Jurnal ELEMENTER
(Elektro dan Mesin Terapan)
Vol. 11, No. 1, Mei 2025
DOI: 10.35143/elementer.v11i1
P-ISSN: 2443-4167
E-ISSN: 2460-5263

Implementasi Transport Layer Security dengan Algoritma AES untuk Meningkatkan Keamanan Komunikasi pada Jaringan IoT Berbasis MQTT Menggunakan ESP32

Hamid Azwar[1], Felix Gary[2]

[1]Jurusan/Fakultas/Departemen Teknologi Industri, Politeknik Caltex Riau, Pekanbaru 28265, Indonesia
[2]Jurusan/Fakultas/Departemen Teknologi Industri, Politeknik Caltex Riau, Pekanbaru 28265, Indonesia
Corresponding Author: felixgary766@gmail.com

Riwayat Artikel
Diserahkan : 1, Desember, 2024
Diterima : 17, April, 2025
Direvisi : 23 , Mei, 2025
Dipublikasi : 31 Mei 2025

Abstrak

Keamanan data merupakan aspek krusial dalam implementasi Internet of Things (IoT), khususnya pada komunikasi data yang rentan terhadap serangan seperti man-in-the-middle dan penyadapan. Keterbatasan sumber daya pada perangkat IoT menjadi tantangan dalam penerapan protokol keamanan. Penelitian ini bertujuan mengimplementasikan Transport Layer Security (TLS) dengan algoritma Advanced Encryption Standard (AES) pada protokol Message Queuing Telemetry Transport (MQTT) guna meningkatkan keamanan komunikasi data. Metode yang digunakan adalah eksperimen dengan membandingkan kinerja sistem pada perangkat ESP32, baik dengan maupun tanpa penerapan TLS. Parameter yang diuji meliputi penggunaan RAM, penggunaan CPU, dan delay komunikasi. Hasil menunjukkan bahwa penerapan TLS mengurangi sisa RAM rata-rata menjadi 180.256 byte dari total 520 KB, meningkatkan penggunaan CPU menjadi 0,53%, dan meningkatkan delay komunikasi menjadi 0,475 detik, dibandingkan dengan sistem tanpa TLS yang masing-masing mencatatkan 224.825 byte RAM, 0,2% CPU, dan 0,248 detik delay. Peningkatan konsumsi sumber daya tergolong rendah dan masih dalam batas wajar, sehingga dapat disimpulkan bahwa penerapan TLS pada perangkat ESP32 layak dilakukan untuk meningkatkan integritas dan kerahasiaan data dalam sistem IoT tanpa berdampak signifikan terhadap performa.

Kata kunci: IoT, MQTT, AES, TLS, ESP32

Abstract

Data security is a critical aspect in the implementation of the Internet of Things (IoT), especially in communication processes that are vulnerable to attacks such as man-in-the-middle and data eavesdropping. The limited computational resources of IoT devices present challenges in applying standard security protocols. This study aims to implement Transport Layer Security (TLS) using the Advanced Encryption Standard (AES) algorithm on the Message Queuing Telemetry Transport (MQTT) protocol to enhance secure data communication. The method used is experimental by comparing the system performance of an ESP32 device with and without TLS. The parameters observed include RAM usage, CPU usage, and communication delay. The results show that TLS implementation reduces the average available RAM to 180,256 bytes out of 520 KB, increases CPU usage to 0.53%, and increases communication delay to 0.475 seconds. In comparison, without TLS, the system achieves 224,825 bytes RAM, 0.2% CPU usage, and 0.248 seconds delay. The increase in resource consumption is relatively low and within acceptable limits. It is concluded that TLS implementation on ESP32 is feasible for enhancing data confidentiality and integrity in IoT systems without significantly degrading performance.

