

IMPLEMENTASI DAN ANALISIS KEAMANAN PROTOKOL MQTT TLS TERHADAP SERANGAN PACKET SNIFFING PADA SISTEM IOT BERBASIS ESP32

Nova Disjayanto, Transisma Budiarta, Asep Suryanta

Rekayasa Keamanan Siber, Politeknik Angkatan Darat

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

mhs.cyber8.novadj@poltekad.ac.id

ABSTRAK

Perkembangan Internet of Things (IoT) yang semakin pesat meningkatkan kebutuhan terhadap sistem komunikasi data yang aman dan realtime. Salah satu protokol yang umum digunakan pada sistem IoT adalah Message Queuing Telemetry Transport (MQTT) karena ringan dan efisien dalam penggunaan bandwidth, namun MQTT standar masih memiliki kelemahan keamanan karena data dapat dibaca secara langsung melalui packet sniffing menggunakan Wireshark. Penelitian ini bertujuan mengimplementasikan dan menganalisis keamanan protokol MQTT berbasis Transport Layer Security (TLS) terhadap serangan packet sniffing pada sistem IoT berbasis ESP32. Metode penelitian dilakukan dengan merancang sistem menggunakan ESP32 dan sensor DHT22 untuk membaca data suhu dan kelembaban, kemudian data dikirim melalui protokol MQTT TLS menuju broker MQTT dan ditampilkan secara realtime pada dashboard Node-RED. Selanjutnya dilakukan pengujian dan analisis keamanan menggunakan Wireshark dengan membandingkan komunikasi MQTT tanpa TLS dan MQTT TLS. Hasil penelitian menunjukkan bahwa implementasi MQTT TLS mampu meningkatkan keamanan komunikasi data pada sistem IoT, ditandai dengan payload pada MQTT tanpa TLS masih dapat dibaca secara langsung, sedangkan pada MQTT TLS berubah menjadi TLS Application Data sehingga isi payload terenkripsi dan tidak dapat dibaca. Dengan demikian, MQTT TLS efektif meningkatkan keamanan komunikasi IoT berbasis ESP32 dari risiko packet sniffing.

Kata Kunci: IoT, ESP32, MQTT, TLS, Wireshark, Packet Sniffing, Node-RED.

1. PENDAHULUAN

Perkembangan teknologi Internet of Things (IoT) pada era digital terus mengalami peningkatan sehingga mendorong pemanfaatan berbagai perangkat yang saling terhubung melalui jaringan internet untuk melakukan proses monitoring, pengendalian, dan pertukaran data secara realtime. Implementasi IoT telah banyak diterapkan pada berbagai bidang, seperti industri, pertanian, kesehatan, transportasi, smart home, hingga sistem pertahanan dan keamanan karena mampu meningkatkan efisiensi serta kemudahan dalam pengelolaan data [1].

Salah satu perangkat yang banyak digunakan dalam implementasi IoT adalah ESP32, yaitu mikrokontroler yang telah dilengkapi dengan modul WiFi dan Bluetooth terintegrasi. ESP32 memiliki performa yang baik, konsumsi daya yang relatif rendah, serta biaya implementasi yang terjangkau sehingga banyak dimanfaatkan dalam pengembangan sistem monitoring berbasis IoT. Pada penelitian ini, ESP32 digunakan sebagai perangkat utama untuk membaca data dari sensor DHT22 dan mengirimkan data sensor menuju broker MQTT.

Komunikasi data merupakan bagian penting dalam sistem IoT. Salah satu protokol yang banyak digunakan adalah Message Queuing Telemetry Transport (MQTT) karena memiliki ukuran paket yang kecil, ringan, dan efisien dalam penggunaan bandwidth. Meskipun demikian, MQTT standar belum memiliki mekanisme keamanan bawaan sehingga data yang dikirimkan masih berpotensi mengalami penyadapan (packet sniffing), manipulasi data,

maupun serangan *man-in-the-middle*. Pada komunikasi MQTT tanpa enkripsi, payload data masih dapat dibaca secara langsung menggunakan aplikasi analisis jaringan seperti Wireshark [3].

Untuk meningkatkan keamanan komunikasi data, diperlukan penerapan Transport Layer Security (TLS) pada protokol MQTT. TLS berfungsi mengenkripsi komunikasi antara client dan broker MQTT sehingga kerahasiaan, integritas, dan autentikasi data dapat terjaga selama proses transmisi. Sehingga data yang dikirimkan tidak dapat dibaca oleh pihak yang tidak berwenang. Dengan penerapan MQTT TLS, payload yang sebelumnya dapat ditampilkan dalam bentuk *plaintext* akan berubah menjadi TLS Application Data, sehingga kerahasiaan dan integritas data dapat terjaga selama proses transmisi [4].

Pada penelitian ini dirancang sistem monitoring berbasis IoT menggunakan ESP32 dan sensor DHT22 untuk membaca data suhu dan kelembaban. Data sensor dikirimkan melalui komunikasi MQTT TLS menuju broker MQTT dan ditampilkan secara realtime pada dashboard Node-RED. Selanjutnya dilakukan proses packet capture menggunakan Wireshark untuk menganalisis perbedaan komunikasi MQTT tanpa TLS dan MQTT TLS terhadap serangan packet sniffing [5].

