STORAGE – Jurnal Ilmiah Teknik dan Ilmu Komputer, Vol. 4 No. 4, November 2025, 302 – 314
e-ISSN: 2828-5344 | DOI: 10.55123

KEAMANAN TRANSMISI DATA PADA SISTEM IOT MELALUI METODE CHECKSUM SHA-256, AES-128 UNTUK PENCEGAHAN DATA TAMPERING

[1)]I Putu Bayu Krisna Priastawan, [2)]Nugroho budhisantosa, [3)]Popong Setiawati, [4)]Imam Sutanto

[1,2,3,4)] Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Esa Unggul
[1)]priastawaniputu@student.esaunggul.ac.id, [2)]Nugroho.budhisantosa@esaunggul.ac.id,
[3)]popong.setiawati@esaunggul.ac.id, [4)]Imam.sutanto@esaunggul.ac.id

INFO ARTIKEL

Riwayat Artikel :
Diterima : 9 September 2025
Disetujui : 12 September 2025

Kata Kunci :
Cyber Security, *MQTT*, *Data Tampering*, *SHA-256*, *AES-128*

ABSTRAK

MQTT adalah protokol komunikasi dalam IoT yang beroperasi menggunakan skema *publish-subscribe*, di mana *publisher* mengirimkan data ke broker dan data tersebut diteruskan kepada *subscriber* yang berlangganan pada *topic* terkait. Melindungi data dari ancaman serangan siber menjadi salah satu aspek penting, terutama dari upaya manipulasi seperti *data tampering attack* yang dapat membuatnya menjadi tidak akurat dan bahkan berpotensi menghasilkan informasi palsu. Penggunaan metode seperti *checksum SHA-256*, teknik *salt*, dan *AES-128* dapat dimanfaatkan untuk mencegah ancaman serangan *data tampering*. Berdasarkan hasil pengujian, didapatkan rata-rata pengujian deteksi dan pencegahan *data tampering attack* menunjukkan 100%, dan rata-rata data yang berhasil diterima oleh *subscriber* adalah 94% dari seluruh transmisi *topic* data selama pengujian. Sistem keamanan yang dibangun memang belum sempurna, karena sistem baru dapat mencegah serangan di sisi *subscriber*. Untuk sepenuhnya mencegah *data tampering attack*, diperlukan Langkah keamanan tambahan seperti *TLS*, *SSL*, atau menggunakan port 8883 pada *MQTT* agar komunikasi dapat berlangsung secara privat.

ARTICLE INFO

Article History :
Received : Sept 9, 2025
Accepted : Sept 12, 2025

Keywords:
Cyber Security, *MQTT*, *Data Tampering*, *SHA-256*, *AES-128*

ABSTRACT

MQTT is a communication protocol in IoT that operates using a publish-subscriber scheme, where the publisher sends data to the broker and the data is forwarded to subscribers that are subscribed to the respective topic. Protecting data from cyberattack threats is one of the critical aspects, particularly against manipulation attempts such as data tampering attacks that can make it inaccurate and even potentially produce false information. Therefore, methods such as the SHA-256 checksum, the salt technique, and AES-128 can be utilized to prevent data tampering threats. Based on the testing results, the average detection and prevention rate of data tampering attacks reached 100%, and the average amount of data successfully received by the subscriber was 94% of the total topic data transmissions during the testing. The developed security system is not yet perfect, because the system can only prevent attacks on the subscriber side. To fully prevent data tampering attack, additional security measures such as TLS, SSL, or using port 8883 on MQTT are required so that communication can take place privately.

1 PENDAHULUAN

Internet of Things (IoT) adalah konsep suatu perangkat ditanamkan teknologi yang dapat memungkinkan perangkat-perangkat tersebut terhubung untuk saling berinteraksi satu sama lain melalui internet. Perangkat IoT dapat berupa sensor, perangkat cerdas (*smart device*), ataupun *smart system* lainnya. Perangkat-perangkat tersebut mampu mengumpulkan data dari lingkungan sekitar dan berbagi data dengan perangkat lain (Rachmadini & Puspasari, 2020). Perangkat IoT berkomunikasi menggunakan protokol MQTT yang memiliki konsep sederhana dengan skema *publish-subscribe* yang terdiri dari *publisher, broker,* dan *subscriber*.

