Journal of Applied Informatics and Computing (JAIC)
Vol.10, No.4, August 2026, pp. 4144~4156
e-ISSN: 2548-6861

Performance and Security Analysis of MQTT over TLS on ESP32-Based IoT Systems

M.Azhar [1]*, Muhamad Azwar [2]*, Ondi Asroni [3]*
*Program Studi Teknologi Informasi, Universitas Bumigora
2001020004@universitasbumigora.ac.id [1], azwar@universitasbumigora.ac.id [2], ondi817@gmail.com [3]

Article Info

Article history:

Received 2026-06-14
Revised 2026-08-10
Accepted 2026-08-13

Keyword:

Internet of Things (IoT),
Message Queuing Telemetry Transport (MQTT),
Transport Layer Security (TLS),
ESP32-C3 Super Mini,
Wireshark.

ABSTRACT

The Message Queuing Telemetry Transport (MQTT) protocol is widely used in Internet of Things (IoT) systems due to its lightweight communication mechanism and low resource consumption. However, MQTT does not provide built-in security features, making data transmission vulnerable to packet sniffing and data manipulation attacks. This study aims to analyze the impact of Transport Layer Security (TLS) implementation on the security and performance of MQTT communication in an ESP32-C3 Super Mini-based IoT system. An experimental method was employed by comparing MQTT communication without TLS (port 1883) and with TLS (port 8883) using the HiveMQ Cloud broker. Performance evaluation was conducted by measuring latency and packet size, while security analysis was performed through packet sniffing using Wireshark. The results show that TLS successfully encrypts MQTT payloads, preventing unauthorized access to transmitted data. However, TLS introduces communication overhead, increasing the average latency from 45.25 ms to 51.76 ms (14.39%) and the average packet size from 79 bytes to 103.2 bytes (30.63%) due to the TLS handshake process and encryption mechanism. Despite these increases, the communication performance remained stable and consistent, indicating an acceptable trade-off between security and performance. These findings demonstrate that TLS is a feasible solution for securing MQTT communication in ESP32-based IoT systems while maintaining reliable communication performance.

I. PENDAHULUAN

Perkembangan teknologi Internet of Things (IoT) mengalami peningkatan yang sangat pesat dalam beberapa tahun terakhir [1]. IoT memungkinkan berbagai perangkat fisik untuk saling terhubung dan bertukar data melalui jaringan internet, sehingga mendukung berbagai aplikasi seperti smart home, smart city, industri 4.0, hingga sistem monitoring lingkungan [2]. Perangkat IoT umumnya memiliki keterbatasan sumber daya, seperti daya komputasi, memori, dan konsumsi energi, sehingga membutuhkan protokol komunikasi yang ringan dan efisien [3].

Salah satu protokol yang banyak digunakan dalam sistem IoT adalah Message Queuing Telemetry Transport (MQTT), yang dirancang untuk komunikasi berbasis publish-subscribe dengan bandwidth rendah dan latensi minimal [4]. Keunggulan MQTT terletak pada kesederhanaannya, efisiensi penggunaan sumber daya, serta kemampuannya untuk beroperasi pada jaringan yang tidak stabil [5]. Oleh karena itu, MQTT banyak diimplementasikan pada perangkat IoT berbasis mikrokontroler seperti ESP32.

Namun demikian, MQTT secara default tidak menyediakan mekanisme keamanan yang memadai [6]. Komunikasi MQTT tanpa lapisan keamanan tambahan dilakukan dalam bentuk plaintext, sehingga rentan terhadap berbagai jenis serangan seperti packet sniffing, man-in-the-middle, serta manipulasi data [7]. Kondisi ini menjadi permasalahan serius, terutama ketika sistem IoT digunakan pada jaringan publik atau lingkungan yang tidak terpercaya.

Untuk mengatasi permasalahan tersebut, diperlukan mekanisme keamanan yang mampu melindungi kerahasiaan dan integritas data selama proses komunikasi [8]. Salah satu solusi yang umum digunakan adalah penerapan Transport Layer Security (TLS), yang merupakan protokol kriptografi

untuk menyediakan komunikasi yang aman melalui jaringan komputer [9]. Dengan menerapkan TLS pada MQTT, data yang dikirimkan akan dienkripsi sehingga tidak dapat dibaca oleh pihak yang tidak berwenang [10].

Meskipun TLS mampu meningkatkan keamanan komunikasi, penerapannya pada perangkat IoT dengan sumber daya terbatas menimbulkan tantangan tersendiri, terutama terkait keterbatasan memori, daya komputasi, dan konsumsi energi [11]. Proses enkripsi, dekripsi, serta handshake pada TLS dapat menambah beban komputasi dan meningkatkan latency komunikasi [12]. Selain itu, penggunaan TLS juga dapat menyebabkan peningkatan ukuran paket data akibat adanya tambahan header dan metadata keamanan [13]. Oleh karena itu, diperlukan analisis lebih lanjut untuk mengetahui sejauh mana dampak penerapan TLS terhadap performa komunikasi MQTT pada perangkat IoT.

Beberapa penelitian sebelumnya telah membahas implementasi MQTT dan TLS dalam konteks IoT [14]. Namun, sebagian besar penelitian hanya berfokus pada aspek keamanan tanpa melakukan analisis mendalam terhadap performa sistem, khususnya pada perangkat dengan keterbatasan sumber daya seperti ESP32 [15]. Ringkasan penelitian terdahulu yang menjadi dasar penelitian ini ditunjukkan pada Tabel I.

TABEL I
PERBANDINGAN PENELITIAN TERDAHULU

Peneliti	Fokus Penelitian	Hasil	Keterbatasan
Reihan dkk. (2022)	Implementasi TLS pada MQTT berbasis Raspberry Pi	TLS meningkatkan keamanan komunikasi	Tidak menganalisis latency dan packet size
Seoane dkk. (2021)	Evaluasi performa MQTT dan CoAP dengan mekanisme keamanan	Mekanisme keamanan meningkatkan keamanan tetapi menambah latency dan overhead komunikasi	Tidak menggunakan perangkat ESP32 dan tidak membahas analisis packet size MQTT secara spesifik
Azwar dkk. (2025)	MQTT over TLS pada ESP32	Komunikasi lebih aman menggunakan TLS	Fokus pada implementasi, belum mengukur overhead komunikasi
Eldho dkk. (2025)	Evaluasi performa MQTT pada IoT	MQTT memiliki performa yang baik untuk IoT	Tidak membahas keamanan TLS
Penelitian ini	Analisis MQTT dengan dan tanpa TLS pada ESP32-C3 Super Mini	Mengukur keamanan, latency, packet size, overhead, dan standar deviasi	Memberikan analisis performa dan keamanan secara komprehensif serta membahas trade-off antara keamanan dan performa.

Berdasarkan Tabel I, dapat diketahui bahwa penelitian-penelitian sebelumnya masih memiliki beberapa keterbatasan. Sebagian besar penelitian hanya berfokus pada implementasi TLS sebagai mekanisme keamanan tanpa melakukan evaluasi secara kuantitatif terhadap dampaknya terhadap performa komunikasi. Selain itu, analisis mengenai *latency*, *packet size*, *overhead*, serta konsistensi performa pada implementasi MQTT over TLS menggunakan perangkat ESP32 masih relatif terbatas. Oleh karena itu, diperlukan penelitian yang mampu memberikan evaluasi keamanan dan performa secara bersamaan sehingga diperoleh gambaran yang lebih komprehensif mengenai penerapan TLS pada sistem IoT [16].

Berdasarkan permasalahan tersebut, penelitian ini bertujuan untuk mengimplementasikan Transport Layer Security (TLS) pada komunikasi MQTT berbasis ESP32-C3 Super Mini dan menganalisis pengaruhnya terhadap keamanan serta performa komunikasi. Analisis dilakukan dengan membandingkan komunikasi MQTT tanpa TLS dan MQTT dengan TLS berdasarkan parameter *latency*, *packet size*, *overhead*, serta analisis keamanan melalui pengamatan lalu lintas jaringan menggunakan Wireshark. [17].

Berbeda dengan penelitian sebelumnya yang umumnya hanya berfokus pada implementasi TLS sebagai mekanisme keamanan komunikasi MQTT, penelitian ini tidak hanya mengevaluasi aspek keamanan, tetapi juga menganalisis secara kuantitatif dampak penerapan TLS terhadap performa komunikasi pada ESP32-C3 Super Mini. Kebaruan penelitian ini terletak pada penyajian analisis yang mencakup *latency*, *packet size*, *overhead komunikasi*, dan standar deviasi latency sebagai indikator konsistensi performa, serta evaluasi keamanan berdasarkan hasil analisis paket menggunakan Wireshark. Dengan demikian, penelitian ini memberikan evaluasi yang lebih komprehensif mengenai *trade-off* antara keamanan dan performa pada implementasi MQTT berbasis TLS.