Berdasarkan permasalahan tersebut, penelitian ini bertujuan untuk mengimplementasikan dan menganalisis keamanan protokol MQTT TLS terhadap serangan *packet sniffing* pada sistem IoT berbasis ESP32. Hasil penelitian diharapkan dapat

menunjukkan efektivitas penggunaan TLS dalam meningkatkan keamanan komunikasi data, sehingga komunikasi data pada sistem IoT menjadi lebih aman, terenkripsi, dan mampu mengurangi risiko kebocoran informasi akibat serangan packet sniffing dan tidak mudah disadap oleh pihak yang tidak berwenang [6].

2. TINJAUAN PUSTAKA

2.1. Internet of Things (IoT)

Internet of Things (IoT) merupakan konsep yang memungkinkan berbagai perangkat elektronik saling terhubung melalui jaringan internet untuk melakukan pertukaran data dan komunikasi secara otomatis. Teknologi IoT mendukung proses monitoring dan pengendalian sistem secara realtime dengan memanfaatkan perangkat seperti sensor, mikrokontroler, aktuator, dan server. Saat ini, IoT telah banyak diterapkan pada berbagai bidang, seperti industri, pertanian, kesehatan, transportasi, smart home, dan sistem keamanan. Pada penelitian ini, IoT diimplementasikan menggunakan ESP32 dan sensor DHT22 untuk melakukan monitoring suhu dan kelembaban secara realtime melalui komunikasi MQTT TLS [1]

2.2. ESP32

ESP32 merupakan mikrokontroler yang dikembangkan oleh Espressif Systems dan banyak digunakan pada sistem Internet of Things (IoT) karena telah dilengkapi dengan modul WiFi dan Bluetooth terintegrasi. Selain memiliki performa yang baik dan konsumsi daya yang relatif rendah, ESP32 juga mendukung berbagai protokol komunikasi, seperti MQTT dan HTTPS, serta implementasi Transport Layer Security (TLS). Dukungan tersebut memungkinkan ESP32 digunakan sebagai perangkat utama untuk membangun sistem monitoring berbasis IoT dengan komunikasi data yang aman melalui proses enkripsi [2].

2.3. Sensor DHT22

Sensor DHT22 merupakan sensor digital yang digunakan untuk mengukur suhu dan kelembaban udara dengan tingkat akurasi yang lebih baik dibandingkan DHT11. Pada penelitian ini, sensor DHT22 berfungsi sebagai perangkat input untuk memperoleh data suhu dan kelembaban lingkungan, kemudian data tersebut diproses oleh ESP32 dan dikirimkan menuju broker MQTT menggunakan protokol MQTT TLS untuk ditampilkan pada dashboard monitoring secara realtime [3].

2.4. Message Queuing Telemetry Transport (MQTT)

Message Queuing Telemetry Transport (MQTT) merupakan protokol komunikasi ringan berbasis mekanisme *publish-subscribe* yang dirancang untuk perangkat dengan keterbatasan bandwidth dan sumber daya. MQTT menggunakan broker sebagai perantara komunikasi antara *publisher* dan *subscriber*, sehingga

mampu mendukung pertukaran data secara realtime dengan penggunaan bandwidth yang rendah. Karakteristik tersebut menjadikan MQTT sebagai salah satu protokol yang banyak digunakan pada sistem Internet of Things (IoT) [4].

2.5. MQTT TLS

Meskipun MQTT merupakan protokol komunikasi yang ringan dan efisien, protokol ini tidak memiliki mekanisme keamanan bawaan sehingga data yang dikirimkan masih berpotensi disadap melalui serangan *packet sniffing*. Untuk meningkatkan keamanan komunikasi, MQTT dapat dikombinasikan dengan **Transport Layer Security (TLS)** yang mengenkripsi komunikasi antara *client* dan *broker*. Penerapan MQTT TLS mampu menjaga kerahasiaan, integritas, dan autentikasi data sehingga komunikasi pada sistem Internet of Things (IoT) menjadi lebih aman [5].

2.6. Transport Layer Security (TLS)

Transport Layer Security (TLS) merupakan protokol keamanan yang digunakan untuk melindungi komunikasi data pada jaringan komputer melalui proses enkripsi. TLS menjaga kerahasiaan (*confidentiality*), integritas (*integrity*), dan autentikasi (*authentication*) data dengan melakukan proses *handshake* untuk pertukaran kunci enkripsi sebelum komunikasi berlangsung. Pada penelitian ini, TLS diterapkan untuk mengamankan komunikasi MQTT antara ESP32 dan broker sehingga data yang dikirimkan terlindungi dari akses pihak yang tidak berwenang [6]

2.7. Node-RED

Node-RED merupakan platform pemrograman berbasis *flow* yang digunakan untuk mengintegrasikan perangkat IoT melalui antarmuka visual berbasis *drag and drop*. Node-RED mendukung berbagai protokol komunikasi, termasuk MQTT, sehingga banyak dimanfaatkan sebagai dashboard monitoring pada sistem IoT. Pada penelitian ini, Node-RED digunakan untuk menerima dan menampilkan data suhu serta kelembaban yang dikirimkan oleh ESP32 melalui broker MQTT secara **realtime** [7]

2.8. Wireshark dan Packet Sniffing

Wireshark merupakan perangkat lunak analisis jaringan yang digunakan untuk menangkap dan menganalisis paket data yang melintas pada suatu jaringan komputer. Pada penelitian ini, Wireshark dimanfaatkan untuk melakukan *packet sniffing* guna membandingkan komunikasi MQTT tanpa TLS dan MQTT TLS. Hasil *capture* menunjukkan bahwa payload pada MQTT tanpa TLS masih dapat dibaca dalam bentuk *plaintext*, sedangkan pada MQTT TLS payload telah berubah menjadi **TLS Application Data** sehingga isi data tidak dapat dibaca secara langsung karena telah terenkripsi [8].

