BAB 2
TINJAUAN PUSTAKA

2.1 Penelitian dahulu

Penelitian (Sicari et al., 2015) menunjukkan bahwa *Internet of Things* (IoT) telah mengubah cara kita berinteraksi dengan dunia di sekitar kita, menciptakan jaringan perangkat yang saling terhubung dan berkomunikasi. Namun, seiring dengan pertumbuhan pesat teknologi ini, keamanan jaringan IoT menjadi perhatian utama. Menurut dokumentasi (Michon et al., 2020), perangkat ESP32, sebagai salah satu mikrokontroler yang populer dalam pengembangan IoT, memerlukan pendekatan keamanan yang kuat untuk melindungi data sensitif dan mencegah akses tidak sah. (Frustaci et al., 2018) menekankan pentingnya keamanan dalam ekosistem IoT yang semakin kompleks, menjadi dasar penelitian ini yang bertujuan untuk menganalisis dan mengembangkan solusi keamanan jaringan IoT berbasis ESP32 dengan memanfaatkan protokol HTTP dan enkripsi TLS. Penelitian sebelumnya mengenai evaluasi live streaming menggunakan Dynamic Adaptive Streaming over HTTP (DASH) pada jaringan LAN menunjukkan pentingnya protokol HTTP dalam transmisi data yang stabil dan efisien (Moera, 2016).

(Prayama et al., 2021) menjelaskan bahwa protokol *HyperText Transfer Protocol* (HTTP) telah lama menjadi tulang punggung komunikasi *web*, namun penggunaannya dalam konteks IoT memerlukan pertimbangan khusus. (Kumar et al., 2016) mengungkapkan bahwa meskipun HTTP menawarkan fleksibilitas dan kompatibilitas yang luas, ia juga membawa risiko keamanan jika tidak diimplementasikan dengan benar. Oleh karena itu, penelitian ini akan menggali cara mengoptimalkan penggunaan HTTP dalam lingkungan IoT berbasis ESP32, dengan fokus pada aspek keamanan dan efisiensi. Studi (Naik, 2017)menunjukkan bahwa optimalisasi protokol HTTP dapat secara signifikan meningkatkan performa dan keamanan perangkat IoT dengan sumber daya terbatas.

(Rescorla, 2018) menegaskan bahwa enkripsi *Transport Layer Security* (TLS) merupakan komponen kunci dalam menjaga kerahasiaan dan integritas data yang ditransmisikan melalui jaringan. (Restuccia et al., 2020) mengidentifikasi bahwa penerapan TLS pada perangkat dengan sumber daya terbatas seperti ESP32

menghadirkan tantangan tersendiri, terutama dalam hal kinerja dan konsumsi daya. Penelitian ini akan mengeksplorasi metode implementasi enkripsi TLS yang efektif pada ESP32, dengan mempertimbangkan batasan hardware dan kebutuhan aplikasi IoT. Studi lanjutan oleh (Moreira et al., 2018) telah mendemonstrasikan bahwa optimisasi algoritma TLS dapat mengurangi overhead komputasi pada perangkat mikrokontroler secara signifikan.

Tabel 2. 1 Penelitian Terdahulu

Penelitian	**Judul**	**Variabel yang Diteliti**	**Metode yang Digunakan**	**Kekurangan**
Sicari, S., et al. (2015)	Security, privacy and trust in Internet of Things: The road ahead	Keamanan, privasi, dan kepercayaan di IoT	Analisis literatur	Tidak memberikan solusi teknis konkret terkait tantangan keamanan.
Moera, (2016)	Evaluasi Live Streaming Menggunakan Dynamic Adaptive Streaming Over Http (Dash) Pada Jaringan Lan	Throughput, Delay dan Bitrate Video	Metode evaluasi kinerja	Penelitian ini terbatas pada 20 *client* di jaringan LAN, tanpa mencakup jaringan yang lebih luas atau analisis kualitas subjektif. Penjelasan metodologi juga kurang rinci untuk memfasilitasi replikasi studi.
Michon et al (2021)	ESP32 Series Datasheet	ESP32 dan spesifikasinya	Studi teknis	Hanya membahas aspek teknis hardware, tidak mencakup penerapan keamanan.
Frustaci et al. (2018)	Evaluating critical security issues of the IoT	Isu keamanan kritis pada IoT	Evaluasi kritis	Terbatas pada evaluasi teoretis tanpa uji coba implementasi