Kontribusi utama penelitian ini adalah memberikan evaluasi empiris mengenai *trade-off* antara keamanan dan performa pada implementasi MQTT dengan TLS menggunakan ESP32-C3 Super Mini dan broker HiveMQ Cloud. Penelitian ini menyajikan analisis kuantitatif berupa *latency*, *packet size*, *overhead*, dan standar deviasi, sehingga dapat menjadi referensi dalam penerapan komunikasi MQTT yang aman pada perangkat IoT dengan keterbatasan sumber daya [18].

Hasil dari penelitian ini diharapkan dapat menjadi referensi bagi pengembang sistem IoT dalam menentukan strategi keamanan yang tepat, khususnya dalam memilih antara efisiensi performa dan tingkat keamanan yang diinginkan. Selain itu, penelitian ini juga dapat menjadi dasar

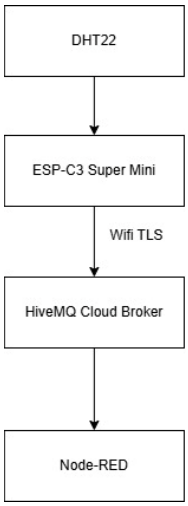
untuk pengembangan lebih lanjut dalam optimalisasi protokol komunikasi yang aman dan efisien untuk perangkat IoT.

II. METODE

Penelitian ini menggunakan metode eksperimen dengan pendekatan kuantitatif untuk menganalisis pengaruh penerapan *Transport Layer Security* (TLS) terhadap performa dan keamanan komunikasi Message *Queuing Telemetry Transport* (MQTT) pada sistem *Internet of Things* (IoT) berbasis ESP32. Eksperimen dilakukan dengan membandingkan dua skenario komunikasi, yaitu MQTT tanpa TLS dan MQTT dengan TLS menggunakan konfigurasi perangkat, jaringan, serta parameter pengujian yang sama. Data hasil pengujian dikumpulkan melalui proses *packet sniffing* menggunakan Wireshark, kemudian dianalisis berdasarkan nilai *latency*, ukuran paket (*packet size*), dan karakteristik keamanan komunikasi.

A. Arsitektur Sistem

Sistem yang dikembangkan terdiri atas ESP32 sebagai *publisher*, sensor DHT22 sebagai sumber data suhu dan kelembapan, broker HiveMQ Cloud sebagai perantara komunikasi, Node-RED sebagai *subscriber*, serta Wireshark sebagai alat analisis lalu lintas jaringan. Seluruh komponen saling terhubung melalui jaringan *Wi-Fi* sehingga komunikasi MQTT dapat berlangsung secara *real-time.* Arsitektur sistem yang digunakan pada penelitian ini ditunjukkan pada Gambar 1.



Gambar 1. Arsitektur Sistem

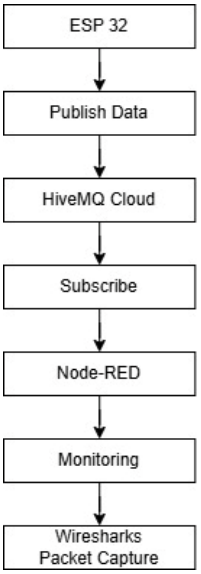
Berdasarkan Gambar 1, ESP32 membaca data dari sensor DHT22 kemudian mengirimkannya ke broker HiveMQ Cloud menggunakan protokol MQTT. Selanjutnya broker meneruskan data kepada Node-RED sebagai *subscriber*. Pada skenario tanpa TLS komunikasi berlangsung melalui port 1883 sehingga isi *payload* masih dapat diamati menggunakan Wireshark. Sebaliknya, pada skenario TLS komunikasi dilakukan melalui port 8883 menggunakan *WiFiClientSecure* sehingga *payload* terenkripsi selama proses transmisi.

B. Skenario Pengujian

Pengujian dilakukan dengan membandingkan dua skenario komunikasi yang memiliki konfigurasi perangkat dan jaringan yang sama. Perbedaan utama terletak pada penggunaan mekanisme keamanan TLS.

Pada skenario pertama, komunikasi MQTT dilakukan tanpa TLS sehingga data dikirim dalam bentuk plaintext melalui port 1883. Pada skenario kedua, komunikasi dilakukan menggunakan TLS melalui port 8883 sehingga seluruh data dienkripsi sebelum ditransmisikan ke broker MQTT.

Setiap skenario diuji sebanyak lima kali Kedua skenario menggunakan konfigurasi perangkat keras, jaringan, serta broker MQTT yang sama sehingga perbedaan hasil pengukuran hanya dipengaruhi oleh penerapan TLS. ESP32 mengirimkan data suhu dan kelembapan setiap 5 detik menggunakan *Quality of Service* (QoS) level 0 melalui topik esp32/dht22/suhu dan esp32/dht22/kelembaban. Dengan konfigurasi tersebut, hasil pengujian dapat dibandingkan secara langsung pada kondisi yang setara. Skenario pengujian yang digunakan ditunjukkan pada Gambar 2.



Gambar 2. Skenario Pengujian

Kedua skenario pengujian menggunakan konfigurasi perangkat dan jaringan yang sama sehingga perbedaan hasil pengukuran hanya dipengaruhi oleh penerapan TLS. ESP32 mengirimkan data suhu dan kelembapan ke broker HiveMQ Cloud menggunakan protokol MQTT dengan *Quality of Service* (QoS) level 0 melalui dua topik, yaitu esp32/dht22/suhu dan esp32/dht22/kelembaban. Data dipublikasikan setiap 5 detik menggunakan koneksi *Wi-Fi*. Pada skenario TLS, komunikasi dilakukan melalui port 8883 menggunakan pustaka *WiFiClientSecure* dan autentikasi broker berbasis *Certificate Authority* (CA), sedangkan skenario tanpa TLS menggunakan port 1883.

C. Parameter pengujian

Parameter pengujian ditetapkan secara konsisten pada kedua skenario komunikasi, yaitu MQTT tanpa TLS dan MQTT dengan TLS, agar hasil pengukuran dapat dibandingkan secara objektif. Seluruh parameter yang digunakan meliputi konfigurasi broker MQTT, *Quality of Service* (QoS), interval pengiriman data, ukuran *payload*, topik MQTT, serta jumlah pengulangan pengujian. Penggunaan parameter yang sama bertujuan untuk memastikan bahwa perbedaan nilai *latency* maupun ukuran paket hanya dipengaruhi oleh penerapan TLS.

TABEL II
PARAMETER PENGUJIAN

Parameter	Nilai
Broker MQTT	HiveMQ Cloud
MQTT Version	3.1.1
Port Tanpa TLS	1883
Port Dengan TLS	8883
QoS	0
Publish Interval	5 detik
Topic	esp32/dht22/suhu, esp32/dht22/kelembaban
Payload	Nilai suhu dan kelembapan (*string*)
Retain	False
Jumlah Pengujian	5 kali setiap skenario

Berdasarkan Tabel II, seluruh parameter diterapkan sama pada kedua skenario pengujian sehingga kondisi eksperimen tetap konsisten. Data sensor dipublikasikan oleh ESP32 setiap 5 detik menggunakan QoS level 0, kemudian diterima oleh Node-RED melalui broker HiveMQ Cloud. Setiap skenario diuji sebanyak lima kali untuk memperoleh nilai rata-rata yang digunakan dalam analisis performa komunikasi.

1) Latency

Latency merupakan waktu yang dibutuhkan paket data untuk berpindah dari perangkat pengirim hingga diterima oleh perangkat penerima. Nilai *latency* diperoleh dari selisih waktu pengiriman dan waktu penerimaan paket berdasarkan informasi *timestamp* yang ditampilkan oleh Wireshark. Persamaan Persamaan (1) digunakan untuk menghitung nilai *latency*.
Keterangan:

$$Latency = t_{terima} - t_{kirim} \quad (1)$$

- *Latency* = waktu tunda komunikasi (ms)
- t_{terima}= waktu paket diterima
- t_{kirim}= waktu paket dikirim

2) Ukuran Paket Data

Ukuran paket data digunakan untuk mengetahui besarnya data yang ditransmisikan pada masing-masing skenario komunikasi. Nilai ini diperoleh dari informasi panjang paket (*packet length*) yang ditampilkan pada hasil *packet capture* Wireshark.

