with TLS in IoT Networks," in *Proceedings of the 4th International Iraqi Conference on Engineering Technology and Their Applications (IICETA 2021)*, 2021. doi: 10.1109/IICETA51758.2021.9717495.

[8] T. Prantl, L. Iffländer, S. Herrnleben, S. Engel, S. Kounev, and C. Krupitzer, "Performance Impact Analysis of Securing MQTT Using TLS," in *Proceedings of the ACM/SPEC International Conference on Performance Engineering (ICPE 2021)*, 2021. doi: 10.1145/3427921.3450253.

[9] J. Mades, G. Ebelt, B. Janjic, F. Lauer, C. C. Rheinlander, and N. Wehn, "TLS-Level Security for Low Power Industrial IoT Network Infrastructures," in *Proceedings of the 2020 Design, Automation and Test in Europe Conference and Exhibition (DATE 2020)*, 2020. doi: 10.23919/DATE48585.2020.9116285.

[10] M. Backendal, M. Bellare, F. Günther, and M. Scarlata, "When Messages are Keys: Is HMAC a dual-PRF?," in *43rd International Cryptology Conference (CRYPTO 2023)*, 2023. doi: 10.1007/978-3-031-38548-3_22.

[11] M. S. Turan, "Keyed-Hash Message Authentication Code (HMAC)," 2024. doi: 10.6028/NIST.SP.800-224.ipd.

[12] B. Angkasa, Asriyanik, and A. Pambudi, "IMPLEMENTASI ALGORITMA HMAC-SHA-256 UNTUK KEAMANAN PRODUK," *Bit (Fakultas Teknol. Inf. Univ. Budi Luhur)*, vol. 20, no. 2, 2023.

[13] Ö. Şeker, G. Dalkılıç, and U. C. Çabuk, "MARAS: Mutual Authentication and Role-Based Authorization Scheme for Lightweight Internet of Things Applications," *Sensors*, vol. 23, no. 12, 2023, doi: 10.3390/s23125674.

[14] T. Lawrence, I. Ali, T. Christopher, and F. Li, "A bandwidth efficient HMAC-based authentication scheme for network coding," *J. Inf. Secur. Appl.*, vol. 55, 2020, doi: 10.1016/j.jisa.2020.102658.

[15] L. A. Lizama-Pérez, "Digital signatures over HMAC entangled chains," *Eng. Sci. Technol. an Int. J.*, vol. 32, 2022, doi: 10.1016/j.jestch.2021.11.002.

[16] I. Petrescu, E. Niculae, V. Vulturescu, A. Dimitrescu, and L. M. Ungureanu, "Transport and Application Layer Protocols for IoT : Comprehensive Review," *Technologies*, vol. 15, 2025, doi: 10.3390/technologies13120583.

[17] B. Mishra and A. Kertesz, "The Use of MQTT in M2M and IoT Systems : A Survey," *IEEE Access*, vol. 8, pp. 201071–201086, 2020, doi: 10.1109/ACCESS.2020.3035849.

[18] F. Montori, L. Gigli, L. Sciullo, and M. D. I. Felice, "LA-MQTT : Location-aware Publish-subscribe Communications for the Internet of Things," *ACM Trans. Internet Things,* vol. 3, no. 3, 2023.

[19] M. Kashyap, A. K. Dev, and V. Sharma, "Implementation and analysis of EMQX broker for MQTT protocol in the Internet of Things," *e-Prime - Adv. Electr. Eng. Electron. Energy*, vol. 10, no. May, p. 100846, 2024, doi: 10.1016/j.prime.2024.100846.

[20] B. Dowling, M. Fischlin, F. Günther, and D. Stebila, "A Cryptographic Analysis of the TLS 1 . 3 Handshake Protocol," *J. Cryptol.*, vol. 34, no. 4, pp. 1–69, 2021, doi: 10.1007/s00145-021-09384-1.

[21] L. Sharma, M. Jangid, S. Roy, and A. Jain, "Blake2 Enhanced Cyclic Integrity and Authentication Protocol for SDN IoT Security," *Discov. Internet Things*, 2026.