	world: Present and future challenges			nyata.
Prayaman et al (2014)	*HyperText Transfer Protocol* (HTTP/1.1) : Message Syntax and Routing	Protokol HTTP/1.1	Studi protokol komunikasi	Tidak mencakup aspek keamanan pada protokol HTTP secara spesifik.
Kumar et al (2016)	Security in Internet of Things: Challenges, Solutions and Future Directions	Tantangan dan solusi keamanan di IoT	Survei literatur	Kurang fokus pada solusi implementatif untuk perangkat IoT dengan sumber daya rendah.
naik et al. (2017)	Lightweight and secure protocol for HTTP-based IoT applications	Protokol HTTP ringan dan aman untuk IoT	Pengembangan protokol	Tidak menyebutkan detail implementasi enkripsi atau pengujian pada perangkat.
Rescorla, E. (2018)	The *Transport Layer Security* (TLS) Protocol Version 1.3	Protokol TLS versi 1.3	Studi protokol	Fokus pada deskripsi protokol TLS tanpa mencakup implementasi pada IoT.
Restuccia et al. (2020)	Lightweight and secure TLS implementations for Internet of Things	Implementasi TLS ringan pada IoT	Implementasi protokol	Mengabaikan tantangan konsumsi daya perangkat selama implementasi TLS.
Moreira, N., et al. (2018)	CryptoGC: Efficient Cryptography for Constrained Devices with Garbage Collection	Kriptografi efisien untuk perangkat terbatas	Pengembangan algoritma	Tidak mengeksplorasi kompatibilitas dengan berbagai perangkat IoT.

2.2 Landasan Teori

2.2.1 Internet of Things (IoT)

Internet of Things (IoT) dapat didefinisikan sebagai interkoneksi objek-objek cerdas yang memungkinkan komunikasi antar perangkat dan dengan pengguna. IoT merujuk pada jaringan perangkat fisik yang terhubung dan bertukar data melalui internet, membawa revolusi dalam berbagai sektor seperti industri, kesehatan, dan *smart home*(Selay et al., 2022). IoT mengintegrasikan berbagai teknologi dan protokol komunikasi untuk menciptakan ekosistem perangkat yang saling terhubung dan dapat berinteraksi secara otomatis.

Meskipun IoT menawarkan banyak manfaat, teknologi ini juga membawa tantangan keamanan yang signifikan. Masalah privasi, kerahasiaan data, dan integritas sistem menjadi perhatian utama seiring dengan meningkatnya jumlah perangkat yang terhubung. Beberapa area kritis dalam keamanan IoT meliputi otentikasi perangkat, manajemen identitas, dan kontrol akses. Oleh karena itu, pengembangan solusi keamanan yang kuat dan dapat diandalkan menjadi prioritas dalam evolusi teknologi IoT.

2.2.2 ESP32

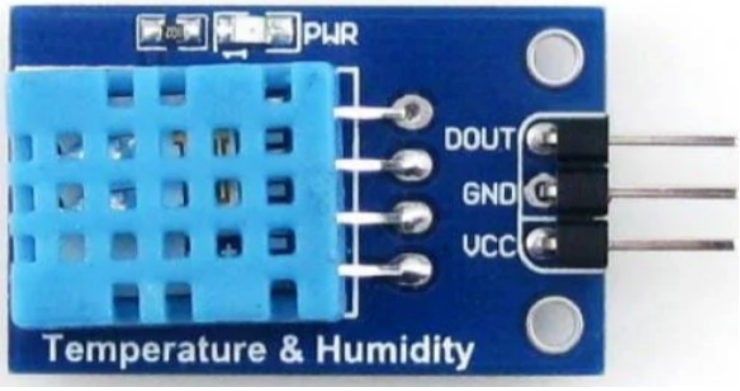
ESP32 adalah sebuah mikrokontroler yang dikembangkan oleh *Espressif Systems*, menawarkan kemampuan Wi-Fi dan *Bluetooth* terintegrasi yang menjadikannya pilihan populer untuk proyek IoT (Nizam et al., 2022). Menjelaskan bahwa ESP32 memiliki kelebihan dalam hal kinerja, konsumsi daya rendah, dan fleksibilitas, membuatnya ideal untuk berbagai aplikasi IoT. ESP32 dilengkapi dengan prosesor *dual-core*, memori yang cukup besar, dan berbagai antarmuka I/O, memungkinkan pengembang untuk menciptakan solusi IoT yang kompleks dan efisien.