3) Persentase Overhead

Besarnya pengaruh TLS terhadap performa komunikasi dihitung menggunakan persentase *overhead*. Nilai ini menunjukkan peningkatan *latency* maupun ukuran paket data setelah TLS diterapkan. Persamaan (2) digunakan untuk menghitung persentase *overhead*.

$$Overhead(\%) = \frac{NilaiTLS - NilaiTanpaTLS}{NilaiTanpaTLS} \times 100\% \quad (2)$$

Keterangan:

- $Nilai_{TLS}$= hasil pengukuran pada komunikasi menggunakan TLS
- $Nilai_{TanpaTLS}$= hasil pengukuran pada komunikasi tanpa TLS

4) Analisis Keamanan

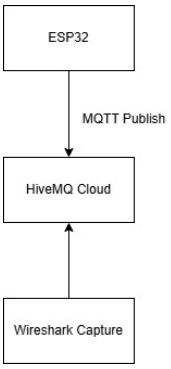
Analisis keamanan dilakukan dengan mengamati bentuk *payload* yang ditransmisikan pada kedua skenario pengujian. Komunikasi tanpa TLS dianggap tidak aman apabila informasi masih dapat dibaca secara langsung dalam bentuk *plaintext*. Sebaliknya, komunikasi dengan TLS dianggap lebih aman apabila *payload* telah berubah menjadi *ciphertext* sehingga tidak dapat diinterpretasikan oleh pihak yang tidak berwenang. Parameter pengujian tersebut selanjutnya digunakan sebagai acuan dalam proses pengumpulan data dan analisis performa komunikasi MQTT pada kedua skenario pengujian.

D. Teknik Pengumpulan Data

Teknik pengumpulan data pada penelitian ini dilakukan melalui metode eksperimen dengan membandingkan dua skenario komunikasi, yaitu MQTT tanpa *Transport Layer Security* (TLS) dan MQTT dengan TLS. Data diperoleh melalui proses *packet sniffing* menggunakan aplikasi Wireshark untuk merekam seluruh paket komunikasi antara ESP32 dan broker HiveMQ Cloud selama proses pengiriman data berlangsung. Setiap skenario pengujian dilakukan sebanyak lima kali dengan konfigurasi perangkat, jaringan, serta parameter komunikasi yang sama sehingga hasil pengukuran dapat dibandingkan secara objektif.

Data yang dikumpulkan meliputi nilai *latency*, ukuran paket (*packet size*), serta karakteristik komunikasi jaringan pada masing-masing skenario pengujian. Selain itu, hasil tangkapan paket menggunakan Wireshark juga digunakan untuk mengamati perbedaan isi *payload* antara komunikasi MQTT tanpa TLS dan MQTT dengan TLS sebagai indikator penerapan mekanisme enkripsi selama proses transmisi data.

Pengukuran *latency* dilakukan berdasarkan selisih waktu antara paket MQTT *PUBLISH* yang dikirim oleh ESP32 dan paket MQTT yang diterima oleh broker pada Wireshark. Nilai waktu diperoleh dari kolom *Time* pada hasil *capture* paket, kemudian dihitung untuk setiap proses pengiriman data. Seluruh hasil pengukuran dicatat pada masing-masing skenario dan selanjutnya digunakan dalam proses analisis performa komunikasi.



Gambar 3. Proses Pengumpulan Data Menggunakan Wireshark

Data hasil pengukuran dari kedua skenario selanjutnya dianalisis menggunakan metode statistik deskriptif untuk memperoleh nilai rata-rata, standar deviasi, serta persentase *packet overhead*. Hasil analisis tersebut digunakan sebagai dasar dalam membandingkan performa komunikasi MQTT tanpa TLS dan MQTT dengan TLS.

E. Teknik Analisis Data

Data yang dikumpulkan meliputi nilai *latency*, ukuran paket (*packet size*), serta karakteristik komunikasi jaringan pada masing-masing skenario pengujian. Selain itu, hasil tangkapan paket menggunakan Wireshark juga digunakan untuk mengamati perbedaan isi *payload* antara komunikasi MQTT tanpa TLS dan MQTT dengan TLS sebagai indikator penerapan mekanisme enkripsi selama proses transmisi data

Selain menghitung nilai rata-rata, penelitian ini juga menghitung standar deviasi untuk mengetahui tingkat variasi data pada setiap skenario pengujian. Nilai standar deviasi digunakan sebagai indikator konsistensi hasil pengukuran, sehingga dapat diketahui apakah data yang diperoleh memiliki penyebaran yang kecil atau besar selama proses eksperimen berlangsung.

Persamaan (3) menghitung Rata-rata (*Mean)*

$$\tilde{x} = \frac{\sum x_i}{n} \quad (3)$$

Keterangan:

- $\bar{x}$= nilai rata-rata
- x_i= data hasil pengujian
- n= jumlah pengujian

Persamaan (3) digunakan untuk menghitung nilai rata-rata dari lima kali pengujian pada setiap skenario. Nilai rata-rata digunakan sebagai representasi hasil pengukuran sehingga dapat menggambarkan performa sistem secara umum dan mengurangi pengaruh variasi pada masing-masing pengujian.

Persamaan (4) Standar Deviasi

$$s = \sqrt{\frac{\sum(xi - \bar{x})^2}{n-1}} \quad (4)$$

Keterangan:

- s= standar deviasi
- x_i= data ke-i
- $\bar{x}$= rata-rata
- n= jumlah data

Persamaan (4) digunakan untuk menghitung standar deviasi dari hasil pengujian. Nilai standar deviasi menunjukkan tingkat penyebaran data terhadap nilai rata-rata sehingga dapat digunakan untuk mengetahui konsistensi hasil pengukuran. Semakin kecil nilai standar deviasi, maka semakin stabil dan konsisten performa sistem yang dihasilkan selama proses pengujian.

F. Perangkat Keras dan Perangkat Lunak

Penelitian ini menggunakan kombinasi perangkat keras (*hardware*) dan perangkat lunak (*software*) yang saling terintegrasi untuk membangun sistem komunikasi MQTT berbasis *Internet of Things* (IoT). Perangkat keras berfungsi sebagai media akuisisi dan pengiriman data sensor, sedangkan perangkat lunak digunakan untuk proses pemrograman, komunikasi data, visualisasi, serta analisis lalu lintas jaringan selama proses pengujian.

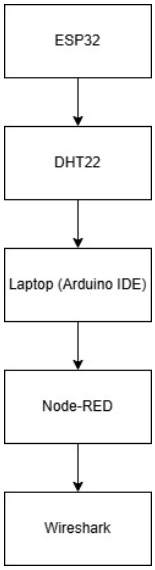
TABEL III

Spesifikasi Perangkat Keras dan Perangkat Lunak

Komponen	Spesifikasi
Mikrokontroler	ESP32-C3 Super Mini, CPU RISC-V 160 MHz, RAM 400 KB, Flash 4 MB
Sensor	DHT22
Broker MQTT	HiveMQ Cloud
IDE	Arduino IDE
Library MQTT	PubSubClient
Library TLS	WiFiClientSecure
Subscriber	Node.js v22.17.0
Analisis Paket	Wireshark 4.6.2
Sistem Operasi	Windows 11 64-bit
Media Komunikasi	*Wi-Fi* 2,4 GHz

ESP32 digunakan sebagai *publisher* yang bertugas membaca data suhu dan kelembapan dari sensor DHT22 kemudian mengirimkannya ke broker HiveMQ Cloud menggunakan protokol MQTT. Pada sisi penerima, Node-RED berfungsi sebagai *subscriber* untuk menerima dan menampilkan data yang dikirim oleh ESP32. Seluruh komunikasi jaringan dianalisis menggunakan Wireshark untuk memperoleh informasi mengenai *latency*, ukuran paket (*packet size*), serta karakteristik komunikasi pada masing-masing skenario pengujian.

Pada skenario komunikasi menggunakan TLS, ESP32 memanfaatkan pustaka *WiFiClientSecure* untuk membangun koneksi terenkripsi dengan broker HiveMQ Cloud melalui port 8883. Sebelum koneksi dilakukan, perangkat melakukan sinkronisasi waktu menggunakan *Network Time Protocol* (NTP) agar proses validasi sertifikat digital dapat berjalan dengan baik. Selanjutnya, *Root CA Certificate* dimuat ke dalam ESP32 menggunakan fungsi setCACert() sehingga komunikasi TLS dapat berlangsung secara aman.