Keywords: IoT, MQTT, AES, TLS, ESP32

1. Pendahuluan

Keberadaan internet telah menandai evolusi kehidupan global melalui kemunculan Internet of Things (IoT), yang secara signifikan membantu meringankan pekerjaan manusia dan menjadi kebutuhan utama di masa depan. Berdasarkan laporan dari *Exploding Topics*[1], jumlah perangkat IoT diprediksi mencapai 25 triliun pada tahun 2030. IoT merupakan konsep di mana benda atau objek fisik dilengkapi teknologi seperti sensor dan perangkat lunak agar dapat berkomunikasi, bertukar data, dan saling terhubung melalui internet [2]. Dalam sistem IoT, protokol komunikasi menjadi komponen penting. Salah satu protokol yang paling banyak digunakan adalah MQTT (Message Queuing Telemetry Transport), yang dirancang ringan dan efisien untuk perangkat dengan keterbatasan sumber daya [3]. Namun, secara default, MQTT hanya menyediakan autentikasi dasar menggunakan username dan password tanpa enkripsi. Hal ini membuka peluang bagi serangan *man-in-the-middle* (MITM) yang memungkinkan penyerang membaca, mengubah, atau menyisipkan data tanpa terdeteksi [4]. Laporan *Digital Defense Report 2022* oleh Microsoft [5] menunjukkan bahwa meskipun teknologi keamanan perangkat lunak dan keras berkembang pesat, perlindungan pada perangkat IoT masih tertinggal, sehingga kasus serangan terhadap perangkat IoT terus terjadi. Untuk itu, diperlukan mekanisme enkripsi dalam komunikasi data IoT untuk menjamin kerahasiaan dan integritas informasi. Berbagai metode enkripsi seperti Elliptic Curve Cryptography (ECC) dan Advanced Encryption Standard (AES) telah terbukti efektif dalam melindungi data pada layanan cloud [6] maupun dalam skema akses berbasis atribut [7]. Beberapa penelitian sebelumnya [8] [9] telah mengusulkan integrasi TLS/SSL pada protokol MQTT untuk mencegah serangan sniffing dan penyadapan data. Studi lain [10] menunjukkan bahwa AES lebih efisien dibanding RSA pada perangkat IoT karena kebutuhan memorinya yang lebih rendah dan proses enkripsi yang cepat [11]. Namun, sebagian besar penelitian tersebut hanya memfokuskan pada efisiensi algoritma atau simulasi perangkat lunak, tanpa mengevaluasi pengaruh nyata terhadap performa perangkat secara langsung (misalnya RAM, CPU, dan delay) saat digunakan di perangkat embedded seperti ESP32.

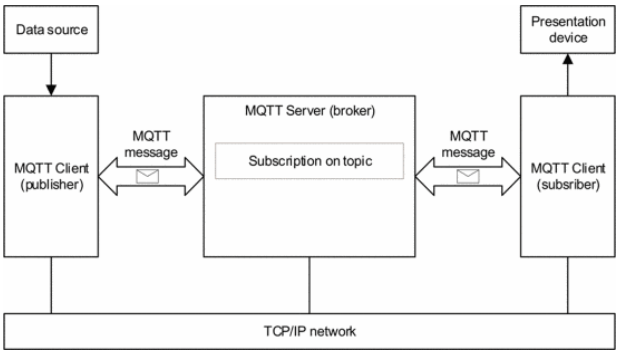
Oleh karena itu, penelitian ini berfokus pada implementasi langsung protokol TLS berbasis AES-256 untuk mengamankan komunikasi antara perangkat ESP32, smartphone, dan MQTT broker. Sistem diuji melalui pengiriman data terenkripsi yang ditransmisikan via WiFi menggunakan protokol MQTT dengan broker yang dijalankan pada Virtual Machine. Selain itu, Wireshark digunakan untuk memantau lalu lintas data dan menganalisis performa jaringan. Kontribusi utama dari penelitian ini adalah memberikan evaluasi eksperimental menyeluruh terhadap dampak implementasi TLS-AES terhadap performa sumber daya perangkat IoT secara real-time dan nyata, yang sebelumnya belum banyak dikaji secara detail.

2. Model Overview

2.1. MQTT

MQTT adalah protokol konektivitas ringan untuk Internet of Things (IoT) yang dirancang untuk komunikasi antar mesin (Machine to Machine/M2M). Dengan jejak kode yang kecil, MQTT sangat cocok untuk diterapkan pada perangkat kecil. Selain itu, protokol ini memenuhi kebutuhan akan konsumsi daya yang rendah, penggunaan bandwidth yang minimal, dan latensi yang rendah [12].

Namun, keunggulan-keunggulan tersebut bukanlah alasan utama penggunaan MQTT dalam arsitektur manajemen jaringan yang diusulkan dalam makalah ini. Karakteristik paling penting dari MQTT dalam konteks arsitektur ini adalah pola komunikasi berbasis publikasi dan langganan (publish and subscribe), yang memungkinkan pengiriman dan penerimaan pesan secara efisien antara entitas komunikasi, seperti perangkat dan aplikasi (Gateway dan Server) [13].



Gambar 1 Arsiterktur MQTT [13]

Dalam arsitektur MQTT, terdapat server yang disebut "broker" sebagai inti dari sistem. Perangkat yang terhubung dengan broker dikenal sebagai "klien". Broker berfungsi sebagai router yang menangani pertukaran data antara klien.

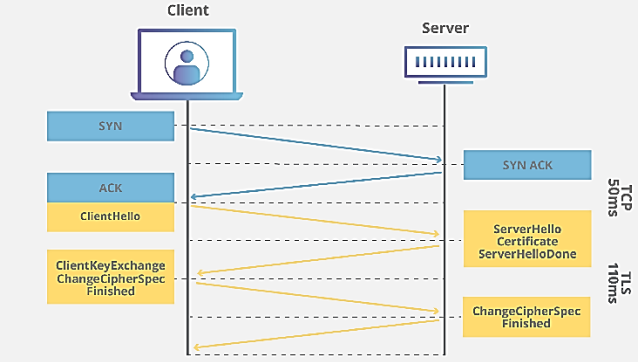
Data yang dipertukarkan antara klien dikelompokkan berdasarkan isi ke dalam kategori tertentu yang disebut "topik". Ketika sebuah klien ingin mendistribusikan data, ia menerbitkan (publish) data tersebut dalam topik tertentu, sehingga klien tersebut berperan sebagai "penerbit" (publisher). Broker kemudian mengirimkan data yang diterima kepada semua klien yang telah menunjukkan minat terhadap topik tersebut melalui proses berlangganan (subscription). Klien yang menerima data topik disebut sebagai "pelanggan" (subscriber) untuk topik tersebut. Klien yang bertindak sebagai publisher untuk satu topik juga dapat bertindak sebagai subscriber untuk topik lain.