Gambar 2. 1 ESP32 38 pin
Sumber : https://share.google/m7PKMhX6NoNf7auZD

Meskipun ESP32 menawarkan banyak keunggulan, menyoroti pentingnya mempertimbangkan aspek keamanan dalam implementasi ESP32 untuk IoT. Mereka mengidentifikasi beberapa potensi kerentanan yang perlu *diaddress*, termasuk keamanan *bootloader*, enkripsi flash, dan manajemen kunci. mendemonstrasikan implementasi protokol keamanan pada ESP32, menunjukkan bahwa dengan konfigurasi yang tepat, ESP32 dapat menjadi platform yang aman untuk aplikasi IoT(Prafanto et al., 2021). Namun, mereka juga menekankan pentingnya pembaruan firmware secara teratur dan implementasi *best practices* keamanan untuk memastikan keamanan jangka panjang perangkat berbasis ESP32.

2.2.3 DHT11

Sensor DHT11 merupakan salah satu sensor digital yang digunakan untuk mendeteksi suhu dan kelembaban udara dalam suatu lingkungan. Sensor ini cukup banyak digunakan pada sistem *Internet of Things* (IoT) karena memiliki bentuk yang sederhana, harga yang relatif terjangkau, serta mudah diintegrasikan dengan berbagai jenis mikrokontroler seperti ESP32 dan Arduino. Sensor jenis ini cukup banyak dipilih karena data keluaran yang dihasilkan sudah dalam bentuk digital sehingga tidak memerlukan lagi proses konversi dari sinyal analog (Saptadi, 2014). Pada proyek akhir ini, sensor DHT11 digunakan sebagai perangkat input utama untuk memperoleh data suhu dan kelembaban ruangan secara *real-time*.

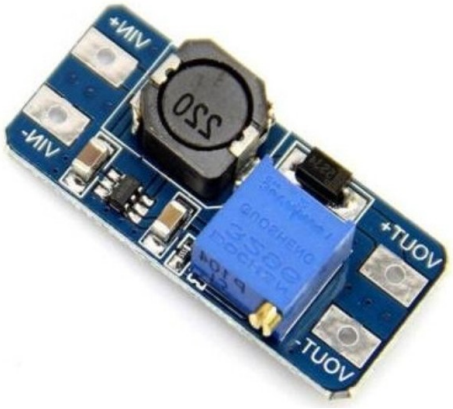


Gambar 2. 2 Sensor DHT11
Sumber : https://share.google/W0AMfslTM8rhRsvGb

Penggunaan sensor DHT11 pada sistem monitoring ini bertujuan untuk mengetahui kondisi lingkungan yang kemudian dikirimkan ke *server* menggunakan jaringan *WiFi* melalui ESP32. Data hasil pembacaan sensor akan ditampilkan pada LCD 20x4 serta *dashboard* berbasis *web* yang dibuat menggunakan Flask. Sensor DHT11 memiliki tingkat akurasi yang cukup baik untuk kebutuhan monitoring dasar dengan rentang pengukuran suhu sekitar 0°C hingga 50°C dan kelembaban 20% hingga 90%. Selain itu, komunikasi digital yang dimiliki sensor ini membuat proses pembacaan data menjadi lebih stabil dan mudah diterapkan pada sistem monitoring berbasis IoT.

2.2.4 MT3608

Modul MT3608 merupakan salah satu modul DC-DC *step up converter* yang digunakan untuk menaikkan tegangan dari sumber daya yang lebih rendah menjadi tegangan yang lebih tinggi sesuai kebutuhan perangkat. Modul ini cocok digunakan karena memiliki rentang tegangan yang sesuai dengan kebutuhan yaitu input 2.5Vdc sd 4.2 Vdc dan output 5 V(Bagenda, 2019). Pada proyek akhir ini, modul MT3608 digunakan sebagai penstabil dan peningkat tegangan dari baterai lithium agar mampu menyuplai daya yang sesuai untuk ESP32 dan perangkat pendukung lainnya. Dengan adanya modul ini, sistem dapat tetap bekerja secara mandiri tanpa harus selalu terhubung ke laptop atau sumber daya USB.