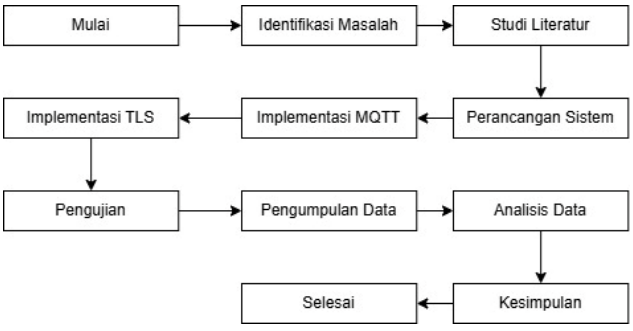
Gambar 4. Perangkat Keras dan Perangkat Lunak yang Digunakan dalam Penelitian

Gambar 4 menunjukkan perangkat keras dan perangkat lunak yang digunakan selama proses penelitian. Perangkat keras terdiri atas ESP32 sebagai *publisher* dan sensor DHT22 sebagai sumber data suhu serta kelembapan. Sementara itu, perangkat lunak yang digunakan meliputi Arduino IDE untuk pemrograman ESP32, HiveMQ Cloud sebagai broker MQTT, Node-RED sebagai *subscriber*, serta Wireshark untuk menangkap dan menganalisis lalu lintas komunikasi selama proses pengujian.

G. Flowchart Sistem

Alur kerja sistem diawali dengan pembacaan data suhu dan kelembaban oleh sensor DHT22. Data yang diperoleh kemudian diproses oleh ESP32 sebelum dikirim ke *broker* MQTT melalui jaringan Wi-Fi. Pada skenario yang menggunakan TLS, ESP32 terlebih dahulu melakukan proses *handshake* dan verifikasi sertifikat sebelum pertukaran data dilakukan.

Data yang berhasil diterima oleh *broker* MQTT kemudian diteruskan ke Node-RED untuk ditampilkan secara *real-time*. Selama proses komunikasi berlangsung, Wireshark digunakan untuk menangkap paket data sebagai bahan analisis performa dan keamanan. Alur kerja sistem secara keseluruhan ditunjukkan pada Gambar 3.



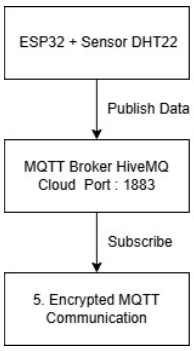
Gambar 5. Flowchart Sistem

Berdasarkan Gambar 5, proses sistem diawali dengan inisialisasi perangkat ESP32, sensor DHT22, serta konfigurasi jaringan *Wi-Fi*. Setelah perangkat berhasil terhubung ke jaringan, ESP32 melakukan sinkronisasi waktu menggunakan *Network Time Protocol* (NTP). Proses ini diperlukan agar sertifikat digital pada saat komunikasi TLS dapat diverifikasi dengan benar. Selanjutnya ESP32 melakukan konfigurasi broker MQTT, memuat *Root CA Certificate*, kemudian membangun koneksi ke broker HiveMQ Cloud.

Setelah koneksi berhasil dibangun, ESP32 membaca data suhu dan kelembapan dari sensor DHT22. Data tersebut kemudian dipublikasikan ke broker MQTT sesuai interval pengiriman yang telah ditentukan. Pada skenario MQTT tanpa TLS, komunikasi dilakukan melalui port 1883, sedangkan pada skenario MQTT dengan TLS komunikasi menggunakan port 8883 sehingga seluruh *payload* dienkripsi selama proses transmisi. Proses pengiriman data berlangsung secara berulang hingga sistem dihentikan.

H. Alur Komunikasi Data

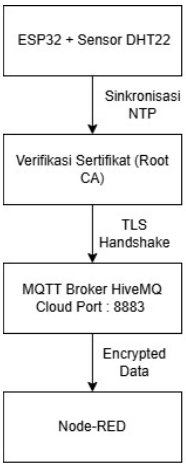
Alur komunikasi data dimulai ketika ESP32 berhasil terhubung ke jaringan *Wi-Fi* dan membangun koneksi dengan broker HiveMQ Cloud. Pada komunikasi MQTT tanpa TLS, proses koneksi berlangsung secara langsung melalui port 1883 menggunakan protokol TCP. Sebaliknya, pada komunikasi MQTT dengan TLS, sebelum data dipublikasikan dilakukan proses *TLS handshake* untuk membentuk kanal komunikasi yang aman.



Gambar 6. Alur Komunikasi MQTT Tanpa TLS

Pada Gambar 6 komunikasi MQTT tanpa TLS, data dikirim langsung dari ESP32 menuju broker HiveMQ Cloud melalui port 1883 tanpa proses enkripsi sehingga *payload* masih dapat dibaca selama proses transmisi.

Pada Gambar 7 komunikasi MQTT dengan TLS, ESP32 melakukan sinkronisasi waktu dan verifikasi sertifikat sebelum membangun koneksi aman (*TLS handshake*). Setelah sesi aman terbentuk, data dikirim dalam keadaan terenkripsi melalui port 8883 menuju broker HiveMQ Cloud dan diterima oleh Node-RED.



Gambar 7. Alur Komunikasi MQTT Dengan TLS

Pada proses *TLS handshake*, ESP32 terlebih dahulu memverifikasi sertifikat digital broker menggunakan *Root CA Certificate* yang telah disimpan pada perangkat. Setelah proses autentikasi berhasil, kedua perangkat melakukan pembentukan sesi komunikasi terenkripsi (*secure session*) sehingga seluruh *payload* MQTT dikirim dalam bentuk terenkripsi. Mekanisme tersebut memberikan jaminan terhadap aspek kerahasiaan (*confidentiality*), integritas (*integrity*), serta autentikasi (*authentication*) selama proses komunikasi berlangsung.

I. Konfigurasi TLS

Untuk meningkatkan keamanan komunikasi MQTT, penelitian ini mengimplementasikan *Transport Layer Security* (TLS) pada komunikasi antara ESP32 dan broker HiveMQ Cloud. Implementasi TLS bertujuan untuk melindungi proses pertukaran data dari ancaman penyadapan (*eavesdropping*), modifikasi data, maupun akses tidak sah selama proses transmisi. Pada penelitian ini komunikasi MQTT tanpa TLS menggunakan port 1883, sedangkan komunikasi MQTT dengan TLS menggunakan port 8883.

ESP32 membangun koneksi TLS menggunakan pustaka *WiFiClientSecure* yang tersedia pada *framework* Arduino IDE. Sebelum melakukan koneksi ke broker, perangkat terlebih dahulu melakukan sinkronisasi waktu menggunakan *Network Time Protocol* (NTP). Sinkronisasi waktu diperlukan agar masa berlaku sertifikat digital dapat diverifikasi dengan benar selama proses autentikasi TLS. Setelah waktu berhasil disinkronkan, *Root CA Certificate* dimuat ke dalam perangkat menggunakan fungsi setCACert() sehingga identitas broker HiveMQ Cloud dapat diverifikasi sebelum sesi komunikasi dibentuk.

TABEL IV
KONFIGURASI TLS YANG DIGUNAKAN

Parameter	Konfigurasi
Protokol	MQTT over TLS
Broker	HiveMQ Cloud
Port	8883
Library TLS	WiFiClientSecure
Sertifikat	Root CA Certificate
Validasi Sertifikat	setCACert()
Sinkronisasi Waktu	NTP
Media Komunikasi	Wi-Fi

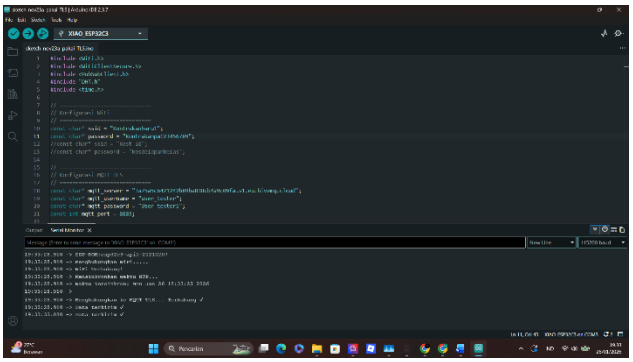
Berdasarkan Tabel 4 konfigurasi tersebut, sebelum proses pengiriman data MQTT dilakukan, ESP32 terlebih dahulu membangun koneksi aman melalui mekanisme *TLS handshake*. Proses ini bertujuan untuk melakukan autentikasi broker dan membentuk sesi komunikasi terenkripsi sebelum *payload* dikirimkan.