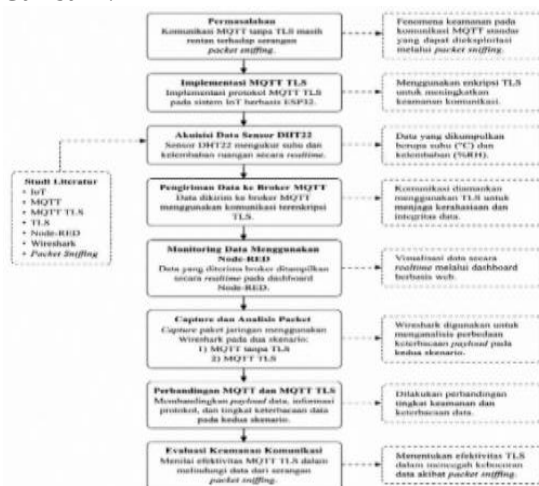
2.9. Penelitian Terdahulu

Penelitian mengenai penerapan keamanan pada komunikasi MQTT telah banyak dilakukan. Hamid Azwar dan Felix Gary [11] menunjukkan bahwa implementasi TLS dan AES pada komunikasi MQTT berbasis ESP32 mampu meningkatkan keamanan data, meskipun menyebabkan peningkatan penggunaan RAM dan delay yang masih berada dalam batas wajar. Selanjutnya, penelitian oleh Amir Akbar Wicaksono dkk. [12] membuktikan bahwa MQTT merupakan protokol yang ringan, efisien dalam penggunaan bandwidth, serta mampu mendukung komunikasi data secara realtime pada sistem IoT berbasis ESP32.

Penelitian lain yang dilakukan oleh Miftahul Hamdi dkk. [10] menunjukkan bahwa penerapan MQTT over TLS pada IoT Gateway mampu meningkatkan keamanan komunikasi tanpa memberikan penurunan performa yang signifikan. Selain itu, penelitian Muhammad dkk. [16] menyimpulkan bahwa komunikasi MQTT menggunakan TLS memberikan perlindungan payload yang lebih baik dibandingkan komunikasi MQTT tanpa enkripsi. Berdasarkan penelitian-penelitian tersebut, penelitian ini memiliki perbedaan dengan melakukan implementasi MQTT TLS pada sistem IoT berbasis ESP32 serta menganalisis efektivitasnya terhadap serangan packet sniffing menggunakan Wireshark.

2.10. Kerangka Pemikiran

Kerangka pemikiran penelitian ditunjukkan pada Gambar 1.



Gambar 1. Kerangka Pemikiran

Berdasarkan studi literatur, komunikasi MQTT tanpa mekanisme keamanan tambahan masih rentan terhadap serangan *packet sniffing* karena payload data dapat dibaca secara langsung. Oleh karena itu, penelitian ini menerapkan protokol MQTT TLS pada sistem IoT berbasis ESP32 untuk mengenkripsi komunikasi data. Selanjutnya, dilakukan analisis menggunakan Wireshark dengan membandingkan komunikasi MQTT tanpa TLS dan MQTT TLS guna

mengevaluasi efektivitas TLS dalam meningkatkan keamanan komunikasi data.

3. METODE PENELITIAN

3.1. Tahapan Penelitian

Tahapan penelitian diawali dengan studi literatur untuk mengumpulkan referensi yang berkaitan dengan Internet of Things (IoT), ESP32, sensor DHT22, protokol MQTT, Transport Layer Security (TLS), Node-RED, dan teknik *packet sniffing* menggunakan Wireshark. Selanjutnya dilakukan perancangan sistem yang meliputi topologi jaringan, diagram blok, serta alur komunikasi antara ESP32, sensor DHT22, broker MQTT, Node-RED, dan Wireshark. Setelah perancangan selesai, dilakukan implementasi sistem dengan mengonfigurasi ESP32 agar dapat mengirimkan data suhu dan kelembaban melalui protokol MQTT menggunakan komunikasi MQTT tanpa TLS dan MQTT TLS.

Tahap berikutnya adalah pengujian sistem dengan membandingkan komunikasi MQTT tanpa TLS dan MQTT TLS menggunakan Wireshark untuk melakukan *packet capture* dan menganalisis keamanan komunikasi data. Selain itu, dilakukan pengujian keberhasilan pengiriman data ke dashboard Node-RED secara *realtime*. Hasil pengujian kemudian dianalisis berdasarkan keterbacaan *payload*, keberhasilan komunikasi, serta perbandingan tingkat keamanan kedua metode sehingga dapat diperoleh kesimpulan mengenai efektivitas implementasi MQTT TLS pada sistem IoT berbasis ESP32.

3.2. Spesifikasi Perangkat Keras

Tabel 1. Perangkat keras penelitian

No	Perangkat	Fungsi
1	ESP32 DevKit V1	Mikrokontroler
2	DHT22	Sensor suhu dan kelembaban
3	Laptop/PC	Broker MQTT, Node-RED, Wireshark
4	Router WiFi	Media komunikasi jaringan

Berdasarkan Tabel 1, sistem penelitian menggunakan ESP32 DevKit V1 sebagai mikrokontroler utama, sensor DHT22 sebagai perangkat akuisisi data suhu dan kelembaban, serta laptop atau PC untuk menjalankan broker MQTT, Node-RED, dan Wireshark. Seluruh perangkat terhubung melalui jaringan WiFi yang disediakan oleh router sehingga proses pengiriman, monitoring, dan analisis keamanan komunikasi data dapat dilakukan secara realtime.