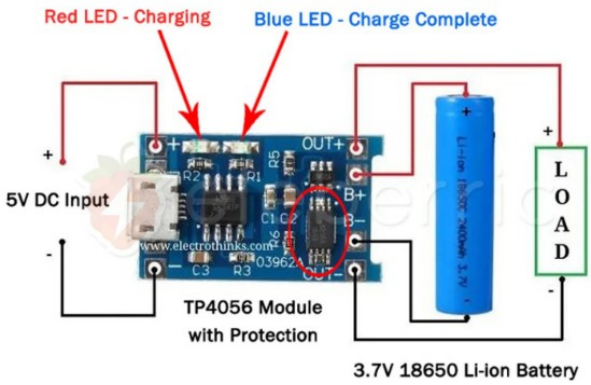


Gambar 2. 3 Modul MT3608
Sumber : https://share.google/y2f5DoHe6aRFzxhal

Penggunaan modul MT3608 pada sistem monitoring berbasis IoT ini bertujuan untuk menjaga kestabilan tegangan yang masuk ke ESP32 sehingga perangkat dapat bekerja dengan lebih optimal. Tegangan dari baterai lithium yang awalnya sekitar 3,7 volt dinaikkan menjadi 5 volt agar sesuai dengan kebutuhan catu daya ESP32 dan LCD. Selain itu, modul ini juga memiliki pengaturan *output* tegangan menggunakan potensiometer sehingga pengguna dapat menyesuaikan tegangan keluaran sesuai kebutuhan sistem. Dengan ukuran yang ringkas dan konsumsi daya yang efisien, modul MT3608 menjadi salah satu komponen pendukung yang penting dalam implementasi sistem monitoring berbasis IoT pada proyek akhir ini.

2.2.5 TP4056

Modul TP4056 merupakan sebuah modul yang digunakan untuk melakukan pengisian daya baterai *lithium* secara otomatis. Modul ini cukup sering digunakan pada proyek elektronika maupun *Internet of Things* (IoT) karena penggunaannya yang sederhana dan sudah dilengkapi dengan sistem proteksi baterai. Pada proyek akhir ini, TP4056 digunakan sebagai pengatur proses pengisian baterai *lithium* yang menjadi sumber daya utama untuk ESP32 dan komponen lainnya. Dengan adanya modul ini, alat monitoring dapat bekerja secara mandiri tanpa harus selalu terhubung langsung ke laptop ataupun adaptor USB. Selain itu, TP4056 juga membantu menjaga keamanan baterai dari kondisi *overcharge* dan *overdischarge* sehingga penggunaan baterai menjadi lebih aman dan stabil.



Gambar 2. 4 Modul TP4056
Sumber : https://share.google/eWxWq8P49mLEeUmUw

Seperti yang terlihat pada Gambar 2.4, modul TP4056 memiliki beberapa bagian penting seperti input tegangan 5V, terminal baterai lithium, serta terminal output menuju beban (*load*). Pada gambar tersebut juga terlihat indikator LED merah yang menandakan proses charging sedang berlangsung dan LED biru yang menunjukkan baterai sudah terisi penuh (Atsal et al., 2023). Baterai lithium 18650 dihubungkan ke pin B+ dan B− pada modul, kemudian output dari modul digunakan untuk menyuplai daya ke sistem monitoring. Dengan konfigurasi tersebut, proses pengisian daya dan penyaluran tegangan ke perangkat dapat berjalan dengan lebih praktis dan efisien.