Data menjadi sebuah salah satu aset paling berharga dalam aplikasi IoT, oleh karena itu melindungi data dari upaya manipulasi seperti serangan *data tampering* sangatlah krusial. Serangan *data tampering* dapat memanipulasi nilai data untuk mempengaruhi Keputusan perangkat IoT menjadikannya tidak akurat, bahkan menghasilkan sebuah informasi palsu, dan menyebabkan gangguan kelancaran pada lingkungan *smart system* IoT pengguna(Kumar Shrivastava et al., 2019). Seperti kasus serangan pada fasilitas pengolahan air di amerika serikat, pelaku serangan *cyber* yang tidak teridentifikasi tersebut, berhasil mendapatkan akses illegal ke *system supervisory control and data acquisition* (SCADA) pada sebuah fasilitas pengolahan air minum di amerika serikat. Pelaku memanfaatkan perangkat lunak SCADA untuk melakukan manipulasi data dengan meningkatkan kadar natrium hidroksida, bahan kimia kaustik yang digunakan dalam proses pengolahan air. Namun, petugas fasilitas dengan cepat menyadari perubahan tersebut dan segera mengembalikan dosis ke tingkat yang aman(TLP:WHITE Compromise of U.S. Water Treatment Facility SUMMARY, 2021). Meskipun tidak terdapat dampak apapun yang terjadi dari serangan tersebut berkat tanggapan yang cepat dari petugas, insiden ini menyoroti pentingnya meningkatkan keamanan siber dalam melindungi infrastruktur dari ancaman serupa yang memungkinkan akan terjadi lagi.

Penggunaan metode *checksum* SHA-256, teknik *salt*, AES-128 dapat dimanfaatkan dalam mencegah ancaman serangan *data tampering*. *checksum* adalah string biner berukuran tetap yang dibuat dari blok data secara acak, checksum berfungsi untuk memeriksa integritas data, memastikan apakah data telah mengalami perubahan setelah checksum dibuat. Checksum direpresentasikan dalam bentuk string *hexadecimal*, misalnya seperti (2cae915ae0e…) dengan Panjang berkisar antara 32 hingga 128 digit. *Checksum* memiliki 3 fungsi diantaranya, *pre-image-resistance* (ketahanan terhadap tebakan awal), *second pre-image-resistance* (ketahanan terhadap tebakan kedua), dan *collision resistance* (mencegah dua data berbeda menghasilkan *hash* yang sama)(Meylan et al., 2020). SHA-256 adalah algoritma *hash* dari keluarga SHA-2 yang menghasilkan *output* berupa *message digest* sepanjang 256 *bit*. SHA-256 dapat memproses pesan dengan Panjang hingga 2^{64} *bit* yang akan diolah dalam blok berukuran 512 *bit*. Tahapan awal proses SHA-256 melibatkan *padding*, yaitu menambahkan *bit* tertentu pada pesan input hingga panjangnya menjadi kelipatan 512 *bit*. Setelah mengubah *input* pesan menjadi beberapa blok, selanjutnya yaitu menambahkan *bit* bernilai 1 dan 0 pada blok terakhir. Setelah pesan dibagi dan dilakukan *padding*, blok-blok tersebut digunakan sebagai *input* untuk proses perhitungan SHA-256 pada pemrosesan utama yang dilakukan melalui fungsi kompresi. Fungsi kompresi dalam SHA-256 memproses setiap blok pesan melalui 64 putaran (*rounds*) dengan menggunakan *message schedule* serta parameter yang telah ditentukan dalam algoritma SHA-256 (Anugrah et al., 2019). *Salt* adalah data tambahan berupa teks unik yang digunakan untuk menambahkan nilai khusus pada sebuah pesan sebelum proses *hashing* dilakukan. Penggunaan nilai *salt* dapat dipilih secara tetap ataupun acak. Penambahan *salt* bertujuan untuk meningkatkan keamanan informasi sebuah data, dengan memastikan bahwa meskipun data asli memiliki nilai yang sama, maka hasil *hash* yang dihasilkan akan berbeda karena adanya variasi pada nilai *salt*. nilai *hash* yang dilengkapi dengan *salt* dapat mengatasi risiko serangan *brute force* dan *rainbow table*(Rathod et al., 2020). AES merupakan salah satu algoritma yang berfungsi untuk melindungi privasi dan kerahasiaan data yang dikirimkan melalui jaringan komputer. Algoritma ini termasuk dalam kategori *symmetric block cipher* dan

menjadi yang paling banyak digunakan. AES menggunakan satu kunci yang sama untuk proses enkripsi dan dekripsi, sehingga baik pengirim maupun penerima harus memiliki kunci yang identik. AES dirancang dengan struktur yang khusus dalam melakukan proses enkripsi dan dekripsi terhadap data yang bersifat sensitive. Penggunaan AES memungkinkan sangat sulit untuk diretas oleh pihak yang tidak bertanggung jawab untuk memperoleh data asli dari hasil enkripsi dengan algoritma tersebut. AES mendukung tiga variasi panjang kunci, yaitu 128 bit, 192 bit, dan 256 bit(Al-Mashhadani & Shujaa, 2022).