III. HASIL DAN PEMBAHASAN

Penelitian ini bertujuan untuk menganalisis dampak penerapan *Transport Layer Security* (TLS) terhadap keamanan dan performa komunikasi pada protokol *Message Queuing Telemetry Transport* (MQTT) berbasis *Internet of Things* (IoT) menggunakan perangkat ESP32. Pengujian dilakukan dengan dua skenario utama, yaitu komunikasi MQTT tanpa TLS dan komunikasi MQTT dengan TLS, untuk mengevaluasi perbedaan dari aspek keamanan serta kinerja sistem.

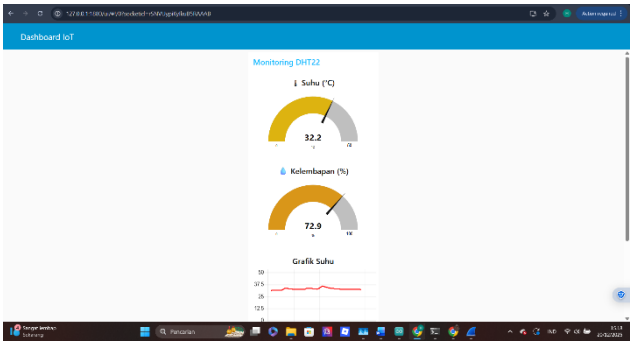
A. Hasil Implementasi Sistem

Sistem IoT pada penelitian ini berhasil diimplementasikan menggunakan ESP32-C3 Super Mini sebagai perangkat utama, sensor DHT22 sebagai sensor suhu dan kelembapan, broker HiveMQ Cloud sebagai server MQTT, serta Node-RED sebagai media visualisasi data. Sistem diuji pada dua skenario komunikasi, yaitu MQTT tanpa TLS dan MQTT dengan TLS, untuk mengevaluasi pengaruh penerapan Transport Layer Security terhadap keamanan dan performa komunikasi data.



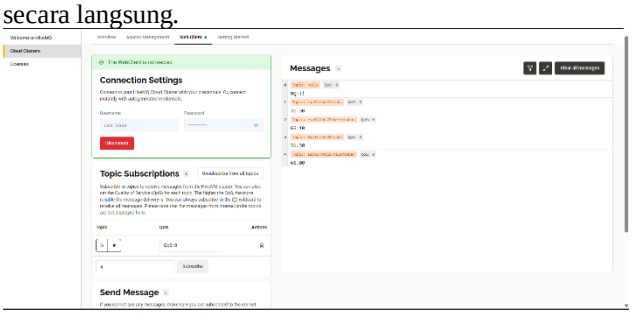
Gambar 8. Hasil Implementasi ESP32-C3 Super Mini pada Komunikasi MQTT over TLS

Berdasarkan Gambar 8, ESP32-C3 Super Mini berhasil terhubung ke jaringan Wi-Fi, melakukan sinkronisasi waktu menggunakan Network Time Protocol (NTP), membangun koneksi MQTT over TLS dengan broker HiveMQ Cloud, dan mengirimkan data sensor secara berkala. Sinkronisasi waktu dilakukan sebelum proses koneksi TLS agar validasi sertifikat digital dapat berjalan dengan benar. Hasil ini menunjukkan bahwa implementasi komunikasi aman menggunakan TLS telah berhasil dilakukan.



Gambar 9. Dashboard Monitoring Data Sensor pada Node-RED

Berdasarkan Gambar 9, data suhu dan kelembapan yang dikirimkan oleh ESP32 berhasil diterima oleh Node-RED dan divisualisasikan secara real-time dalam bentuk gauge dan grafik. Hal ini menunjukkan bahwa proses komunikasi MQTT berjalan dengan baik sehingga data dapat dipantau secara langsung.



Gambar 10. Hasil Pengiriman Data pada Broker HiveMQ Cloud

Berdasarkan Gambar 10, broker HiveMQ Cloud berhasil menerima data yang dipublikasikan oleh ESP32 melalui topik MQTT yang telah dikonfigurasi. Data suhu dan kelembapan muncul pada Web Client sesuai dengan topik publikasi sehingga membuktikan bahwa proses publish dan subscribe berjalan dengan baik.

Berdasarkan hasil implementasi tersebut, seluruh komponen sistem yang terdiri atas ESP32-C3 Super Mini, sensor DHT22, broker HiveMQ Cloud, dan Node-RED berhasil berfungsi sesuai rancangan. Sistem mampu melakukan komunikasi data menggunakan MQTT tanpa TLS maupun MQTT dengan TLS sehingga siap digunakan untuk proses pengujian performa dan analisis keamanan yang dibahas pada subbab berikutnya.

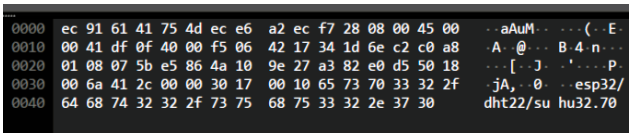
B. Hasil Pengujian MQTT Tanpa TLS

Pengujian MQTT tanpa TLS dilakukan untuk memperoleh nilai dasar (baseline) performa komunikasi sebelum mekanisme keamanan TLS diterapkan. Pada skenario ini, komunikasi dilakukan menggunakan protokol MQTT melalui port 1883 tanpa proses enkripsi. Pengujian dilakukan sebanyak lima kali pengulangan menggunakan payload berukuran tetap, interval pengiriman setiap 5 detik, dan Quality of Service (QoS) level 0. Parameter yang diamati meliputi latency komunikasi dan ukuran paket (packet size).

TABEL V
HASIL PENGUJIAN MQTT TANPA TLS

Pengujian	Latency (ms)	Packet Size (Byte)
1	40,50	79
2	48,72	79
3	45,87	79
4	47,08	79
5	44,08	79

Berdasarkan Tabel 5, hasil pengujian menunjukkan bahwa komunikasi MQTT tanpa TLS menghasilkan rata-rata latency yang relatif rendah karena data dikirim tanpa melalui proses enkripsi maupun autentikasi. Selain itu, ukuran paket yang dikirim juga lebih kecil karena hanya berisi header protokol MQTT dan payload data sensor. Kondisi ini menunjukkan bahwa komunikasi tanpa TLS memberikan performa yang lebih ringan, namun data yang dikirim masih dapat dibaca dalam bentuk plaintext sehingga berpotensi menimbulkan risiko terhadap kerahasiaan informasi apabila terjadi penyadapan jaringan.



Gambar 11. Hasil Capture Paket MQTT Tanpa TLS Menggunakan Wireshark

Berdasarkan Gambar 11, paket MQTT dapat diamati secara langsung menggunakan Wireshark tanpa adanya proses enkripsi. Payload pesan masih dapat dibaca dalam bentuk teks sehingga informasi yang dikirimkan melalui jaringan berpotensi diketahui oleh pihak yang tidak berwenang apabila komunikasi tidak dilindungi oleh mekanisme keamanan tambahan.

Hasil tersebut menunjukkan bahwa MQTT tanpa TLS memiliki keunggulan dari sisi performa karena tidak memerlukan proses enkripsi maupun TLS handshake. Namun demikian, tidak adanya mekanisme kerahasiaan, autentikasi, dan perlindungan integritas menyebabkan komunikasi menjadi lebih rentan terhadap penyadapan maupun modifikasi data selama proses transmisi.

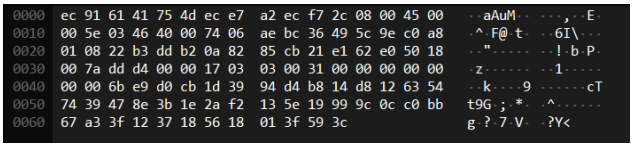
C. Hasil Pengujian MQTT dengan TLS

Pengujian MQTT dengan TLS dilakukan untuk mengevaluasi pengaruh penerapan Transport Layer Security terhadap performa komunikasi data pada sistem IoT. Pada skenario ini komunikasi dilakukan melalui port 8883 menggunakan protokol MQTT over TLS dengan broker HiveMQ Cloud. Pengujian dilakukan sebanyak lima kali pengulangan menggunakan payload yang sama, interval pengiriman data setiap 5 detik, serta Quality of Service (QoS) level 0 sehingga hasil dapat dibandingkan secara langsung dengan skenario MQTT tanpa TLS.