2.2.6 LCD 20x4 i2c

LCD 20x4 I2C merupakan salah satu perangkat *output* yang digunakan untuk menampilkan informasi dalam bentuk teks pada sistem elektronika maupun *Internet of Things* (IoT). LCD ini mampu menampilkan hingga 20 karakter dan 4 baris tampilan sehingga informasi yang ditampilkan menjadi lebih banyak dan mudah dibaca. Pada proyek akhir ini, LCD 20x4 I2C digunakan untuk menampilkan data suhu, kelembaban, serta status koneksi dari sistem monitoring yang dibuat. Penggunaan modul I2C pada LCD juga membantu mengurangi jumlah pin yang digunakan pada ESP32 karena komunikasi data hanya menggunakan dua jalur, yaitu SDA dan SCL.

Gambar 2. 5 LCD 20x4 I2C
Sumber : https://share.google/rgFSqDd4jEfWHUFZG

Seperti yang terlihat pada Gambar 2.5, LCD 20x4 memiliki layar tampilan yang cukup lebar sehingga dapat digunakan untuk menampilkan beberapa informasi sekaligus. LCD ini terhubung dengan ESP32 menggunakan komunikasi I2C yang membuat proses pemasangan menjadi lebih sederhana dibandingkan LCD biasa. Pada implementasinya, data yang diperoleh dari sensor DHT11 akan diproses oleh ESP32 kemudian ditampilkan secara langsung pada LCD agar pengguna dapat memantau kondisi suhu dan kelembaban secara *real-time*. Selain itu, LCD juga digunakan untuk menampilkan status koneksi *WiFi* maupun TLS sehingga pengguna dapat mengetahui kondisi sistem secara langsung tanpa harus membuka *dashboard web*.

2.2.7 Protokol HTTP dalam IoT

HTTP adalah protokol aplikasi yang digunakan secara luas untuk sistem informasi terdistribusi. Dalam konteks IoT, HTTP sering digunakan karena sifatnya yang sederhana dan kompatibilitasnya dengan infrastruktur *web* yang ada. Protokol ini mempermudah komunikasi antara perangkat IoT dan *server* melalui internet, sehingga menjadi pilihan utama untuk berbagai aplikasi IoT.

Meskipun demikian, HTTP memiliki tantangan dalam hal efisiensi, terutama pada perangkat dengan sumber daya terbatas. Salah satu cara untuk meningkatkan performa HTTP dalam IoT adalah dengan menggunakan metode kompresi data atau optimasi pengelolaan sesi untuk mengurangi konsumsi

bandwidth. Selain itu, implementasi HTTP versi terbaru memberikan keuntungan berupa efisiensi yang lebih baik melalui multiplexing, pengurangan *latency*, dan penggunaan *header* yang lebih kecil.

Dengan optimasi dan teknologi pendukung yang tepat, HTTP tetap menjadi solusi yang dapat diandalkan untuk mengelola komunikasi perangkat IoT, terutama dalam sistem yang memerlukan interaksi dengan layanan *web* atau aplikasi berbasis cloud.

2.2.8 Transport Layer Security (TLS)

TLS adalah protokol kriptografi yang menyediakan komunikasi aman melalui jaringan, menjamin kerahasiaan dan integritas data yang ditransmisikan(Dharmawan et al., 2022). Dalam konteks IoT, TLS berperan penting dalam mengamankan komunikasi antara perangkat dan *server*, melindungi dari berbagai serangan seperti *Man-in-the-Middle* dan *eavesdropping*.

Optimasi diperlukan untuk implementasi TLS pada perangkat dengan sumber daya terbatas, termasuk penggunaan algoritma kriptografi yang lebih ringan dan optimasi proses *handshake*. Panduan praktis untuk mengamankan komunikasi IoT mencakup pemilihan *cipher suite* yang tepat, manajemen sertifikat yang efektif, dan implementasi *perfect forward secrecy*. Dengan pendekatan yang sesuai, TLS dapat diimplementasikan secara efisien bahkan dalam lingkungan IoT yang terbatas.