Adapun beberapa penelitian yang telah dilakukan sebelumnya, seperti pada penelitian dengan judul Rancang Bangun Sistem Keamanan Perangkat IoT Dengan metode Autentikasi Menggunakan JSON WEB Token Pada Protokol MQTT. Penelitian ini bertujuan untuk mengamankan protokol komunikasi MQTT yang rentan terhadap serangan siber dengan menerapkan autentikasi menggunakan JSON Web Token dan protokol TLS. hasil yang didapat pada penelitian ini menunjukkan sistem yang dibangun dapat melakukan autentikasi pada setiap token JWT yang diterima, mendeteksi token JWT yang memiliki tanda tandang *invalid* dan mengirimkan notifikasi melalui aplikasi telegram, serta sistem yang dibangun dapat menangkal serangan MITM berupa *sniffing* dengan menerapkan protokol keamanan TLS(*Transport Layer Security*)(Mangkurat, n.d.).

Adapun penelitian lainnya dengan judul IoT *Security Using AES Encryption Technology based ESP32 Platform*. Penelitian ini bertujuan untuk mengamankan data pada sistem IoT dengan algoritma AES. Proses enkripsi pada algoritma AES dapat menjaga kerahasiaan dan keaslian data, serta mencegah akses tidak sah. Hasil yang didapat pada penelitian ini menunjukkan sistem mampu mengenkripsi dan mendekripsi data dengan benar, baik dari data yang dikirimkan maupun data yang diterima. Penggunaan algoritma AES pada penelitian tersebut, mampu memberikan perlindungan yang kuat terhadap keamanan data sensor serta menjaga keaslian data (Al-Mashhadani & Shujaa, 2022).

Adapun penelitian lainnya dengan judul *Preventing Data Tampering in IoT Networks*. Tujuan penelitian ini adalah untuk melindungi data dari kerusakan ataupun manipulasi yang dilakukan oleh pihak yang tidak bertanggung jawab. Penelitian ini membangun sistem *honey sensors* untuk mengatasi permasalahan tersebut. *honey sensors* adalah nilai data palsu atau sebuah umpan yang sengaja disisipkan ke dalam data asli untuk mengidentifikasi upaya manipulasi oleh penyerang. Hasil dari pengujian yang dilakukan pada penelitian ini, sistem *honey sensor* dapat dimanfaatkan dalam mendeteksi upaya modifikasi atau perubahan data secara tidak sah(Kumar Shrivastava et al., 2019).

Adapun penelitian lainnya dengan judul Develop security and privacy algorithms for cyber system. Penelitian ini membangun sistem keamanan menggunakan protokol TLS, yang merupakan protokol yang menjaga komunikasi dan melindungi data yang dipertukarkan antar perangkat, sekaligus memastikan keaslian serta integritas data tersebut. TLS memiliki dua protokol utama yaitu TLS *handshake protocol* dan TLS *record protocol*. Protokol ini membagi pesan keluar menjadi blok-blok yang dapat dikendalikan dan menyatukan Kembali pesan yang masuk, melakukan enkripsi pada pesan keluar serta dekripsi pada pesan masuk menggunakan algoritma enkripsi seperti (AES, Camellia, ARIA, atau lainnya). Menerapkan *message authentication code* (MAC) pada pesan keluar dan memverifikasi pesan masuk dengan MAC yang sama. Setelah proses pada TLS *record* selesai, data terenkripsi yang akan dikirimkan diteruskan ke lapisan *transmisiion control protocol* (TCP) untuk ditransmisikan. Hasil eksperimen menunjukkan kinerja unggul dari metode yang diajukan dalam mengamankan sebuah *physical system* dengan mendeteksi dan mencegah serangan siber yang menargetkan *server web*. Dan membuktikan penerapan aspek keamanan, privasi, dan integritas pada *physical system*(Khudhur & Croock, 2021).

Adapun penelitian lainnya dengan judul Research of Secure Communication of Esp32 IoT Embedded System to .NET Core Cloud Structure using MQTTS SSL/TLS. Pada penelitian ini digunakan protokol keamanan SSL dan TLS yang digunakan pada port 8883 yang tersedia pada komunikasi MQTT untuk

mengenkripsi komunikasi dengan menggunakan *private key* dan *public key*, *certificate* untuk menjamin mengamankan koneksi serta mengenkripsi pertukaran data antar komponen sistem. Proses enkripsi berhasil diterapkan pada penelitian ini, pesan yang dipertukarkan antar IoT *Cloud* dan perangkat IoT. Dengan adanya enkripsi, pesan menjadi terenkapsulasi sehingga tidak dapat disadap(Nikolov Neven & Nakov Ognyan, 2019).