TABEL VI
HASIL PENGUJIAN MQTT DENGAN TLS

Pengujian	Latency (ms)	Packet Size (Byte)
1	52.08	108
2	52.08	108
3	50.46	84
4	52.08	108
5	52.08	108

Berdasarkan Tabel 6, penerapan TLS menyebabkan rata-rata latency lebih tinggi dibandingkan komunikasi MQTT tanpa TLS. Peningkatan tersebut terjadi karena sebelum proses pertukaran data dilakukan, sistem terlebih dahulu melaksanakan proses TLS handshake, validasi sertifikat digital, serta pembentukan sesi komunikasi yang terenkripsi. Selain itu, ukuran paket juga mengalami peningkatan akibat penambahan informasi keamanan yang dibawa oleh protokol TLS.



Gambar 12. Hasil Pengujian Dengan TLS

Berdasarkan Gambar 12, payload MQTT tidak lagi dapat dibaca secara langsung karena seluruh data telah dienkripsi menggunakan protokol TLS. Wireshark hanya mampu menampilkan informasi header komunikasi TLS tanpa memperlihatkan isi pesan sensor. Kondisi ini menunjukkan bahwa mekanisme TLS berhasil menjaga kerahasiaan data selama proses transmisi.

Selain memberikan kerahasiaan data (confidentiality), penggunaan TLS juga menyediakan mekanisme autentikasi server melalui sertifikat digital serta menjaga integritas data menggunakan Message Authentication Code (MAC). Dengan demikian, data yang dikirimkan memiliki perlindungan terhadap modifikasi selama transmisi dan mengurangi risiko serangan seperti penyadapan (*eavesdropping*) maupun *man-in-the-middle*. Namun, penelitian ini belum melakukan simulasi serangan secara langsung sehingga evaluasi keamanan difokuskan pada implementasi mekanisme TLS dan hasil analisis paket menggunakan Wireshark.

D. Analisis Perbandingan MQTT Tanpa TLS dan MQTT dengan TLS

Berdasarkan Gambar 14, payload MQTT tidak lagi dapat dibaca secara langsung karena seluruh data telah dienkripsi menggunakan protokol TLS. Wireshark hanya mampu menampilkan informasi header komunikasi TLS tanpa memperlihatkan isi pesan sensor. Kondisi ini menunjukkan bahwa mekanisme TLS berhasil menjaga kerahasiaan data selama proses transmisi.

TABEL VII

PERBANDINGAN MQTT TANPA TLS DAN MQTT DENGAN TLS

Parameter	MQTT Tanpa TLS	MQTT Dengan TLS
Port Komunikasi	1883	8883
Rata-rata Latency (ms)	45,25	51,76
Standar Deviasi (ms)	3,15	0,72
Packet Size (Byte)q	79	108
Enkripsi Data	Tidak	Ya
Payload Terbaca	Ya	Tidak
Autentikasi Server	Tidak	Ya
Integritas Data	Tidak	Ya

Berdasarkan Tabel 7, rata-rata *latency* pada komunikasi MQTT tanpa TLS sebesar 45,25 ms dengan standar deviasi 3,15 ms, sedangkan pada MQTT dengan TLS meningkat menjadi 51,76 ms dengan standar deviasi 0,72 ms. Nilai standar deviasi yang lebih kecil pada skenario MQTT dengan TLS menunjukkan bahwa waktu transmisi data cenderung lebih konsisten dibandingkan komunikasi tanpa TLS. Meskipun rata-rata *latency* meningkat akibat proses *TLS handshake*, enkripsi, dan autentikasi sertifikat, variasi waktu pengiriman relatif kecil sehingga komunikasi tetap berlangsung secara stabil selama proses pengujian.

Hasil pengujian menunjukkan bahwa rata-rata latency meningkat setelah penerapan TLS. Peningkatan ini merupakan konsekuensi dari mekanisme keamanan tambahan yang dijalankan oleh protokol TLS, seperti negosiasi algoritma kriptografi, pertukaran kunci (*key exchange*), dan proses autentikasi server. Walaupun terjadi peningkatan waktu *transmisi*, komunikasi tetap berlangsung secara stabil selama proses pengujian.

Selain meningkatkan latency, penerapan TLS juga menyebabkan ukuran paket menjadi lebih besar dibandingkan MQTT tanpa TLS. *Penambahan* ukuran paket terjadi karena adanya header TLS, informasi sertifikat, dan data autentikasi yang menyertai setiap proses komunikasi. Meskipun demikian, peningkatan ukuran paket masih tergolong kecil sehingga tidak memberikan dampak signifikan terhadap proses transmisi data sensor.

Dari sisi keamanan, MQTT tanpa TLS hanya menyediakan mekanisme komunikasi dasar sehingga payload dapat dibaca apabila lalu lintas jaringan berhasil disadap. Sebaliknya, MQTT dengan TLS memberikan perlindungan terhadap kerahasiaan (*confidentiality*), autentikasi (*authentication*), dan integritas data (*integrity*) melalui mekanisme enkripsi serta sertifikat digital. Oleh karena itu, penerapan TLS mampu meningkatkan keamanan komunikasi data pada sistem IoT meskipun memerlukan tambahan sumber daya dan waktu transmisi.

Secara keseluruhan, hasil pengujian menunjukkan adanya *trade-off* antara keamanan dan performa. MQTT tanpa TLS menawarkan komunikasi yang lebih cepat dengan ukuran paket yang lebih kecil, sedangkan MQTT dengan TLS memberikan tingkat keamanan yang lebih tinggi dengan konsekuensi peningkatan latency dan ukuran paket. Berdasarkan hasil tersebut, penerapan TLS dinilai layak digunakan pada sistem IoT yang memprioritaskan keamanan data dibandingkan peningkatan performa yang relatif kecil.

E. Analisis Overhead TLS

Analisis *overhead* dilakukan untuk mengetahui dampak penerapan *Transport Layer Security* (TLS) terhadap performa komunikasi MQTT pada sistem IoT. *Overhead* dalam penelitian ini dianalisis berdasarkan perubahan nilai *latency* dan ukuran paket (*packet size*) antara komunikasi MQTT tanpa TLS dan MQTT dengan TLS. Hasil analisis ini digunakan untuk mengevaluasi apakah peningkatan keamanan yang diberikan oleh TLS sebanding dengan penurunan performa yang terjadi.

TABEL VIII

OVERHEAD PENERAPAN TLS PADA KOMUNIKASI MQTT

Parameter	MQTT Tanpa TLS	MQTT Dengan TLS	Overhead
Latency (ms)	45,25	51,76	14,39%
Packet Size (Byte)	79	51,76	36,71%

Berdasarkan Tabel 8, penerapan TLS menyebabkan peningkatan rata-rata *latency* sebesar 14,39% dan peningkatan ukuran paket sebesar 36,71% dibandingkan komunikasi MQTT tanpa TLS. Peningkatan tersebut merupakan konsekuensi dari mekanisme keamanan tambahan yang diterapkan oleh protokol TLS, seperti proses *TLS handshake*, validasi sertifikat digital, pembentukan sesi komunikasi terenkripsi, serta penambahan informasi keamanan pada setiap paket data.

Salah satu faktor utama yang menyebabkan peningkatan *latency* adalah proses *TLS handshake*. Sebelum data sensor dikirimkan, klien dan broker terlebih dahulu melakukan proses negosiasi parameter keamanan, verifikasi sertifikat digital, pertukaran kunci (*key exchange*), serta pembentukan *session key* untuk menghasilkan kanal komunikasi yang aman. Tahapan tersebut membutuhkan waktu pemrosesan dan pertukaran paket tambahan sehingga meningkatkan waktu koneksi awal dibandingkan komunikasi MQTT tanpa TLS. Penelitian sebelumnya juga melaporkan bahwa proses *TLS handshake* memberikan tambahan *computational overhead*, *bandwidth overhead*, dan waktu koneksi pada perangkat IoT yang memiliki sumber daya terbatas. Oleh karena itu, peningkatan *latency* yang diperoleh pada penelitian ini sejalan dengan karakteristik implementasi TLS pada lingkungan IoT[10].

Selain memengaruhi *latency*, penerapan TLS juga meningkatkan ukuran paket yang dikirimkan melalui jaringan. Hal ini disebabkan oleh adanya penambahan header TLS, informasi autentikasi, dan data kriptografi yang diperlukan untuk menjaga kerahasiaan serta integritas komunikasi. Meskipun ukuran paket meningkat, tambahan tersebut masih berada pada batas yang wajar dan tidak mengganggu proses komunikasi pada sistem IoT yang digunakan dalam penelitian ini.

Nilai standar deviasi *latency* pada skenario MQTT dengan TLS sebesar 0,72 ms menunjukkan bahwa variasi waktu transmisi relatif kecil selama lima kali pengujian. Hal ini mengindikasikan bahwa penerapan TLS memberikan waktu komunikasi yang lebih konsisten meskipun menghasilkan *overhead latency* dibandingkan komunikasi MQTT tanpa TLS.