2.2.9 Flask

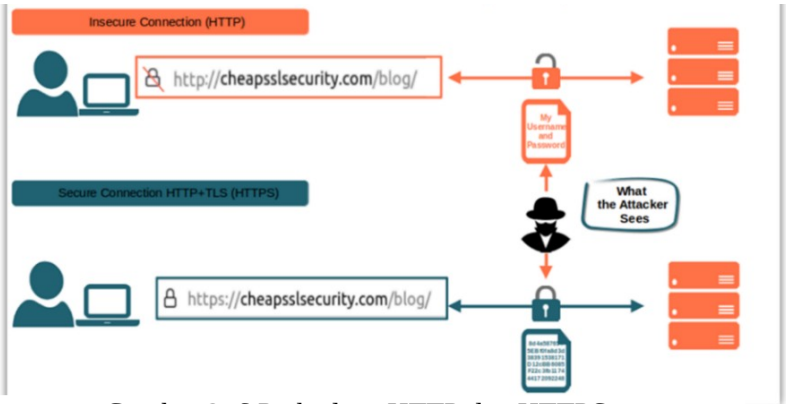
Flask merupakan salah satu *Framework* berbasis bahasa pemrograman Python yang digunakan untuk membangun aplikasi *web* dengan konsep sederhana dan ringan. *Framework* ini banyak digunakan dalam pengembangan sistem *Internet of Things* (IoT) karena memiliki proses konfigurasi yang mudah serta mampu menangani komunikasi data antara perangkat dan *server* secara *real-time*. Pada proyek akhir ini, *Flask* digunakan sebagai *web server* yang berfungsi untuk menerima data suhu dan kelembaban yang dikirim oleh ESP32 melalui jaringan *WiFi*. Selain itu, *Flask* juga digunakan untuk menampilkan data monitoring ke dalam bentuk *dashboard web* sehingga pengguna dapat melihat kondisi sensor secara langsung melalui *browser*.

Penggunaan *Flask* pada sistem ini dipilih karena memiliki performa yang cukup baik untuk pengembangan aplikasi monitoring berbasis IoT dengan skala kecil hingga menengah. *Flask* juga mendukung komunikasi menggunakan protokol HTTP maupun HTTPS sehingga keamanan pertukaran data dapat diterapkan menggunakan TLS (*Transport Layer Security*). Dengan adanya Flask, proses pengiriman data dari sensor DHT11 menuju *server* menjadi lebih terstruktur dan mudah untuk dikembangkan lebih lanjut, seperti penambahan database, grafik monitoring, maupun integrasi dengan perangkat lain. Selain itu, struktur program *Flask* yang sederhana memudahkan proses implementasi dan pengujian sistem pada proyek akhir ini.

2.2.10 Keamanan Jaringan IoT

Keamanan jaringan IoT mencakup berbagai aspek untuk melindungi perangkat dan data dalam ekosistem IoT yang kompleks. Tantangan utama dalam keamanan IoT meliputi otentikasi, kontrol akses, dan privasi. Pendekatan keamanan berlapis disarankan untuk memastikan perlindungan yang efektif, mulai dari perangkat fisik hingga layanan cloud.

Protokol HTTP yang digunakan dalam IoT sering dipadukan dengan *Transport Layer Security* (TLS) untuk memastikan komunikasi yang aman. TLS menyediakan enkripsi data, autentikasi, dan integritas, melindungi komunikasi dari ancaman seperti serangan *Man-in-the-Middle* dan *eavesdropping* (Wahyudi, 2024). Penerapan HTTP dengan TLS (sering disebut HTTPS) memungkinkan perangkat IoT berkomunikasi secara aman dengan *server*, menjaga kerahasiaan informasi sensitif seperti data pengguna atau kontrol perangkat.



Gambar 2. 6 Perbedaan HTTP dan HTTPS
Sumber : https://images.app.goo.gl/nuZzWD9HZmiELowS

Gambar 2.6 tersebut menjelaskan perbedaan antara protokol HTTP (*HyperText Transfer Protocol*) dan HTTPS (HTTP Secure) dalam konteks keamanan komunikasi data. Pada bagian atas, protokol HTTP digambarkan sebagai koneksi yang tidak aman karena data yang dikirimkan antara pengguna dan *server* tidak dilindungi oleh enkripsi. Akibatnya, informasi sensitif seperti nama pengguna dan kata sandi dapat dengan mudah diakses oleh pihak ketiga yang berhasil menyadap koneksi. Hal ini menimbulkan risiko terhadap kerahasiaan dan integritas data, karena data yang dikirimkan dapat terlihat secara langsung oleh penyerang tanpa perlindungan.