2.2. Transport Layer Security (TLS)

TLS merupakan sebuah keamanan komunikasi antara *client* dan server yang di mana memberikan enkripsi symmetric, asymmetric, dan hash yang menyediakan *integrity*, *confidentiality*, dan *authentication* [14]. Protokol TLS memiliki 2 aspek yaitu *handshake protocol* dan *record protocol*. *Handshake protocol* menetapkan komunikasi pada perangkat menggunakan kriptografi *asymmetric* (*public key*) untuk mengamankan komunikasi tersebut.

Record protocol bertujuan untuk memverifikasi *integrity* dan *confidentiality* pada data.

Dalam penerapan TLS perlunya sebuah sertifikat yang digunakan untuk memvalidasi identitas server. Sertifikat ini dikeluarkan oleh *ceritiface authority* (CA) yang diberikan kepada perorangan ataupun kepada bisnis yang memiliki domain. Sertifkat ini berisi informasi pemilik domain berserta *public key*.



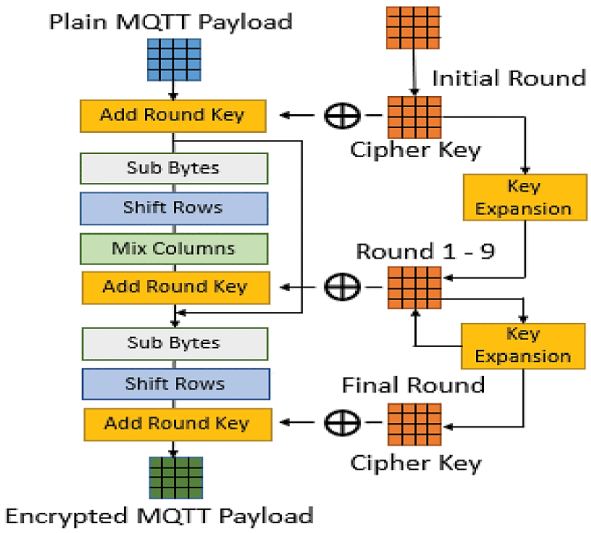
Gambar 2 TLS Handshake [15]

TLS *handshake* merupakan serangkaian datagram atau pesan yang saling bertukar antara klien dan server. Jumlah tahap yang terjadi pada TLS bervariasi berdasarkan algoritma yang digunakan dan *cipher suites* yang didukung kedua belah pihak. Sebuah TLS *handshake* merupakan sebuah proses yang memulai sesi komunikasi menggunakan TLS. Selama proses *handshake*, kedua pihak akan bertukar pesan mengakui satu sama lain, menverifikasi satu sama lain, menetapkan algoritma kriptografi yang akan digunakan, dan setuju dengan session key (kunci yang dipakai untuk validasi) [15].

2.3. Advanced Encryption Standard (AES)

Advanced Encryption Standard (AES) adalah algoritma yang digunakan untuk mengenkripsi data digital. Algoritma ini awalnya dikenal dengan nama Rijndael dan telah diadopsi secara luas di seluruh dunia. Secara teori, membobol kode yang dienkripsi dengan AES hampir tidak mungkin dilakukan karena kombinasi kunci yang sangat besar. AES menggunakan algoritma kunci simetris, yang berarti proses enkripsi dan dekripsi menggunakan kunci yang sama.

AES bekerja berdasarkan prinsip jaringan substitusi-permutasi, yang membuatnya cepat baik pada platform perangkat lunak maupun perangkat keras. AES menggunakan tiga ukuran kunci, yaitu 128-bit, 192-bit, dan 256-bit. Setiap ukuran kunci memengaruhi perilaku algoritma secara berbeda. Ukuran kunci yang lebih besar tidak hanya menyediakan lebih banyak bit untuk mengacak data tetapi juga meningkatkan kompleksitas algoritma cipher. Kombinasi substitusi, permutasi, dan penggunaan ukuran blok yang tetap menjadikan AES sangat efisien sekaligus aman.