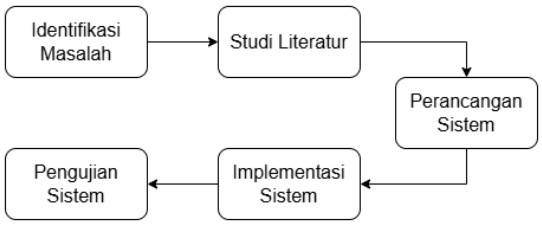
Dalam memaksimalkan keamanan integritas data terhadap ancaman serangan seperti *data tampering*, diperlukan sebuah metode checksum SHA-256, teknik *salt*, AES-128 yang digunakan pada penelitian ini dengan tujuannya sebagai berikut:

Pertama, Menghasilkan sebuah sistem keamanan transmisi data pada perangkat IoT yang dapat mencegah serangan *data tampering*.

Kedua, Mengetahui efektivitas dari sistem keamanan tersebut dalam melindungi integritas data dan mencegah serangan *data tampering* yang dilakukan secara manual, dari uji simulasi yang dilakukan melalui *tools MITMProxy(Marco TORCHIANO Doc Maximilian HILS, 2024)*.

2 METODE

Tahapan yang dilakukan pada penelitian ini dilakukan sebagai berikut:

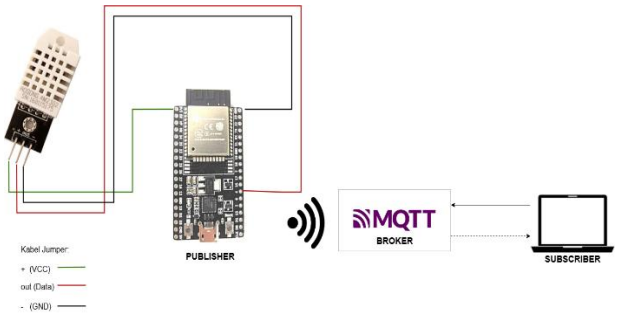


Gambar 1. Rencana Tahapan Penelitian

Tahapan awal penelitian dimulai dengan identifikasi masalah serta studi literatur yang telah dilakukan pada bab sebelumnya pada penelitian ini. Rencana penelitian dilanjutkan dengan tahapan-tahapan berikutnya yang mencakup perancangan sistem, implementasi sistem, pengujian sistem, dan penyusunan laporan.

2.1 Perancangan Sistem

Pada penelitian ini, berfokus pada perancangan sistem keamanan pada perangkat IoT menggunakan metode *checksum* SHA-256, teknik *salt*, serta algoritma AES-128 yang digunakan untuk menjaga keamanan integritas data. Perancangan sistem dilakukan pada perangkat IoT *Microcontroller NodeMCU* ESP32 serta sensor DHT22. Berikut adalah gambar rancangan sistem yang dapat dilihat dibawah ini.