Sementara itu, bagian bawah menggambarkan koneksi HTTPS yang menggunakan enkripsi TLS (*Transport Layer Security*) untuk melindungi data yang ditransmisikan. Dengan penerapan TLS, data akan dienkripsi sebelum dikirimkan, sehingga meskipun data berhasil disadap oleh pihak ketiga, informasi tersebut hanya akan terlihat sebagai string acak yang tidak dapat dibaca tanpa kunci dekripsi. Teknologi ini menjamin kerahasiaan, integritas, dan autentikasi data, serta mampu mencegah serangan seperti *Man-in-the-Middle attack*. Oleh karena itu, penerapan HTTPS dengan TLS menjadi langkah penting dalam meningkatkan keamanan komunikasi jaringan, khususnya pada perangkat berbasis IoT yang memiliki kerentanan tinggi terhadap ancaman keamanan.

Untuk perangkat IoT dengan sumber daya terbatas, optimasi implementasi TLS diperlukan agar efisien, seperti menggunakan cipher suite ringan atau pengelolaan sertifikat yang disederhanakan. Dengan penerapan protokol HTTP yang dikombinasikan dengan enkripsi TLS, sistem IoT dapat menyediakan keamanan komunikasi yang kuat tanpa mengorbankan fleksibilitas dan kompatibilitasnya dengan infrastruktur *web* yang ada.

2.2.11 Ancaman Siber Terhadap HTTPS

Dalam pengembangan sistem keamanan berbasis HTTPS dan enkripsi TLS, terdapat berbagai jenis serangan siber yang dapat mengancam integritas dan kerahasiaan data yang ditransmisikan. Berikut ini merupakan beberapa serangan yang relevan untuk dibahas dalam konteks makalah "Analisis dan Pengembangan

Keamanan Jaringan IoT Berbasis ESP32 Menggunakan Protokol HTTP dan Enkripsi TLS":

1) SSL Stripping

   Serangan SSL Stripping memanfaatkan celah dalam transisi dari HTTP ke HTTPS. Serangan ini menurunkan koneksi yang seharusnya aman (HTTPS) menjadi tidak aman (HTTP). Contohnya, jika pengguna mengunjungi halaman *web* melalui URL HTTP, peretas dapat mencegat koneksi tersebut dan memastikan data dikirim tanpa enkripsi. Akibatnya, informasi sensitif seperti kata sandi atau data transaksi rentan dicuri karena tidak terlindungi oleh TLS. Metode ini sering kali terjadi pada jaringan publik seperti Wi-Fi di tempat umum.

2) *Man-in-the-Middle* (MITM)

   Serangan MITM adalah salah satu ancaman paling umum terhadap koneksi HTTPS. Dalam serangan ini, peretas memposisikan diri di antara pengguna dan *server* untuk mencegat komunikasi. Dengan mengelabui pengguna untuk percaya bahwa mereka terhubung ke *server* asli, peretas dapat mencuri informasi seperti kredensial login atau detail keuangan. Serangan ini semakin berbahaya jika sertifikat SSL/TLS atau *Certificate Authority* (CA) yang bertanggung jawab atas verifikasi digital berhasil diretas.

3) Serangan pada Sertifikat Tidak Tepercaya

   Sertifikat digital yang tidak tepercaya atau palsu sering digunakan oleh peretas untuk menyamar sebagai *server* asli. Hal ini memungkinkan serangan phishing dengan cara membuat pengguna percaya bahwa mereka sedang berkomunikasi dengan *server* tepercaya, padahal sebenarnya informasi mereka sedang dicuri oleh penyerang.

2.2.12 Visual Code

Visual Studio Code merupakan salah satu text editor yang banyak digunakan dalam pengembangan perangkat lunak karena memiliki tampilan yang ringan, mudah digunakan, serta mendukung berbagai bahasa pemrograman. Aplikasi ini dikembangkan oleh Microsoft dan menyediakan berbagai fitur seperti syntax *highlighting*, *debugging*, terminal, serta *extension* tambahan yang membantu proses pembuatan program menjadi lebih mudah. Pada proyek akhir ini, *Visual Studio*

Code digunakan sebagai media untuk membuat dan mengedit program *Flask* yang berfungsi sebagai *server* monitoring pada sistem IoT berbasis ESP32.