Gambar 3 Arsitektur AES [16]

Algoritma ini terdiri dari sejumlah putaran berulang dari siklus konversi identik yang mengubah plaintext (teks asli) menjadi ciphertext (teks terenkripsi). Setiap putaran terdiri dari empat langkah, kecuali pada putaran terakhir, seperti yang ditunjukkan pada Gambar 3. Langkah-langkah tersebut adalah AddRoundKey, SubBytes, ShiftRows, dan MixColumns. Siklus ini diterapkan dalam urutan terbalik untuk mendekripsi ciphertext. Jumlah iterasi siklus adalah 10, 12, dan 14 untuk panjang kunci masing-masing 128-bit, 192-bit, dan 256-bit [17]. Dengan semua fitur ini, AES (Advanced Encryption Standard) merupakan algoritma enkripsi terkuat saat ini. Jika panjang kunci yang dipilih adalah 128 bit, terdapat 2^{128} kombinasi yang mungkin. Angka ini, yang terdiri dari sekitar 39 digit, membutuhkan waktu sekitar 10 tahun untuk dipecahkan dengan komputer masa kini. Keamanan sistem semakin ditingkatkan dengan mengganti kata sandi secara berkala.

Algoritma Advanced Encryption Standard (AES) dirancang dengan mempertimbangkan kecepatan tinggi dan kebutuhan memori (RAM) yang rendah sebagai kriteria dalam proses pemilihannya. Oleh karena itu, AES dapat bekerja dengan baik pada berbagai jenis perangkat keras, mulai dari kartu pintar 8-bit hingga komputer berkinerja tinggi. Karena dirancang untuk efisiensi pada perangkat keras dengan sumber daya terbatas, AES menjadi pilihan ideal untuk berbagai aplikasi, termasuk pada perangkat kecil seperti kartu pintar hingga sistem besar dengan prosesor modern yang mendukung akselerasi perangkat keras. Akselerasi perangkat keras melalui AES-NI meningkatkan kinerja secara signifikan, memungkinkan proses enkripsi dan dekripsi dilakukan dengan cepat tanpa mengorbankan tingkat keamanan [18].

2.4. Tahapan Sistem Serangan pada Protokol MQTT

Sistem serangan terhadap protokol MQTT melibatkan berbagai teknik yang digunakan oleh penyerang untuk mengkompromikan protokol ini dan mengganggu operasi normalnya. Serangan-serangan ini bertujuan untuk memanipulasi pesan MQTT dan merusak integritas model publish-subscribe [19]. Sistem serangan ini dapat dibagi menjadi tiga tahap utama:

a) Tahap Serangan Koneksi

Pada tahap awal ini, node berbahaya menargetkan koneksi dalam jaringan MQTT. Penyerang dapat melihat pesan MQTT yang dikirimkan dalam jaringan dan memanipulasi konten pesan sebelum meneruskannya ke penerima yang dituju. Manipulasi ini dapat menipu penerima atau memicu tindakan yang tidak diinginkan berdasarkan data yang telah dimanipulasi.

b) Tahap Serangan Autentikasi

Setelah tahap koneksi, penyerang beralih untuk merusak mekanisme autentikasi yang diterapkan dalam jaringan. Metode yang digunakan yaitu mencoba melewati protokol autentikasi dan mengeksploitasi kerentanan pada sistem autentikasi untuk mendapatkan akses tidak sah ke jaringan MQTT.

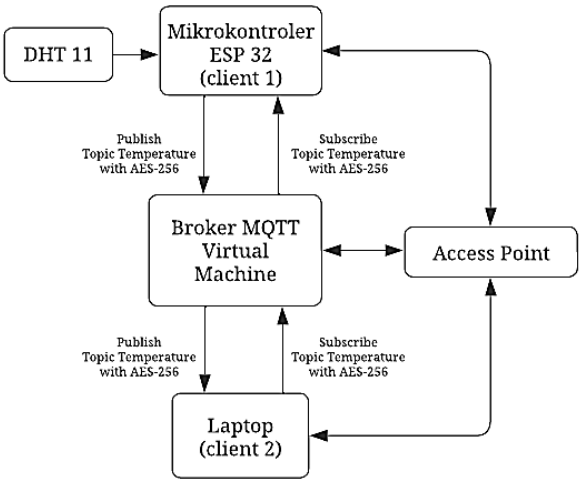
c) Tahap Serangan Komunikasi

Pada tahap terakhir, penyerang berusaha mengganggu saluran komunikasi dalam jaringan MQTT. Teknik yang umum digunakan yaitu Replay Attack diman penyerang menangkap pesan MQTT dan kemudian mengirim ulang pesan tersebut ke broker atau penerima yang dituju. Akibatnya, tindakan tertentu dapat terjadi secara duplikasi, atau informasi sensitif dapat digunakan untuk mendapatkan akses tidak sah atau memicu tindakan yang tidak diinginkan.

Ketiga tahapan ini menunjukkan bagaimana serangan dapat dilakukan secara sistematis untuk merusak protokol MQTT. Dengan memahami setiap fase serangan, langkah-langkah pencegahan dan mitigasi dapat dirancang untuk melindungi jaringan MQTT dari ancaman keamanan ini.

3. Perancangan Sistem

Adapun perancangan system pada penelitian dapat dilihat pada gambar 4 berikut ini.