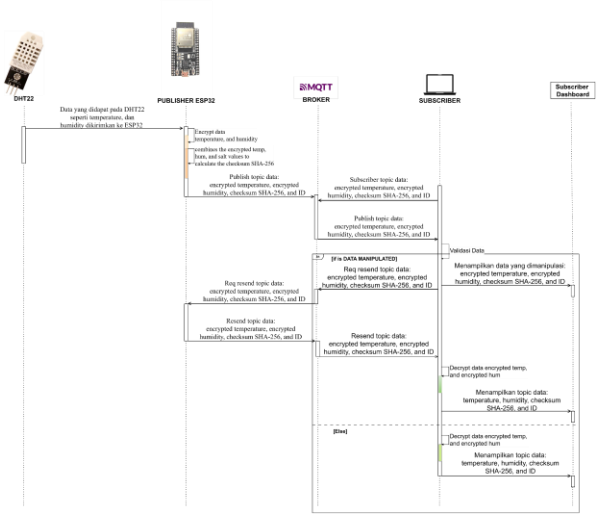
Gambar 2. Model Rancangan Sistem

Pada perancangan sistem seperti gambar 2 diatas, memiliki 3 komponen utama yaitu *publisher*, *broker*, dan *subscriber*. Pada komponen *publisher* terdapat 2 perangkat IoT yaitu *microcontroller nodeMCU* ESP32 dan DHT22, lalu pada komponen MQTT broker terdapat HiveMQ yang digunakan sebagai perantara jalur komunikasi diantara *publisher* dan *subscriber*. dan untuk komponen *subscriber* adalah *node-red* yang merupakan sebuah editor visual *open-source*, yang dimana pada penelitian ini digunakan untuk memonitoring topik yang dikirimkan dari *publisher* melalui MQTT broker(Hue et al., 2022). komponen *publisher* akan mengirimkan data berupa *temperature*, *humidity* yang telah dienkripsi, nilai checksum yang dihasilkan dari perhitungan algoritma SHA-256, serta ID sebagai penanda unik untuk setiap data. Lalu data-data tersebut dikirimkan melalui komponen MQTT broker yaitu HiveMQ dan *subscriber* akan menerima data sesuai dengan apa yang dikirimkan dari *publisher*. Validasi data akan dilakukan dengan membandingkan nilai *checksum* SHA-256 yang diterima dengan nilai *checksum* SHA-256 yang dihitung ulang, apabila hasil dari validasi data menyatakan *data valid* maka proses dekripsi data *temperature*, *humidity* akan dilakukan dengan menggunakan algoritma AES. Jika data dinyatakan tidak valid atau *data manipulated* maka proses dekripsi tidak akan dilakukan. Dan *subscriber* akan melakukan *request* pengiriman ulang data ke *publisher* untuk

mendapatkan Kembali data asli yang tidak dimanipulasi.

2.2 Perancangan Sistem Keamanan

Perancangan sistem keamanan yang diterapkan pada perangkat IoT dengan menggunakan *checksum* SHA-256, teknik *salt*, serta algoritma AES-128, dapat dilihat seperti pada gambar berikut.



Gambar 3. Rancangan Sistem Keamanan

Proses pada gambar diatas, dimulai melalui DHT22 yang mendeteksi *temperature* dan *humidity* di lingkungan sekitarnya. Setelahnya kedua data tersebut dikirimkan ke ESP32, dan data-data tersebut dienkripsi menggunakan algoritma AES-128 untuk mendapatkan *ciphertext* dari data *temperature*, *humidity*. Selanjutnya menggabungkan ciphertext *temperature*, *humidity*, dan nilai *salt* untuk menghitung *hash*-nya menggunakan SHA-256. Lalu data-data seperti *temperature*, dan *humidity* yang telah dienkripsi, nilai *checksum* SHA-256, dan ID sebagai penanda unik untuk setiap data, dikirimkan melalui protokol MQTT broker yaitu HiveMQ ke *subscriber*. Dan *subscriber* akan menerima data-data tersebut, data *temperature*, *humidity* yang terenkripsi, serta nilai *salt* digabungkan untuk melakukan perhitungan ulang disisi *subscriber* menggunakan algoritma SHA-256 untuk menghasilkan checksum. Dan proses akhir dilakukan validasi data dengan melakukan perbandingan diantara checksum yang diterima oleh *subscriber* dengan nilai checksum yang dihasilkan melalui perhitungan ulang di sisi *subscriber*. Proses validasi akan menghasilkan dua kategori *output*, yang dimana ketika nilai *checksum* yang dihitung ulang berbeda dengan nilai *checksum* yang diterima oleh *subscriber*, maka *output* dari proses validasi data akan menginformasikan bahwa *Data Manipulated*. Proses dekripsi akan dihentikan dan *subscriber* akan melakukan *request* pengiriman ulang data ke *publisher* untuk mendapatkan kembali data asli yang tidak dimanipulasi. Sebaliknya ketika kedua nilai *checksum* tersebut memiliki nilai yang sama maka proses dekripsi dilanjutkan untuk mendapatkan *plaintext* data *temperature*, *humidity* serta menginformasikan bahwa *Data Valid*.

2.3 Perancangan Sistem Komunikasi

Dalam rancangan sistem komunikasi pada penelitian ini, menggunakan MQTT broker yaitu HiveMQ. Komunikasi berlangsung dengan skema *publish – subscribe*. Dalam skema ini, *publisher* akan mengirimkan data ke broker melalui jalur tertentu yang disebut sebagai "topik". Dan *subscriber* dapat melakukan *subscriber* dengan jenis topik yang ada dan yang dipilih oleh *subscriber*. Yang dimana pada penelitian ini, perangkat ESP32 sebagai *publisher* akan mengirimkan data *temperature*, *humidity* yang telah dienkripsi dengan algoritma AES-128, nilai *checksum* SHA-256, serta ID sebagai penanda unik untuk setiap data, ke MQTT broker yaitu HiveMQ. Dengan menggunakan *port* 1883 sebagai jalur koneksi untuk komunikasi diantara *publisher* dan *subscriber*. Lalu di sisi *subscriber* akan menerima data *temperature*, *humidity*, yang masih terenkripsi serta nilai checksum SHA-256, dan ID. Data-data *temperature*, *humidity*, dan nilai akan digabungkan untuk melakukan perhitungan ulang dengan algoritma SHA-256. Hasil nilai *checksum* yang didapat dari hasil perhitungan ulang akan melalui proses validasi dengan melewati proses perbandingan nilai *checksum* yang dihitung ulang dengan nilai *checksum* yang diterima untuk memastikan integritas data.