Secara keseluruhan, penerapan TLS memberikan tambahan *overhead* terhadap komunikasi MQTT berupa peningkatan *latency* dan ukuran paket. Namun demikian, peningkatan tersebut masih tergolong dapat diterima apabila dibandingkan dengan manfaat keamanan yang diperoleh, yaitu kerahasiaan data, autentikasi server, dan integritas komunikasi. Oleh karena itu, penggunaan MQTT over TLS dinilai layak diterapkan pada sistem IoT yang memerlukan perlindungan data selama proses transmisi.

F. Perbandingan dengan Penelitian Sebelumnya

Perbandingan dengan penelitian sebelumnya dilakukan untuk mengetahui kesesuaian hasil penelitian ini dengan penelitian lain yang mengimplementasikan *Transport Layer Security* (TLS) pada komunikasi MQTT di lingkungan Internet of Things. Parameter yang dibandingkan meliputi peningkatan *latency*, perubahan ukuran paket (*packet size*), serta dampak penerapan TLS terhadap keamanan komunikasi.

TABEL IX

OVERHEAD PENERAPAN TLS PADA KOMUNIKASI MQTT

Peneliti	Objek Penelitian	Hasil Penelitian	Perbandingan dengan Penelitian Ini
[10]	Implementasi TLS pada MQTT berbasis Raspberry Pi	TLS berhasil meningkatkan keamanan komunikasi, namun menambah *latency* akibat proses enkripsi dan *handshake*.	Sejalan dengan penelitian ini, penerapan TLS meningkatkan *latency* sebesar 14,39%, tetapi komunikasi menjadi lebih aman karena data berhasil dienkripsi.
[13]	Implementasi TLS pada MQTT menggunakan ESP32	TLS meningkatkan keamanan komunikasi pada ESP32 dengan konsekuensi bertambahnya *delay* dan penggunaan sumber daya perangkat.	Hasil penelitian ini juga menunjukkan adanya peningkatan *latency* dan *packet size*, namun masih berada pada batas yang dapat diterima untuk komunikasi IoT.
[18]	MQTT over TLS menggunakan ESP32-C3 Super Mini	Analisis *latency*, *packet size*, *overhead*, standar deviasi, dan	Memberikan analisis yang lebih lengkap mengenai pengaruh TLS terhadap

	dan HiveMQ Cloud	keamanan komunikasi.	performa dan keamanan komunikasi MQTT.

Berdasarkan Tabel 9, hasil penelitian ini menunjukkan kesesuaian dengan penelitian [10] dan [13] yang menyatakan bahwa penerapan *Transport Layer Security* (TLS) pada komunikasi MQTT mampu meningkatkan keamanan data selama proses transmisi, meskipun memberikan tambahan latency akibat proses enkripsi dan *TLS handshake*. Pada penelitian ini, peningkatan *latency* sebesar 14,39% dan peningkatan *packet size* sebesar 36,71% masih berada pada tingkat yang dapat diterima apabila dibandingkan dengan manfaat keamanan yang diperoleh. Hasil tersebut menunjukkan bahwa implementasi TLS tetap layak diterapkan pada sistem IoT yang memerlukan perlindungan terhadap kerahasiaan, autentikasi, dan integritas data.

Meskipun implementasi TLS pada protokol MQTT telah banyak dilaporkan pada penelitian sebelumnya, penelitian ini memiliki beberapa kontribusi yang membedakannya. Pertama, penelitian menggunakan ESP32-C3 Super Mini yang terhubung dengan broker HiveMQ Cloud sebagai lingkungan pengujian. Kedua, penelitian membandingkan secara langsung performa komunikasi MQTT tanpa TLS dan MQTT dengan TLS berdasarkan parameter *latency*, *packet size*, dan analisis paket menggunakan Wireshark. Ketiga, penelitian ini melengkapi hasil pengujian dengan analisis *overhead*, perhitungan standar deviasi *latency*, serta pembahasan *trade-off* antara peningkatan keamanan dan penurunan performa. Dengan demikian, penelitian ini memberikan evaluasi yang lebih komprehensif mengenai implementasi TLS pada komunikasi MQTT berbasis ESP32 di lingkungan IoT.

Secara keseluruhan, hasil penelitian ini memperkuat temuan penelitian terdahulu bahwa penerapan TLS meningkatkan keamanan komunikasi MQTT dengan konsekuensi penurunan performa yang masih berada dalam batas yang dapat diterima. Dengan demikian, penggunaan MQTT over TLS layak diterapkan pada sistem IoT yang memerlukan perlindungan terhadap kerahasiaan dan integritas data selama proses transmisi.

G. Implikasi Penerapan TLS pada Sistem IoT

Hasil penelitian menunjukkan bahwa penerapan *Transport Layer Security* (TLS) pada komunikasi MQTT memberikan peningkatan keamanan dengan konsekuensi adanya *overhead* pada performa komunikasi. Meskipun terjadi peningkatan *latency* dan ukuran paket, nilai tersebut masih berada pada tingkat yang dapat diterima sehingga implementasi TLS tetap layak diterapkan pada berbagai sistem Internet of Things (IoT) yang memerlukan keamanan komunikasi data.

Pada implementasi nyata, perangkat IoT sering digunakan untuk mengirimkan data yang bersifat penting, seperti data lingkungan, kesehatan, industri, maupun *smart home*. Oleh karena itu, perlindungan terhadap kerahasiaan (*confidentiality*), integritas (*integrity*), dan autentikasi (*authentication*) menjadi aspek yang sangat penting. Penggunaan TLS mampu mengurangi risiko penyadapan (*eavesdropping*), manipulasi data selama transmisi, serta serangan *man-in-the-middle* melalui mekanisme enkripsi dan verifikasi sertifikat digital.

Di sisi lain, penerapan TLS menimbulkan tambahan *latency* sebesar 14,39% dan peningkatan ukuran paket sebesar 36,71%. Namun, berdasarkan hasil pengujian, peningkatan tersebut tidak menyebabkan gangguan terhadap proses pengiriman data sensor yang dilakukan setiap lima detik. Dengan demikian, terdapat *trade-off* antara keamanan dan performa, di mana sedikit penurunan performa dapat diterima untuk memperoleh komunikasi yang jauh lebih aman.

Pada implementasi IoT berskala besar [13] yang melibatkan banyak perangkat (*multi-node*), penerapan TLS diperkirakan akan meningkatkan kebutuhan sumber daya jaringan dan komputasi karena setiap perangkat harus melakukan proses *TLS handshake* dan enkripsi data. Oleh sebab itu, perencanaan kapasitas broker, bandwidth jaringan, serta spesifikasi perangkat perlu diperhatikan agar sistem tetap mampu mempertahankan performa komunikasi ketika jumlah perangkat meningkat.\

Secara keseluruhan, hasil penelitian menunjukkan bahwa penerapan TLS pada komunikasi MQTT merupakan pilihan yang tepat bagi sistem IoT yang mengutamakan keamanan data. Meskipun terdapat tambahan *overhead* pada performa komunikasi, manfaat yang diperoleh berupa peningkatan kerahasiaan, autentikasi, dan integritas data menjadikan TLS sebagai mekanisme keamanan yang layak diterapkan pada berbagai aplikasi IoT.

H. Keterbatasan Penelitian

Penelitian ini memiliki beberapa keterbatasan yang perlu diperhatikan dalam menginterpretasikan hasil pengujian. Pengujian hanya dilakukan menggunakan satu jenis perangkat IoT, yaitu ESP32-C3 Super Mini, yang terhubung dengan broker HiveMQ Cloud melalui satu topologi jaringan. Oleh karena itu, hasil yang diperoleh belum dapat digeneralisasikan untuk seluruh jenis perangkat IoT maupun lingkungan jaringan yang berbeda.

Selain itu, parameter performa yang dianalisis pada penelitian ini terbatas pada *latency* dan ukuran paket (*packet size*). Parameter lain seperti *throughput*, penggunaan CPU, konsumsi memori, konsumsi energi, serta waktu yang dibutuhkan selama proses *TLS handshake* belum diukur secara terpisah sehingga masih diperlukan penelitian lanjutan untuk memperoleh evaluasi performa yang lebih komprehensif.

Evaluasi keamanan pada penelitian ini difokuskan pada implementasi TLS dan analisis paket menggunakan Wireshark untuk menunjukkan bahwa payload berhasil dienkripsi selama proses transmisi. Penelitian ini belum melakukan simulasi serangan secara langsung, seperti *man-in-the-middle attack*, *replay attack*, maupun *certificate spoofing*, sehingga efektivitas TLS terhadap berbagai jenis ancaman belum dievaluasi secara eksperimental.