Gambar 4 Diagram Sistem Kerja Perangkat IoT

Pada gambar di atas, Client 1, yaitu ESP32, berfungsi sebagai publisher, sedangkan Client 2, yaitu laptop, berperan sebagai subscriber. Broker adalah sebuah mesin virtual yang telah diinstal dengan protokol MQTT menggunakan Mosquitto. Broker memiliki peran penting sebagai perantara dalam menerima dan meneruskan pesan antara publisher dan subscriber berdasarkan topik yang ditentukan.

Saat sistem berjalan, Client 2 mengirimkan pesan subscribe dengan topik tertentu kepada broker. Broker kemudian memproses dan meneruskan permintaan subscribe tersebut kepada Client 1, yakni ESP32 yang terhubung dengan sensor suhu DHT11. Setelah menerima permintaan tersebut, Client 2 mengirimkan balasan berupa pesan publish dengan topik yang sama kepada broker. Broker selanjutnya meneruskan pesan publish tersebut kepada Client 1.

Proses ini menggambarkan pola komunikasi publish–subscribe pada protokol MQTT, di mana broker bertindak sebagai perantara yang memastikan setiap pesan diteruskan sesuai dengan topik yang telah ditentukan.

4. Hasil dan Pembahasan

4.1. Pengujian Keamanan Tanpa Implementasi Transport Layer Security

Pengujian dilakukan dengan melakukan sniffing pada lalu lintas komunikasi antara client ESP32, client smartphone, dan broker MQTT. Dengan menggunakan wireshark, penyerang dapat melihat isi pesan subscribe dan publish yang dikirimkan oleh client ESP32 (192.168.1.8), client Smartphone (192.168.1.5) dan broker (192.168.1.5).

Source	Destination	Protocol	Length	Info
192.168.1.8	192.168.1.7	MQTT	99	Connect Command
192.168.1.7	192.168.1.8	MQTT	58	Connect Ack
192.168.1.8	192.168.1.7	MQTT	95	Publish Message

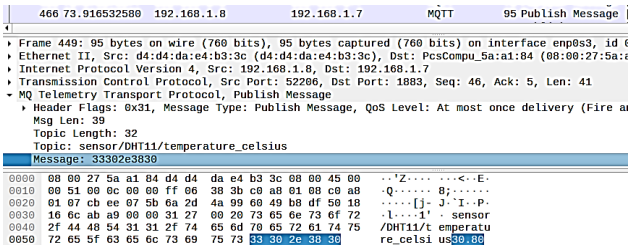
Gambar 5 Komunikasi antara client dan broker tanpa TLS

Komunikasi antara client ESP32 (192.168.1.8) dan broker (192.168.1.7) diawali dengan pesan connect command dari client dan kemudian diterima oleh broker dengan pesan Connect Ack.

```
▸ Frame 416: 99 bytes on wire (792 bits), 99 bytes captured (792 bi
▸ Ethernet II, Src: d4:d4:da:e4:b3:3c (d4:d4:da:e4:b3:3c), Dst: Pcs
▸ Internet Protocol Version 4, Src: 192.168.1.8, Dst: 192.168.1.7
▸ Transmission Control Protocol, Src Port: 52206, Dst Port: 1883, S
▾ MQ Telemetry Transport Protocol, Connect Command
  ▸ Header Flags: 0x10, Message Type: Connect Command
    Msg Len: 43
    Protocol Name Length: 4
    Protocol Name: MQTT
    Version: MQTT v3.1.1 (4)
  ▸ Connect Flags: 0xc2, User Name Flag, Password Flag, QoS Level:
    Keep Alive: 15
    Client ID Length: 11
    Client ID: ESP32Client
    User Name Length: 11
    User Name: testproject
    Password Length: 5
    Password: 12345
```

Gambar 6 Proses autentikasi tanpa TLS

Ketika melakukan pengujian, terdapat informasi mengenai akun user (testproject) dan password (12345) MQTT pada saat proses autentikasi untuk terhubung dengan broker MQTT.

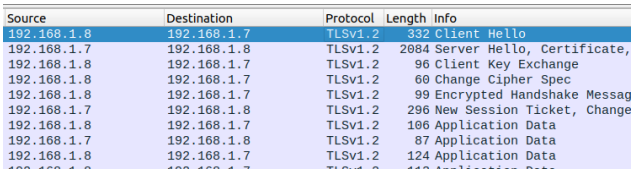


Gambar 7 Data sensor yang dikirimkan tanpa TLS

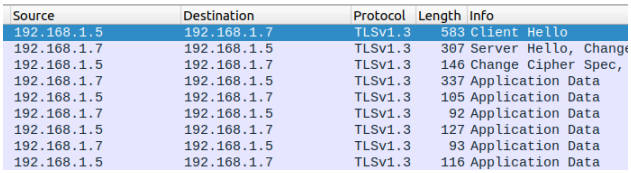
Pada gambar 7 dapat terlihat proses publish ke broker dengan topic "sensor/DHT11/temperature_celcius" dari client ESP32 (192.168.1.8) menuju broker (192.168.1.7) dengan data suhu "30.80". Hal ini dapat dikatakan penggunaan protokol MQTT secara default tidak memberikan keamanan data melainkan hanya memberikan keamanan autentikasi user.