2.4 Implementasi Sistem

2.4.1 Implementasi Program Publisher

Implementasi program *publisher* dilakukan pada *software* Arduino IDE(Kelechi et al., 2021). Tahap implementasi kode program *publisher* ini,

dimulai dengan melakukan koneksi ke jaringan WiFi melalui konfigurasi yang dilakukan dengan menggunakan *SSID* dan *Password* WiFi yang telah ditentukan, lalu dilanjutkan dengan mengatur koneksi ke broker MQTT menggunakan Alamat server broker.hivemq.com dan *port* 1883. Dilanjutkan dengan konfigurasi sensor DHT22, dengan menentukan pin GPIO tempat sensor DHT22 terhubung pada perangkat ESP32. Yang dimana pada penelitian ini, pin yang ditentukan pada perangkat ESP32 ialah pada GPIO 4. Selanjutnya menentukan pin GPIO 3V3 untuk menyediakan daya Listrik dan pin GPIO GND untuk *ground* sensor. Pada perangkat sensor DHT22 akan membaca *temperature* dan *humidity* pada lingkungan sekitar, DHT22 akan memprediksi *temperature* serta *humidity*, dan mengirimkan data tersebut ke ESP32. Lalu data yang dikirimkan tersebut, akan diproses melalui algoritma AES untuk mengenkripsi data *temperature*, *humidity* untuk menghasilkan *ciphertext*. Setelahnya data yang dienkripsi tersebut digabungkan dengan nilai *salt* untuk melakukan perhitungan nilai *checksum* dengan menggunakan algoritma SHA-256. Setelah proses perhitungan menggunakan algoritma SHA-256 selesai, maka didapatkan nilai *checksum* yang akan dikirimkan bersamaan dengan keempat topik seperti berikut:

- /ESP32?IotTopic/temp
- /ESP32/IotTopic/hum
- /ESP32/IotTopic/checksum
- /ESP32/IotTopic/id

Keempat topik diatas seperti *temperature*, *humidity* yang telah dienkripsi, nilai *checksum* SHA-256m serta ID sebagai penanda unik untuk setiap data, akan dikirimkan ke *subscriber* melalui MQTT broker yaitu HiveMQ. Proses pengiriman keempat topik tersebut akan dilakukan secara berulang setiap 25 detik sekali. Implementasi kode program *publisher* disimpan dengan ekstensi file yaitu .ino dan diberikan nama filenya program_publisher_arduino_IDE yang dapat dilihat seperti gambar 4 berikut.

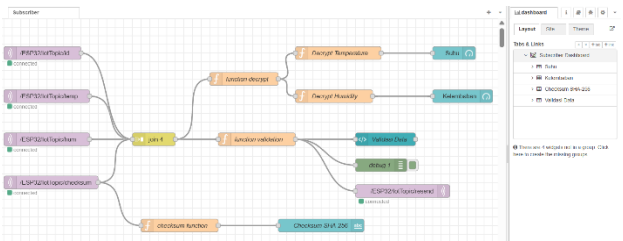
program_publisher_arduino_IDE.ino

```
#include <WiFi.h>
#include <PubSubClient.h>
#include <DHT.h>
#include <Crypto.h>
#include <SHA256.h>
#include <AESLib.h>
#include "Base64.h"

const char* ssid = "Yoon Hee";
const char* password = "ade222kp";
const char* mqtt_server = "broker.hivemq.com";
const int mqtt_port = 1883;

#define DHTPIN 4
#define DHTTYPE DHT22

DHT dht(DHTPIN, DHTTYPE);
WiFiClient espClient;
PubSubClient client(espClient);
AESLib aesLib;

unsigned long lastMsg = 0;
int dataIdCounter = 0;

String lastEncryptedTemp = "";
```