Penggunaan *Visual Studio Code* pada proyek ini membantu proses pengembangan program menjadi lebih terstruktur dan efisien. Melalui aplikasi ini, kode program Python untuk *Flask* dapat dijalankan dan diuji secara langsung menggunakan terminal bawaan. Selain itu, *Visual Studio Code* juga memudahkan proses pengelolaan file program seperti file *dashboard*, sertifikat TLS, dan script monitoring lainnya. Dengan fitur yang cukup lengkap dan tampilan yang sederhana, *Visual Studio Code* menjadi salah satu *tools* pendukung yang membantu dalam proses implementasi sistem monitoring suhu dan kelembaban berbasis IoT pada proyek akhir ini.

2.2.13 OpenSSL

Untuk menguji keamanan enkripsi TLS dalam sistem IoT, digunakan OpenSSL sebagai alat utama. OpenSSL adalah pustaka perangkat lunak sumber terbuka yang mendukung protokol kriptografi seperti SSL (*Secure Sockets Layer*) dan TLS (*Transport Layer Security*). Selain itu, OpenSSL menyediakan berbagai alat bantu untuk mengelola, memverifikasi, dan menganalisis koneksi terenkripsi.

OpenSSL dipilih karena fleksibilitas dan kemampuannya untuk digunakan dalam berbagai skenario pengujian. Alat ini memungkinkan pengujian aspek-aspek penting dari TLS, seperti otentikasi sertifikat *server*, kerahasiaan data, integritas data, dan kinerja. Dengan OpenSSL, pengembang dapat melakukan simulasi koneksi HTTPS untuk memverifikasi bahwa data terenkripsi dengan benar dan tidak dapat diakses oleh pihak yang tidak berwenang.

Gambar 2. 7 OpenSSL
Sumber : https://images.app.goo.gl/qEFKrHyYHh1hLn3N9

Selain itu, OpenSSL mendukung pengujian performa, yang bertujuan untuk mengevaluasi dampak proses enkripsi terhadap kecepatan transmisi data dan efisiensi sistem. Perintah seperti speed dapat digunakan untuk mengukur kecepatan enkripsi dan dekripsi dengan berbagai algoritma.

OpenSSL juga memungkinkan simulasi serangan sederhana, seperti memverifikasi apakah data yang dikirimkan aman dari ancaman *sniffing* atau manipulasi oleh pihak ketiga. Dengan kombinasi fitur-fitur tersebut, OpenSSL menjadi alat yang sangat efektif dalam memastikan implementasi TLS berjalan sesuai dengan standar keamanan yang diinginkan.

Melalui penggunaan OpenSSL, pengujian dapat dilakukan secara komprehensif, sehingga memberikan kepercayaan lebih terhadap sistem IoT yang dikembangkan, khususnya dalam aspek perlindungan data selama transmisi.

2.2.14 Wireshark

Wireshark adalah alat analisis jaringan yang digunakan untuk memonitor dan menganalisis data yang dikirim melalui jaringan secara *real-time* (Luthfansa & Rosiani, 2021). Dalam proyek ini, *Wireshark* digunakan untuk memastikan bahwa data yang dikirimkan dari ESP32 ke *server Flask* melalui protokol HTTPS (HTTP + TLS) telah terenkripsi dengan benar. Dengan *Wireshark*, data yang dikirimkan dapat diperiksa apakah dalam bentuk *plaintext* (teks biasa) atau telah dienkripsi, sehingga membantu memvalidasi implementasi keamanan protokol TLS.

Gambar 2. 8 Wireshark
Sumber : https://images.app.goo.gl/wVomYnbekJW886ct9

Selain itu, *Wireshark* juga memungkinkan analisis detail terhadap proses *Handshake* TLS, termasuk pertukaran kunci, algoritma enkripsi yang digunakan, dan sertifikat yang diterapkan oleh *server*. Penggunaan *Wireshark* memberikan jaminan bahwa komunikasi IoT dalam sistem ini aman dari potensi serangan seperti *sniffing* atau *Man-in-the-Middle* (MITM). Dengan alat ini, pengembang dapat memastikan integritas, kerahasiaan, dan autentikasi data secara menyeluruh.