4.2. Pengujian Keamanan dengan Implementasi Transport Layer Security

Komunikasi antara client dan broker yang sudah diimplementasikan TLS diawali dengan proses handshake untuk menverifikasi satu sama lain, menetapkan algoritma kriptografi yang akan digunakan. Gambar 8 dan 9 menunjukan proses komunikasi antara client ESP32 (192.168.1.8) dengan broker (192.168.1.7) dan client smartphone (192.168.1.5) dengan broker (192.168.1.7).

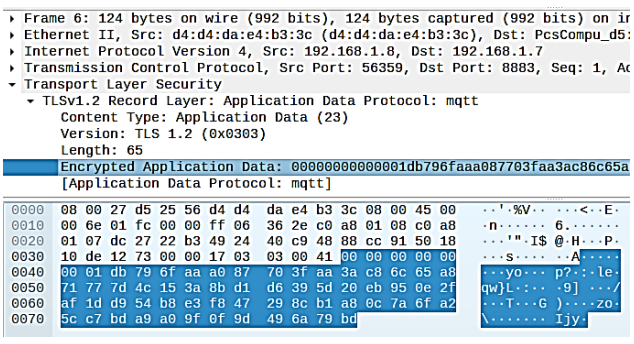


Gambar 8 Komunikasi client smartphone dengan broker menggunakan TLS



Gambar 9 Komunikasi ESP32 dengan broker menggunakan TLS

Pada gambar 10 dapat terlihat proses publish ke broker dengan topic "sensor/DHT11/temperature_celcius" dari client ESP32 (192.168.1.8) menuju broker (192.168.1.7) dengan data yang dikirimkan dari client ESP32 menuju broker hanya terlihat karakter acak, hal ini menandakan enkripsi data yang diimplementasikan berhasil.

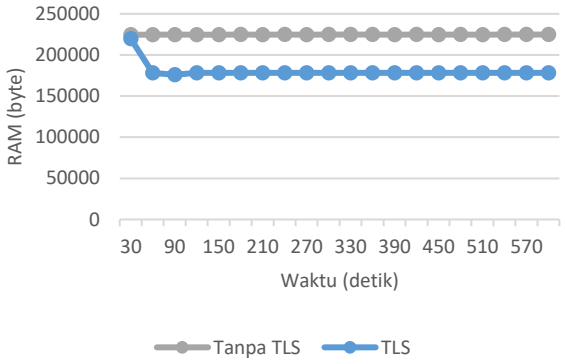


Gambar 10 Data sensor dengan menerapkan TLS

Parameter pengujian penelitian ini adalah RAM, delay, CPU yang digunakan selama proses komunikasi client-broker dengan membandingkan kinerja sebelum dan sesudah implementasi TLS (Transport Layer Security).

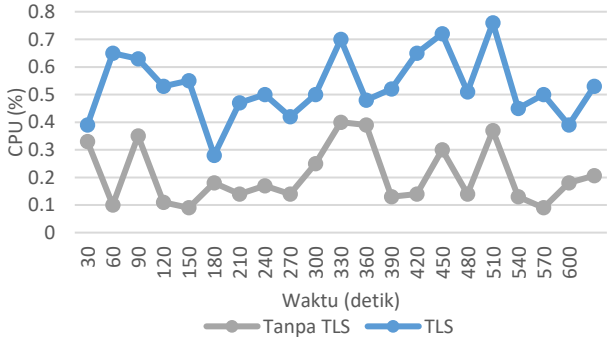
4.3. Pengujian Penggunaan RAM pada ESP32

Pengujian kinerja pada perangkat IoT dilakukan dengan membandingkan perbedaan penggunaan *RAM* serta *delay* sebelum dan sesudah implementasi TLS *(Transport Layer Security)*terhadap mikrotkontroler ESP32.



Gambar 11 Sisa penggunaan RAM pada ESP32

Hasil pengujian kinerja perangkat IoT berdasarkan penggunaan RAM yang diuji selama 10 menit dengan pengambilan data per 30 detik. Pada grafik Gambar 11 garis abu-abu menunjukkan sisa penggunaan RAM pada ESP32 tanpa adanya implementasi TLS dengan algoritma AES sedangkan garis biru menunjukkan sisa penggunaan RAM pada ESP32 dengan implementasi TLS dengan algoritma AES. Pada 30 detik pertama penggunaan RAM dengan algoritma AES (219884 bytes) dan tanpa algorima AES (224800 bytes) hampir sama yang dikarenakan pada proses awal komunnikasi masih melakukan proses validasi antara client dan broker yang di mana tidak banyak menggunaakan resources dari ESP32 sebelum melanjutkan komunikasi pengiriman data. Secara keseluruhan data, nilai rata-rata sisa penggunaan RAM pada ESP32 berkisar 180256 bytes untuk penerapan TLS dan 224825 bytes untuk tanpa penerapan TLS.