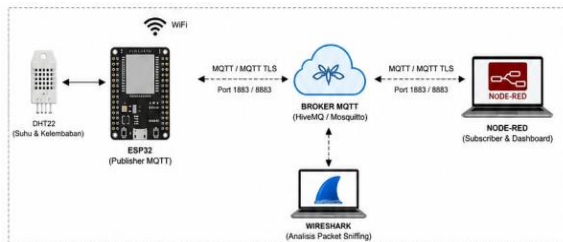
3.3. Spesifikasi Perangkat Lunak

Tabel 2. Perangkat lunak penelitian

No	Software	Fungsi
1	Arduino IDE	Pemrograman ESP32
2	Node-RED	Dashboard Monitoring
3	Wireshark	Packet Capture
4	Mosquitto/HiveMQ	Broker MQTT

Berdasarkan Tabel 2, penelitian ini menggunakan Arduino IDE untuk pemrograman ESP32, Node-RED sebagai dashboard monitoring, Wireshark untuk melakukan analisis paket jaringan, serta Mosquitto atau HiveMQ sebagai broker MQTT. Kombinasi perangkat lunak tersebut mendukung proses implementasi, monitoring data secara *realtime*, dan pengujian keamanan komunikasi MQTT maupun MQTT TLS pada sistem IoT berbasis ESP32.

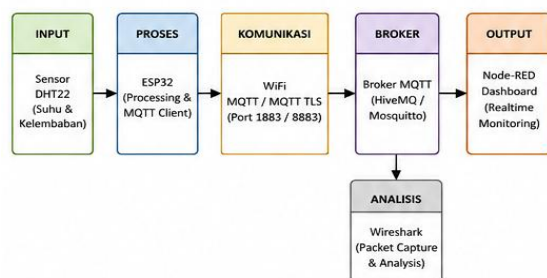
3.4. Topologi Sistem



Gambar 2. Topologi Sistem Penelitian

Berdasarkan Gambar 2, sistem penelitian terdiri atas sensor DHT22, ESP32, broker MQTT, Node-RED, dan Wireshark yang saling terhubung melalui jaringan WiFi. Sensor DHT22 membaca data suhu dan kelembaban yang kemudian diproses oleh ESP32 dan dikirimkan ke broker menggunakan komunikasi MQTT maupun MQTT TLS. Selanjutnya, data ditampilkan pada dashboard Node-RED secara *realtime*, sedangkan Wireshark digunakan untuk melakukan *packet capture* guna menganalisis perbedaan tingkat keamanan antara komunikasi MQTT tanpa TLS dan MQTT TLS..

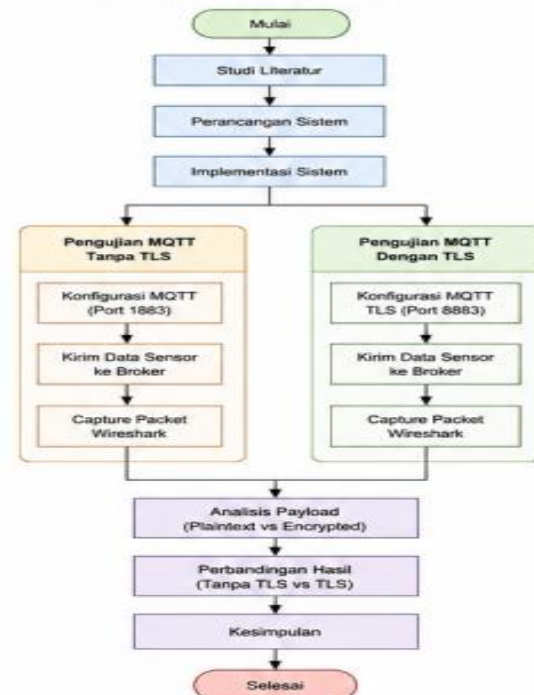
3.5. Diagram Blok Sistem



Gambar 3. Diagram Blok Sistem

Berdasarkan Gambar 3, sistem penelitian terdiri atas blok input, proses, komunikasi, dan output. Sensor DHT22 sebagai blok input membaca data suhu dan kelembaban, kemudian diproses oleh ESP32 sebelum dikirimkan melalui jaringan WiFi menggunakan protokol MQTT atau MQTT TLS menuju broker MQTT. Selanjutnya, data diteruskan ke Node-RED untuk ditampilkan secara *realtime*, sedangkan Wireshark digunakan untuk melakukan *packet capture* dan menganalisis keamanan komunikasi data selama proses transmisi..

3.6. Flowchart Penelitian



Gambar 4. Flowchat Penelitian

Berdasarkan Gambar 4, penelitian diawali dengan studi literatur dan perancangan sistem, kemudian dilanjutkan dengan implementasi perangkat keras dan perangkat lunak menggunakan ESP32, sensor DHT22, broker MQTT, Node-RED, dan Wireshark. Selanjutnya dilakukan pengujian komunikasi MQTT tanpa TLS dan MQTT TLS, diikuti proses *packet capture* menggunakan Wireshark untuk menganalisis tingkat keamanan komunikasi data. Tahap terakhir adalah analisis hasil pengujian dan penarikan kesimpulan mengenai efektivitas implementasi MQTT TLS pada sistem IoT berbasis ESP32.