Gambar 4. Isi Implementasi kode program *Publisher*

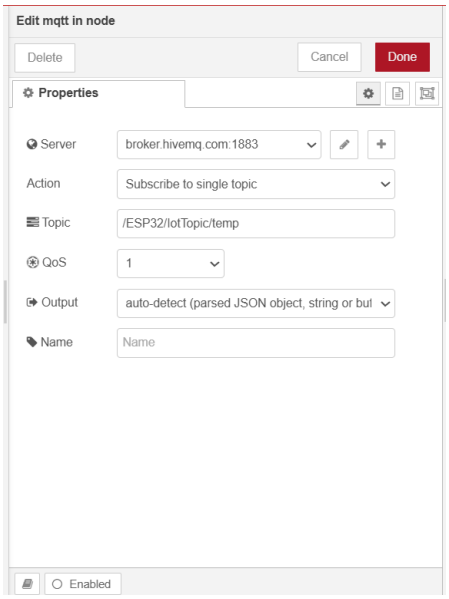
2.4.2 Implementasi Program Subscriber

Implementasi program *subscriber* dilakukan pada *software* Node-Red. Dengan menjalankan *tools* tersebut pada local komputer di CMD dengan memberikan perintah node-red untuk membuka tampilan *dashboard* pada *tools* tersebut. Seperti yang dapat dilihat pada gambar berikut.



Gambar 5. Isi *Flow Subscriber* pada *Dashboard Node-Red*

Pada gambar diatas, merupakan isi tampilan *dashboard* pada node-red yang menampilkan isi *flow subscriber*, dengan masing-masing node yang saling terhubung. Seperti node dengan warna ungu yang terdapat dua jenis yaitu MQTT *IN Nodes* dan MQTT *OUT Nodes*, MQTT *In Node* ini berfungsi untuk menerima topik-topik yang di *subscribe* dari MQTT, dan MQTT *Out Node* digunakan untuk mengirimkan data ke perangkat lain yang berlangganan topik MQTT yang sama. Seperti yang tertera pada gambar 6, melalukan konfigurasi pada *MQTT input nodes* topik *temperature*.



Gambar 6. Konfigurasi *MQTT Input Nodes* topik *Temperature*

pada gambar diatas, dilakukan konfigurasi pada MQTT *Input Nodes* dengan mengisi nama server dan *port* yang telah ditentukan untuk dapat berkomunikasi diantara *publisher*, *broker*, dan *subscriber*. Mengisi nama topik yang akan di *subscribe*, serta mengisi QoS 1 (*Quality of Service* 1). Hal yang serupa dilakukan pada konfigurasi MQTT *Input Nodes* pada topik hum, dan checksum, dengan mengisi nama server serta *port*, mengisi nama topik, dan mengisi QoS. Pada nodes dengan warna oranye yang merupakan *Function Nodes*, dimana terdapat lima nodes yaitu *Function Validation*, *Checksum Function*, *Function Decrypt*, *Decrypt Temperature*, dan *Decrypt Humidity*. Pada nodes *Function Validation*, dibuat kode program validasi dengan menghitung ulang *checksum* di sisi *subscriber* menggunakan algoritma SHA-256, yang dapat dilihat seperti gambar 7 berikut.

```
const aeskey = Buffer.from("0102030405060708091011121314151 6", "hex");
const iv = Buffer.alloc(16, 0);
const preSharedSalt = "#y0u$3cr3t54lt*1sH3R3*";

let resendData = global.get("resendData") || {};

function calculateChecksum(data) {
    let hash = crypto.createHash("sha256").update(data, "utf8").digest();

    let checksum = "";
    for (let i = 0; i < hash.length; i++) {
        checksum += hash[i].toString(16);
    }
    return checksum;
}

function decryptAES(base64Data,aeskey, iv) {
    try {
        const decipher = crypto.createDecipheriv("aes-128-cbc", aeskey, iv);
        decipher.setAutoPadding(false);
        let decrypted = decipher.update(Buffer.from(base64Data, "base64"));
        decrypted = Buffer.concat([decrypted, decipher.final()]);
        return decrypted.toString("utf8").replace(/\0/g, '').trim();
    } catch (error) {
        return null;
    }
}

let encryptedTemp = msg.payload["/ESP32/IotTopic/temp"];
let encryptedHum = msg.payload["/ESP32/IotTopic/hum"];
let receivedChecksum = msg.payload["/ESP32/IotTopic/checksum"];
let id = msg.payload["/ESP32/IotTopic/id"];
```

Gambar 7. Isi Implementasi kode program *Subscriber*

Pada gambar 7, merupakan *Function Validation* yang menerapkan algoritma SHA-256 untuk melakukan perhitungan ulang dalam menghasilkan nilai *checksum* dari topik data yang diterima seperti *encrypted temperature*, dan *encrypted humidity*, serta data *salt*. hasil nilai *checksum* yang dihitung ulang akan dibandingkan dengan nilai *checksum* yang diterima oleh *subscriber*, yang dimana ketika adanya manipulasi data maka akan memberikan hasil *output Data Manipulated*! serta proses dekripsi akan dihentikan serta *subscriber* akan melakukan *request* pengiriman ulang data ke *publisher* untuk mendapatkan data asli yang tidak dimanipulasi. Sebaliknya, ketika tidak ada manipulasi data maka akan menghasilkan *output Data Valid* dan proses dekripsi akan dilanjutkan.