Gambar 12 Penggunaan CPU pada ESP32

4.4. Pengujian Penggunaan CPU pada ESP32

Pengujian kinerja pada perangkat IoT dilakukan dengan membandingkan perbedaan penggunaan CPU sebelum dan

sesudah implementasi TLS terhadap mikrotkontroler ESP32.

Hasil pengujian kinerja perangkat *IoT* berdasarkan penggunaan CPU yang diuji selama 10 menit dengan pengambilan data per 30 detik. Pada grafik Gambar 12 garis abu-abu menunjukkan penggunaan CPU pada ESP32 tanpa adanya implementasi TLS sedangkan garis biru menunjukkan penggunaan CPU pada ESP32 dengan implementasi TLS. Kinerja CPU yang menggunakan TLS lebih tinggi dibandingkan yang tidak menggunakan TLS, hal ini dikarenakan penggunaan TLS memiliki banyak proses untuk bisa mengamankan data sebelum dikirimkan kepada *broker*. Berbeda dengan yang tidak menggunakan TLS yang di mana hanya mengirimkan data langsung menuju *broker* tanpa adanya pengamanan data. Secara keseluruhan data, nilai rata-rata penggunaan CPU yang menggunakan TLS adalah 0,53% dan penggunaan CPU yang tidak menggunakan adalah TLS 0,2%.

4.5. Pengujian Delay Komunikasi pada Client dan Broker

Pengujian selanjutkan menghitung nilai delay yang dihasilkan sebelum dan sesudah implementasi TLS.

Tabel 1 Delay komunikasi MQTT

	Packets	Time Span (s)	Delay (s)
AES	285	600,381	0,475
Tanpa AES	145	585,761	0,248

Pada Tabel 1 terdapat jumlah packets dan time span yang didapatkan dari waktu pengambilan data selama 10 menit sebelum dan sesudah implementasi AES. Pengujian delay yang dihasil pada saat implementasi algoritma AES didapatkan sebesar 0,475 detik sedangkan tanpa algoritma AES sebesar 0,248 detik. Dari data tersebut dapat dilihat bahwasannya dengan implementasi AES dapat meningkatkan delay sebanyak 2 kali lipat dari penggunaan tanpa algoritma AES.

5. Kesimpulan

Berdasarkan hasil pengujian, implementasi Transport Layer Security (TLS) dengan algoritma AES-256 pada protokol MQTT terbukti mampu meningkatkan keamanan komunikasi data, khususnya dari segi integritas dan kerahasiaan, baik pada sisi klien maupun broker. Hasil evaluasi menunjukkan bahwa dampak implementasi TLS terhadap performa perangkat ESP32 relatif kecil dan masih dalam batas wajar untuk sistem IoT dengan keterbatasan sumber daya. Pada saat TLS diterapkan, perangkat mencatat rata-rata sisa RAM sebesar 180.256 byte, penggunaan CPU sebesar 0,53%, dan delay komunikasi sebesar 0,475 detik. Sementara itu, tanpa TLS, sisa RAM tercatat 224.825 byte, penggunaan CPU 0,2%, dan delay 0,248 detik. Perbedaan ini menunjukkan bahwa protokol TLS dapat diterapkan tanpa menyebabkan penurunan performa signifikan. Secara praktis, temuan ini mendukung penerapan sistem IoT yang lebih aman pada skala smart home, industri ringan, atau sistem pemantauan lingkungan, tanpa perlu perangkat dengan spesifikasi tinggi. Untuk pengembangan ke depan, penelitian ini dapat diperluas dengan menguji implementasi pada beragam mikrokontroler atau perangkat IoT dengan arsitektur berbeda, serta mempertimbangkan variasi skenario jaringan seperti kondisi koneksi yang tidak stabil, enkripsi end-to-end penuh, atau integrasi dengan sistem autentikasi lanjutan seperti OAuth atau sertifikat digital.

Ucapan Terima Kasih

Penulis ingin menyampaikan rasa terima kasih kepada Jurusan Teknologi Industri Politeknik Caltex Riau atas dukungan penuh terhadap penelitian ini.

Daftar Pustaka

[1] "60+ Amazing IoT Statistics (2024-2030)," Exploding Topics. Accessed: Nov. 30, 2024. [Online]. Available: https://explodingtopics.com/blog/iot-stats

[2] R. Setiawan, "Memahami Apa Itu Internet of Things," Dicoding Blog. Accessed: Nov. 30, 2024. [Online]. Available: https://www.dicoding.com/blog/apa-itu-internet-of-things/

[3] M. Diono, H. Azwar, and W. Khabzli, "Sistem Monitoring Jaringan Sensor Node Berbasis Protokol MQTT," *J. Elektro Dan Mesin Terap.*, vol. 7, no. 2, pp. 120–126, Nov. 2021, doi: 10.35143/elementer.v7i2.5232.