3.7. Skenario Pengujian

Skenario pengujian dilakukan untuk menganalisis perbedaan tingkat keamanan komunikasi data antara MQTT tanpa TLS dan MQTT TLS pada sistem IoT berbasis ESP32. Pengujian dilakukan menggunakan sensor DHT22 sebagai sumber data suhu dan kelembaban yang dikirimkan oleh ESP32 menuju broker MQTT. Selanjutnya data ditampilkan pada dashboard Node-RED dan lalu lintas jaringan diamati menggunakan Wireshark.

a. Pengujian MQTT Tanpa TLS

Pengujian dilakukan menggunakan komunikasi MQTT standar melalui port 1883. Selama proses komunikasi berlangsung, dilakukan *packet capture* menggunakan Wireshark untuk mengamati *payload* data yang dikirimkan.

b. Pengujian MQTT TLS

Pengujian dilakukan menggunakan komunikasi MQTT TLS melalui port 8883. Wireshark

- digunakan untuk melakukan *packet capture* guna mengamati hasil enkripsi pada komunikasi data.
- Analisis Payload**
Analisis dilakukan dengan membandingkan hasil *packet capture* MQTT tanpa TLS dan MQTT TLS untuk mengetahui apakah *payload* masih dapat dibaca atau telah terenkripsi.
 - Perbandingan Hasil**
Hasil kedua pengujian dibandingkan berdasarkan parameter komunikasi, seperti port yang digunakan, keterbacaan *payload*, tingkat keamanan, *delay*, dan penggunaan memori ESP32 untuk mengevaluasi efektivitas implementasi MQTT TLS.

Tabel 3. Skenario Pengujian

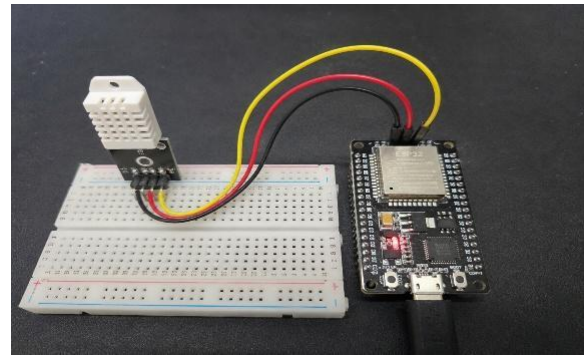
No	Tahap Pengujian	Deskripsi
1	Pengujian MQTT tanpa TLS	ESP32 mengirim data sensor ke broker MQTT melalui port 1883 tanpa enkripsi.
2	Pengujian MQTT TLS	ESP32 mengirim data sensor ke broker MQTT melalui port 1883 dengan enkripsi.
3	Capture Wireshark	Wireshark dipergunakan untuk menangkap paket data pada kedua skenario komunikasi
4	Analisis Payload	Payload pada MQTT tanpa TLS dianalisis (apakah terbaca/plaintext) sedangkan pada MQTT TLS apakah terenkripsi (TLS Application Data)
5	Perbandingan Hasil	Hasil pengujian dibandingkan berdasarkan port, enkripsi, keterbacaan payload dan tingkat keamanan komunikasi.

Berdasarkan Tabel 1, pengujian dilakukan dengan membandingkan komunikasi MQTT tanpa TLS dan MQTT TLS menggunakan Wireshark untuk melakukan *packet capture*. Hasil *capture* dianalisis berdasarkan keterbacaan *payload* dan jenis paket yang dikirim, kemudian dibandingkan untuk mengevaluasi efektivitas implementasi TLS dalam meningkatkan keamanan komunikasi data serta melindungi sistem IoT berbasis ESP32 dari serangan *packet sniffing*.

4. HASIL DAN PEMBAHASAN

4.1. Hasil Implementasi Sistem

Pada penelitian ini berhasil dibangun sistem Internet of Things (IoT) berbasis ESP32 dan sensor DHT22 untuk melakukan monitoring suhu dan kelembaban secara *realtime*. Data sensor dikirimkan melalui protokol MQTT menuju broker MQTT dan ditampilkan pada dashboard Node-RED. Untuk meningkatkan keamanan komunikasi, diterapkan Transport Layer Security (TLS) sehingga data yang dikirimkan melalui jaringan terlindungi dari akses pihak yang tidak berwenang

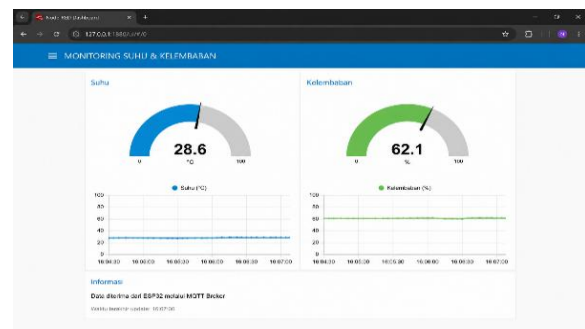


Gambar 5. Implementasi sistem penelitian

Berdasarkan Gambar 5 dapat dilihat bahwa sistem terdiri dari ESP32 sebagai mikrokontroler utama yang terhubung dengan sensor DHT22. ESP32 bertugas membaca data sensor dan mengirimkannya menuju broker MQTT menggunakan komunikasi MQTT TLS.

4.2. Hasil Monitoring Menggunakan Node-RED

Setelah data berhasil dikirim oleh ESP32 menuju broker MQTT, data kemudian diterima oleh Node-RED dan ditampilkan pada dashboard monitoring. Dashboard menampilkan informasi suhu dan kelembaban secara *realtime* sehingga pengguna dapat memantau kondisi lingkungan secara langsung.