Penelitian ini juga belum mengevaluasi performa komunikasi pada kondisi jaringan yang berbeda, seperti

bandwidth rendah, *packet loss*, maupun *latency* jaringan yang tinggi. Selain itu, pengujian hanya dilakukan pada satu publisher dan satu subscriber sehingga belum merepresentasikan kondisi implementasi IoT berskala besar yang melibatkan banyak perangkat secara bersamaan.

Oleh karena itu, penelitian selanjutnya disarankan untuk melakukan pengujian pada berbagai jenis perangkat IoT, kondisi jaringan yang lebih beragam, serta skenario komunikasi berskala besar. Selain itu, penelitian berikutnya dapat membandingkan TLS dengan mekanisme keamanan lainnya, seperti DTLS, VPN, maupun autentikasi berbasis token atau sertifikat, sehingga diperoleh evaluasi yang lebih komprehensif mengenai keamanan komunikasi pada sistem IoT.

IV. KESIMPULAN

Berdasarkan hasil penelitian yang telah dilakukan, penerapan *Transport Layer Security* (TLS) pada protokol *Message Queuing Telemetry Transport* (MQTT) terbukti mampu meningkatkan keamanan komunikasi data pada sistem *Internet of Things* (IoT) berbasis ESP32-C3 Super Mini. Pada skenario komunikasi tanpa TLS, payload masih dapat dibaca melalui proses *packet sniffing* menggunakan Wireshark, sedangkan setelah TLS diterapkan, payload berhasil dienkripsi sehingga tidak dapat dibaca oleh pihak yang tidak berwenang. Selain meningkatkan kerahasiaan data (*confidentiality*), TLS juga mendukung autentikasi server melalui *Certificate Authority* (CA) serta menjaga integritas data selama proses transmisi.

Hasil pengujian menunjukkan bahwa penerapan TLS menyebabkan peningkatan rata-rata *latency* dari 45,25 ms menjadi 51,76 ms, atau meningkat sebesar 14,39%. Selain itu, rata-rata ukuran paket meningkat dari 79 byte menjadi 103,2 byte, atau sebesar 30,63%, sebagai konsekuensi dari proses *TLS handshake*, enkripsi, dan penambahan metadata keamanan. Nilai standar deviasi *latency* yang relatif kecil menunjukkan bahwa performa komunikasi tetap konsisten selama pengujian, sehingga peningkatan *overhead* tersebut masih berada pada tingkat yang dapat diterima untuk implementasi IoT.

Penelitian ini memberikan kontribusi berupa analisis empiris mengenai *trade-off* antara keamanan dan performa pada implementasi MQTT dengan TLS menggunakan ESP32-C3 Super Mini dan broker HiveMQ Cloud. Hasil penelitian menunjukkan bahwa penerapan TLS layak digunakan pada sistem IoT yang membutuhkan keamanan komunikasi, karena peningkatan *overhead* yang terjadi masih sebanding dengan manfaat yang diperoleh dalam menjaga kerahasiaan, integritas, dan autentikasi data selama proses komunikasi.

DAFTAR PUSTAKA

[1] N. Crysostomus, K. Dewi, and D. P. Kesuma, "Perkembangan dan Implementasi Internet of Things di Berbagai Sektor : Systematic Literature Review," vol. 5, no. 1, pp. 1–11, 2025.

[2] I. Imam and M. N. Z. M. N. Zamzami, "Integrasi WSN dan IoT Untuk Sistem Monitoring Rumah Cerdas Berbasis MQTT," *Karapan Netw. J. J. Comput. Technol. Mob. Ad Hoc Netw.*, vol. 1, no. 01, 2025.

[3] M. Y. A. Saputra and Y. Yulindon, "Kajian Pustaka Komparatif Protokol Komunikasi Internet of Things Berdasarkan Latensi, Efesiensi Energi, dan Skalabilitas," *J. Ilm. Penelit. Mhs.*, vol. 4, no. 2, pp. 275–288, 2026.

[4] N. F. Junior, A. A. A. Silva, A. E. Guelfi, and S. T. Kofuji, "Performance evaluation of publish-subscribe systems in IoT using energy-efficient and context-aware secure messages," 2022.

[5] M. N. Nashir, E. S. Ni'mah, M. Fathurahman, R. D. Pramudya, and A. Arfriandi, "Evaluasi Protokol Komunikasi Berbasis Web dan Non-Web pada Sistem IoT: Suatu Systematic Literature Review: Web-based and Non-Web-based IoT Communication Protocols: A Systematic Literature Review," *SISFOTENIKA*, vol. 16, no. 1, pp. 70–81, 2026.

[6] M. Hamdi, S. Rifka, and R. Dewi, "Implementasi Security Offloading Pada Iot Gateway Menggunakan Mqtt Over Tls Dan Firewall Untuk Mengatasi Keterbatasan Keamanan Perangkat IoT," *J. Inform. dan Tek. Elektro Terap.*, vol. 14, no. 2, 2026.

[7] K. Monika, D. Pertiwi, and A. D. Azahra, "Deteksi Serangan DoS pada IoT Berbasis MQTT Menggunakan XGB dan PSO," pp. 2272–2282, 2025, doi: 10.33364/algoritma/v.22-2.2623.

[8] S. U. Anggono, E. Siswanto, and L. R. H. A. Fajri, "User interface berbasis web pada perangkat Internet of Things," *Tek. J. Ilmu Tek. dan Inform.*, vol. 3, no. 1, pp. 35–54, 2023.

[9] R. Singh, A. Gehlot, S. Vaseem, A. Kumar, and D. Buddhi, "Forest 4 . 0 : Digitalization of forest using the Internet of Things (IoT)," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 34, no. 8, pp. 5587–5601, 2022, doi: 10.1016/j.jksuci.2021.02.009.

[10] M. Reihan, I. Rafi, U. B. Hanafi, T. Irfan, and K. Kunci, "Implementasi TLS Sebagai Metode Keamanan Protokol Jaringan Pada MQTT Berbasis Raspberry PI," pp. 13–14, 2022.

[11] A. M. Rivai and M. R. Setiawan, "Internet of Things (IoT) Sebagai Pilar Keamanan Data Pada Sistem Distribusi Di Indonesia," vol. 3, no. 12, pp. 332–341, 2025.

[12] R. S. Siregar *et al.*, "Implementasi Internet of Thing (IOT) untuk Monitoring dan Pengaman Beban Listrik 3 Phase Menggunakan Fuzzy Sugeno," pp. 1013–1025, 2025, doi: 10.33364/algoritma/v.22-1.2343.

[13] H. Azwar, F. Gary, and H. Azwar, "Implementasi Transport Layer Security dengan Algoritma AES untuk Meningkatkan Keamanan Komunikasi pada Jaringan IoT Berbasis MQTT Menggunakan ESP32," vol. 11, no. 1, 2025, doi: 10.35143/elementer.v11i1.

[14] N. Indrasari, A. S. Rohman, T. Genarsih, A. Dwinanda, and S. L. Putri, "Analisis Performa Protokol Mqtt Pada Sistem Monitoring Gas Berbahaya Menggunakan Sensor Mq135 Terintegrasi Android," vol. XV, no. 1, pp. 16–25, 2026, doi: 10.35508/jme.v0i0.28118.

[15] F. K. Sinaga, A. Rabiula, and U. Jambi, "Evaluasi Sistem Penyiraman Tanaman Berbasis ESP32 Hamparan Di Kelurahan Bakung Jaya," vol. 10, no. 1, pp. 351–362, 2026.

[16] K. J. Eldho, T. J. John, D. J. V, and R. V Balakrishnan, "Optimizing Data Transfer Speed and the Performance Evaluation of MQTT in IoT : A Study on MQTT for Atmospheric Condition Monitoring," vol. 10, pp. 231–244, 2025, doi: 10.1109/ICC.2015.7248826.

[17] L. P. Antoro, A. I. Rizal, A. S. Wardhana, M. Zaky, and Z. Muhtadi, "Implementasi Sistem Keamanan Untuk Mendeteksi Gerakan Berbasis Iot Menggunakan ESP32 -CAM," vol. 4, no. November, pp. 1120–1131, 2024.

[18] V. Seoane, C. Garcia-rubio, F. Almenares, and C. Campo, "Performance evaluation of CoAP and MQTT with security support for IoT environments," vol. 197, no. June, 2021.