[4] N. S. Alotaibi, H. I. Sayed Ahmed, S. O. M. Kamel, and G. F. ElKabbany, "Secure Enhancement for MQTT Protocol Using Distributed Machine Learning Framework," *Sensors*, vol. 24, no. 5, p. 1638, Mar. 2024, doi: 10.3390/s24051638.

[5] E. MacBride, "The dark web's criminal minds see Internet of Things as next big hacking prize," CNBC. Accessed: Nov. 30, 2024. [Online]. Available: https://www.cnbc.com/2023/01/09/the-dark-webs-criminal-minds-see-iot-as-the-next-big-hacking-prize.html

[6] M. Sivajyothi and Devi. T, "Analysis of Elliptic Curve Cryptography with AES for Protecting Data in Cloud with improved Time efficiency," in *2022 2nd International Conference on Innovative Practices in Technology and Management (ICIPTM)*, Gautam Buddha Nagar, India: IEEE, Feb. 2022, pp. 573–577. doi: 10.1109/ICIPTM54933.2022.9753926.

[7] D. Han, N. Pan, and K.-C. Li, "A Traceable and Revocable Ciphertext-Policy Attribute-based Encryption Scheme Based on Privacy Protection," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 1, pp. 316–327, Jan. 2022, doi: 10.1109/TDSC.2020.2977646.

[8] National Advanced IPv6 Centre (NAv6), Universiti Sains Malaysia, Penang, Malaysia, A. J. Hintaw, S. Manickam, S. Karuppayah, and M. F. Aboalmaaly, "A Brief Review on MQTT's Security Issues within the Internet of Things (IoT)," *J. Commun.*, pp. 463–469, 2019, doi: 10.12720/jcm.14.6.463-469.

[9] 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 *2021 4th International Iraqi Conference*

on Engineering Technology and Their Applications (IICETA), Najaf, Iraq: IEEE, Sep. 2021, pp. 206–211. doi: 10.1109/IICETA51758.2021.9717495.

[10] A. Rachmayanti and W. Wirawan, "Implementasi Algoritma Advanced Encryption Standard (AES) pada Jaringan Internet of Things (IoT) untuk Mendukung Smart Healthcare," *J. Tek. ITS*, vol. 11, no. 3, pp. A217–A222, Dec. 2022, doi: 10.12962/j23373539.v11i3.97042.

[11] S. A. Sholikhatin, A. P. Kuncoro, A. L. Munawaroh, and G. A. Setiawan, "Comparative Study of RSA Asymmetric Algorithm and AES Algorithm for Data Security," *Edu Komputika J.*, vol. 9, no. 1, pp. 60–67, Jun. 2022, doi: 10.15294/edukomputika.v9i1.57389.

[12] O. Jukic, I. Hedi, and E. Cirikovic, "IoT cloud-based services in network management solutions," in *2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO)*, Opatija, Croatia: IEEE, Sep. 2020, pp. 419–424. doi: 10.23919/MIPRO48935.2020.9245117.

[13] O. Jukic, R. Filjar, I. Hedi, and E. Cirikovic, "MQTT-like Network Management Architecture," in *2021 44th International Convention on Information, Communication and Electronic Technology (MIPRO)*, Opatija, Croatia: IEEE, Sep. 2021, pp. 459–463. doi: 10.23919/MIPRO52101.2021.9596894.

[14] R. R. Pahlevi, P. Sukarno, and B. Erfianto, "Implementation of Event-Based Dynamic Authentication on MQTT Protocol," *J. Rekayasa Elektr.*, vol. 15, no. 2, Sep. 2019, doi: 10.17529/jre.v15i2.13963.

[15] "What happens in a TLS handshake? | SSL handshake." Accessed: Nov. 30, 2024. [Online]. Available: https://www.cloudflare.com/learning/ssl/what-happens-in-a-tls-handshake/

[16] Ü. Güler and R. Sokullu, "Remote Monitoring and Control of Refrigerators Through MQTT Protocol with AES Encryption," in *2023 International Balkan Conference on Communications and Networking (BalkanCom)*, İstanbul, Turkiye: IEEE, Jun. 2023, pp. 1–6. doi: 10.1109/BalkanCom58402.2023.10167993.

[17] C. Paar, J. Pelzl, and B. Preneel, *Understanding cryptography: a textbook for students and practitioners*, 2nd Corrected printing. Berlin Heidelberg: Springer, 2010.

[18] R. Murugesh, "Advanced biometric ATM machine with AES 256 and steganography implementation," in *2012 Fourth International Conference on Advanced Computing (ICoAC)*, Chennai, India: IEEE, Dec. 2012, pp. 1–4. doi: 10.1109/ICoAC.2012.6416799.

[19] A. J. Hintaw, S. Manickam, M. F. Aboalmaaly, and S. Karuppayah, "MQTT Vulnerabilities, Attack Vectors and Solutions in the Internet of Things (IoT)," *IETE J. Res.*, vol. 69, no. 6, pp. 3368–3397, Aug. 2023, doi: 10.1080/03772063.2021.1912651.