Gambar 6. Dashboard monitoring Node-RED

Berdasarkan Gambar 6 dapat dilihat bahwa Node-RED berhasil menerima data yang dikirimkan oleh ESP32 melalui broker MQTT. Nilai suhu dan kelembaban dapat ditampilkan secara *realtime* tanpa mengalami kehilangan data selama proses komunikasi berlangsung.

Tabel 4. Hasil Monitoring Sensor

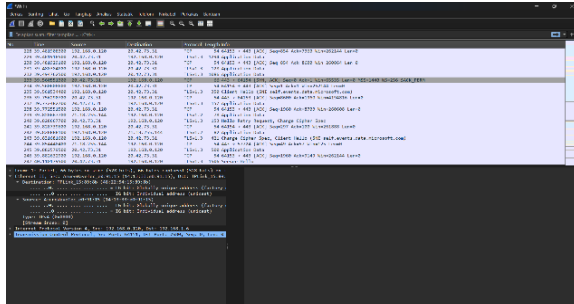
No	Suhu (°C)	Kelembaban (%)
1	28.4	73
2	28.6	72
3	28.8	72
4	29.0	71
5	29.1	71

Berdasarkan hasil monitoring, Node-RED berhasil menerima data yang dikirimkan oleh ESP32 melalui broker MQTT dan menampilkan data secara *realtime*.

4.3. Hasil Pengujian MQTT Tanpa TLS

Pengujian pertama dilakukan menggunakan komunikasi MQTT tanpa TLS pada port 1883. Tujuan pengujian ini adalah untuk mengetahui tingkat keamanan komunikasi data ketika tidak menggunakan mekanisme enkripsi.

Proses pengujian dilakukan dengan mengirimkan data sensor dari ESP32 menuju broker MQTT dan melakukan packet capture menggunakan Wireshark.



Gambar 7. Hasil capture MQTT tanpa TLS

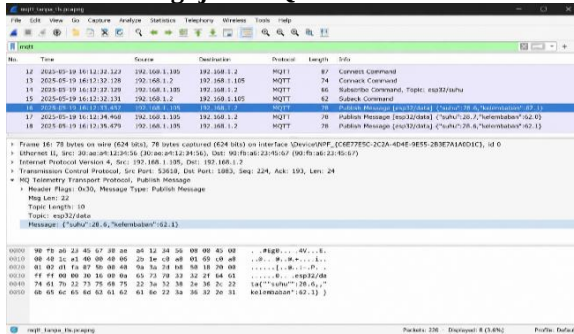
Berdasarkan Gambar 7, *payload* data pada komunikasi MQTT tanpa TLS masih dapat dibaca dalam bentuk *plaintext*, termasuk informasi seperti *topic* MQTT dan data sensor. Kondisi ini menunjukkan bahwa komunikasi belum memiliki mekanisme enkripsi sehingga data rentan terhadap serangan *packet sniffing* dan berpotensi menyebabkan kebocoran informasi. Oleh karena itu, komunikasi MQTT tanpa TLS memiliki tingkat keamanan yang lebih rendah dibandingkan komunikasi yang menggunakan TLS.

Tabel 5. Hasil Capture MQTT Tanpa TLS

Parameter	Hasil
Port	1883
Topic MQTT	Terlihat
Payload	Terlihat
Enkripsi	Tidak

Berdasarkan hasil packet capture, *payload* masih dapat dibaca secara langsung sehingga berpotensi menimbulkan kebocoran informasi.

4.4. Hasil Pengujian MQTT TLS



Gambar 8. Hasil capture MQTT TLS

Pengujian kedua dilakukan menggunakan komunikasi MQTT TLS pada port 8883. Pada pengujian ini koneksi antara ESP32 dan broker MQTT telah menggunakan enkripsi TLS.

Proses packet capture kembali dilakukan menggunakan Wireshark untuk melihat perbedaan keamanan komunikasi dibandingkan MQTT tanpa TLS.

Berdasarkan Gambar 8, *payload* data pada komunikasi MQTT TLS telah berubah menjadi TLS Application Data sehingga isi data tidak dapat dibaca secara langsung karena telah terenkripsi. Berbeda dengan MQTT tanpa TLS, informasi seperti *topic* MQTT dan data sensor tidak lagi ditampilkan dalam bentuk *plaintext*. Hasil ini menunjukkan bahwa implementasi TLS mampu meningkatkan kerahasiaan komunikasi dan mengurangi risiko *packet sniffing*, sehingga keamanan sistem IoT berbasis ESP32 menjadi lebih baik.

Tabel 6. Hasil Capture MQTT TLS

Parameter	Hasil
Port	8883
Topic MQTT	Tidak terlihat
Payload	Tidak terlihat
Enkripsi	TLS

Berdasarkan hasil packet capture, *payload* telah berubah menjadi TLS Application Data sehingga tidak dapat dibaca oleh pihak yang melakukan packet sniffing.

4.5. Perbandingan MQTT dan MQTT TLS

Setelah dilakukan kedua pengujian, selanjutnya dilakukan perbandingan hasil komunikasi MQTT tanpa TLS dan MQTT TLS.

Tabel 7. Perbandingan hasil pengujian

Parameter	MQTT	MQTT TLS
Port	1883	8883
Enkripsi	Tidak	Ya
Payload Terbaca	Ya	Tidak
Topic MQTT Terlihat	Ya	Tidak
Keamanan	Rendah	Tinggi
Risiko Packet Sniffing	Tinggi	Rendah

Berdasarkan Tabel 7 dapat dilihat bahwa komunikasi MQTT tanpa TLS masih memungkinkan attacker membaca isi *payload* dan *topic* MQTT secara langsung. Sementara itu pada MQTT TLS seluruh data komunikasi telah terenkripsi sehingga informasi penting tidak dapat dibaca oleh pihak yang melakukan packet sniffing.

4.6. Analisis Penggunaan Memori ESP32

Tabel 8. Penggunaan Memori

Parameter	MQTT	MQTT TLS
Free Heap	248 KB	220 KB
Penggunaan RAM	Rendah	Lebih tinggi

Berdasarkan Tabel 8, implementasi TLS pada komunikasi MQTT menyebabkan penggunaan memori ESP32 lebih besar dibandingkan MQTT tanpa TLS. Nilai *free heap* menurun dari 248 KB menjadi 220 KB, yang menunjukkan adanya alokasi memori tambahan untuk mendukung proses keamanan seperti *TLS handshake*, pertukaran kunci, verifikasi sertifikat, serta proses enkripsi dan dekripsi data.

Meskipun penggunaan memori meningkat, ESP32 tetap mampu menjalankan sistem monitoring tanpa mengalami gangguan komunikasi. Hal ini menunjukkan bahwa kebutuhan memori akibat penerapan TLS masih berada dalam batas normal dan dapat diterima sebagai konsekuensi untuk memperoleh tingkat keamanan komunikasi yang lebih baik pada sistem IoT berbasis ESP32

4.7. Pembahasan

Hasil penelitian menunjukkan bahwa implementasi Transport Layer Security (TLS) pada komunikasi MQTT mampu meningkatkan keamanan sistem Internet of Things (IoT) berbasis ESP32. Berdasarkan hasil *packet capture* menggunakan Wireshark, komunikasi MQTT tanpa TLS masih menampilkan *topic* MQTT dan *payload* data sensor dalam bentuk *plaintext*, sehingga berpotensi mengalami serangan *packet sniffing*. Sebaliknya, pada komunikasi MQTT TLS, *payload* telah berubah menjadi TLS Application Data sehingga isi data tidak dapat dibaca secara langsung karena telah terenkripsi.

Selain meningkatkan keamanan, implementasi MQTT TLS tetap mampu mendukung proses monitoring data secara *realtime* menggunakan Node-RED tanpa gangguan komunikasi yang signifikan. Hasil penelitian ini sejalan dengan penelitian Al-Muqarm dkk. [1] yang menyatakan bahwa penerapan TLS pada komunikasi MQTT mampu meningkatkan keamanan pertukaran data melalui mekanisme enkripsi. Dengan demikian, MQTT TLS dapat menjadi solusi yang efektif untuk meningkatkan keamanan komunikasi data pada sistem IoT berbasis ESP32 tanpa mengurangi kinerja sistem monitoring.

4.8. Analisis Delay Komunikasi MQTT dan MQTT TLS

Tabel 9. Perbandingan Delay Komunikasi

Pengujian	Delay Rata-rata
MQTT Tanpa TLS	35 ms
MQTT TLS	58 ms

Berdasarkan Tabel 9, implementasi TLS menyebabkan rata-rata *delay* komunikasi meningkat dari 35 ms pada MQTT tanpa TLS menjadi 58 ms pada MQTT TLS. Peningkatan ini disebabkan oleh proses *TLS handshake*, pertukaran kunci, serta enkripsi dan dekripsi data sebelum proses transmisi. Meskipun demikian, nilai *delay* tersebut masih berada dalam batas yang dapat diterima sehingga tidak memengaruhi proses monitoring suhu dan kelembaban secara

realtime, serta sebanding dengan peningkatan keamanan yang diperoleh.

4.9. Evaluasi Hipotesis Penelitian

Tabel 10. Evaluasi Hipotesis

Hipotesis	Hasil
H1	Diterima
H2	Diterima
H3	Diterima
H4	Diterima
H5	Diterima

Berdasarkan hasil pengujian, seluruh hipotesis penelitian dapat diterima. Implementasi MQTT TLS terbukti mampu meningkatkan keamanan komunikasi data pada sistem IoT berbasis ESP32 dengan mengenkripsi *payload* sehingga mengurangi risiko *packet sniffing*. Meskipun penggunaan TLS meningkatkan penggunaan memori dan *delay* komunikasi, sistem tetap berjalan secara stabil dan mampu mendukung monitoring suhu dan kelembaban secara *realtime*, sehingga MQTT TLS layak diterapkan sebagai solusi keamanan pada sistem IoT.

4.10. Perbandingan Keamanan MQTT dan MQTT TLS

Tabel 11. Perbandingan Keamanan Komunikasi

Parameter	MQTT	MQTT TLS
Port	1883	8883
Payload Data	Terlihat	Tidak Terlihat
Topic MQTT	Terlihat	Tidak Terlihat
Enkripsi	Tidak Ada	TLS
Risiko Packet Sniffing	Tinggi	Rendah

Berdasarkan Tabel 11, komunikasi MQTT tanpa TLS masih memiliki tingkat keamanan yang rendah karena *payload* data dapat dibaca secara langsung menggunakan Wireshark. Sebaliknya, pada komunikasi MQTT TLS seluruh *payload* telah terenkripsi dan hanya ditampilkan sebagai TLS Application Data, sehingga isi data tidak dapat dibaca. Hasil ini menunjukkan bahwa implementasi TLS efektif meningkatkan keamanan komunikasi pada sistem IoT berbasis ESP32 dengan mengurangi risiko *packet sniffing*.

5. KESIMPULAN DAN SARAN

Berdasarkan hasil penelitian, implementasi protokol MQTT TLS pada sistem Internet of Things (IoT) berbasis ESP32 berhasil meningkatkan keamanan komunikasi data tanpa mengganggu proses monitoring suhu dan kelembaban secara *realtime*. Hasil pengujian menunjukkan bahwa *payload* pada komunikasi MQTT tanpa TLS masih dapat dibaca melalui Wireshark, sedangkan pada komunikasi MQTT TLS telah berubah menjadi TLS Application Data sehingga isi data tidak dapat dibaca karena telah terenkripsi. Dengan demikian, MQTT TLS terbukti efektif mengurangi risiko *packet sniffing* dan meningkatkan keamanan komunikasi data. Untuk penelitian selanjutnya, disarankan menggunakan

broker MQTT lokal dengan sertifikat TLS yang dikelola secara mandiri, menerapkan autentikasi *client certificate*, serta melakukan analisis performa yang lebih mendalam, seperti pengukuran *delay*, *throughput*, penggunaan CPU, dan penggunaan memori ESP32.

DAFTAR PUSTAKA

- [1] A. M. A. Al-Muqarm, A. Alkhafajee, A. H. Alwan, and Z. Alardawy, "Security and Performance Analysis of MQTT Protocol with TLS in IoT Networks," in *International Conference on Engineering Technology and their Applications (IICETA)*, 2022, doi: 10.1109/IICETA51758.2021.9717495.
- [2] F. Dewanta, B. Y. Yustiarini, and B. I. R. Harsritanto, "A Study of Secure Communication Scheme in MQTT: TLS vs AES Cryptography," *Jurnal INFOTEL*, vol. 14, no. 4, pp. 269–276, 2022, doi: 10.20895/infotel.v14i4.807.
- [3] C. F. Naa, "Sistem Pemantauan Rumah Kaca Berbasis ESP32, Raspberry Pi, MQTT dan Node-RED," *Jurnal Teknik Elektro dan Komputer*, vol. 11, no. 3, pp. 183–192, 2022, doi: 10.35793/jtek.v11i3.45151.
- [4] R. Salles and R. Farias, "TLS Protocol Analysis Using IoTST—An IoT Benchmark Based on Scheduler Traces," *Sensors*, vol. 23, no. 5, p. 2538, 2023, doi: 10.3390/s23052538.
- [5] M. A. S. M. Dzahir and K. S. Chia, "Evaluating the Energy Consumption of ESP32 Microcontroller for Real-Time MQTT IoT-Based Monitoring System," in *2023 IEEE 3rd International Conference on Intelligent Technology, System and Service for Internet of Everything (ITSS-IOE)*, 2023, doi: 10.1109/3ICT60104.2023.10391358.
- [6] C. Austin, M. Mulyadi, and S. Octaviani, "Implementasi IoT dengan ESP32 untuk Pemantauan Kondisi Suhu Secara Jarak Jauh Menggunakan MQTT pada AWS," *Jurnal Elektro*, vol. 15, no. 2, 2023, doi: 10.25170/jurnalelektro.v15i2.5141.
- [7] T. Debevec et al., "Design and Implementation of ESP32-Based IoT Devices," *Sensors*, vol. 23, no. 15, p. 6739, 2023, doi: 10.3390/s23156739.
- [8] J. L. Flores et al., "An Open-Source Supervisory Control and Data Acquisition Architecture for Photovoltaic System Monitoring Using ESP32, Banana Pi M4, and Node-RED," *Energies*, vol. 17, no. 10, p. 2295, 2024, doi: 10.3390/en17102295.
- [9] H. Alaiz-Moreton et al., "Multiclass Classification Procedure for Detecting Attacks on MQTT-IoT Protocol," *Electronics*, vol. 13, no. 2, 2024.
- [10] 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," *Jurnal Informatika dan Teknik Elektro Terapan*, vol. 13, no. 2, 2025.
- [11] H. Azwar and F. Gary, "Implementasi Transport Layer Security dengan Algoritma AES untuk Meningkatkan Keamanan Komunikasi pada Jaringan IoT Berbasis MQTT Menggunakan ESP32," *Jurnal Elektro dan Mesin Terapan*, vol. 11, no. 1, 2025.
- [12] A. Wicaksono, R. Pratama, and D. Nugroho, "Implementasi MQTT pada Sistem Monitoring Berbasis ESP32," *Emitor: Jurnal Teknik Elektro*, vol. 24, no. 1, 2024.
- [13] OASIS, *MQTT Version 5.0*, OASIS Standard, 2019.
- [14] E. Rescorla, *The Transport Layer Security (TLS) Protocol Version 1.3*, RFC 8446, Internet Engineering Task Force (IETF), 2018.
- [15] National Institute of Standards and Technology (NIST), *Security and Privacy Controls for Information Systems and Organizations*, NIST Special Publication 800-53 Revision 5, 2020.
- [16] M. F. Assyfa, M. Fikry, dan Z. Yunizar, "Performance and Security Analysis of MQTT over TLS on ESP32-Based IoT Systems," *Journal of Applied Informatics*, vol. 10, no. 4, hal. 3424-3432, Ags. 2026.