2.5 Pengujian Sistem

Pengujian sistem dilakukan setelah melalui tahapan implementasi sistem. Pada tahapan ini dilakukan pengujian sistem melalui beberapa aspek yang mencakup:

1. Pengujian Fungsional pengembangan sistem perangkat IoT.

   Pengujian dilakukan pada sistem perangkat IoT seperti ESP32, dan DHT22 yang dimana prosesnya akan mengirimkan data *temperature*, *humidity* yang telah dienkripsi, serta nilai *checksum* yang dihasilkan dari perhitungan algoritma SHA-256. Proses pengiriman data akan dilakukan melalui protokol MQTT yaitu HiveMQ sebagai jalur komunikasi untuk mengirimkan data ke *subscriber*. Setelah proses data-data dikirimkan ke *subscriber*, maka data tersebut akan divalidasi untuk memverifikasi integritas data yang telah dikirimkan. Apabila validasi data yang dilakukan menyatakan valid maka proses dekripsi akan dilakukan, jika sebaliknya maka proses dekripsi akan dihentikan dan *subscriber* akan melakukan *request* pengiriman ulang data ke *publisher*. Pengujian ini dilakukan untuk mengetahui apakah sistem yang dibangun dapat memverifikasi integritas data dari proses validasi yang dilakukan di sisi *subscriber*.

2. Pengujian Efektivitas sistem keamanan yang dikembangkan melalui penerapan *checksum* SHA-256, teknik *salt*, serta

algoritma AES-128.

Pengujian dilakukan melalui simulasi serangan *data tampering* yang dilakukan secara manual, dengan menggunakan MITM *Tools*. Proses simulasi serangan *data tampering* dimulai dengan tahapan awal yaitu *ARP Spoofing* pada target korban, yang dimana memaksa target memperbarui *MAC Address* untuk menggunakan alamat *MAC* penyerang, dengan melakukan pengiriman ARP secara berulang-ulang(Olazabal et al., 2022). Ketika *MAC Address* target sama dengan yang dimiliki *MAC Address* penyerang maka proses selanjutnya adalah menggunakan MITM *Tools* untuk memantau komunikasi yang dilakukan oleh target. Selain digunakan untuk memantau komunikasi, *tools* ini digunakan juga untuk melakukan serangan *data tampering* yang dapat mengubah salah satu data yang dikirimkan oleh target. Pengujian ini dilakukan untuk mengetahui efektivitas dari sistem keamanan checksum SHA-256, teknik *salt*, serta algoritma AES-128 dari ancaman serangan *data tampering*.

3 HASIL DAN PEMBAHASAN

3.1 Hasil Pengujian Sistem

Berikut adalah hasil dari kedua pengujian sistem yang telah dilakukan.

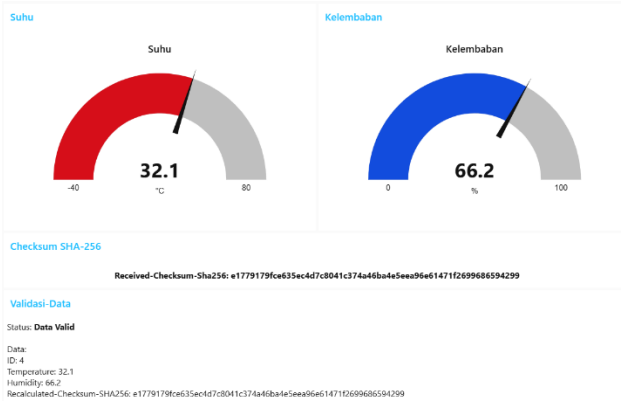
1. Pengujian Fungsional pengembangan sistem perangkat IoT.

Pada pengujian ini, terdapat dua pengujian fungsional pada sistem yang telah dibangun seperti berikut:

- Sistem perangkat IoT mampu mengirimkan sebuah data sensor *temperature*, *humidity* yang telah dienkripsi, serta nilai *checksum* SHA-256 ke *subscriber* melalui protokol MQTT. seperti yang dapat dilihat pada gambar 8, pada tampilan *Serial Monitor* arduino IDE menunjukkan *output* ID, *output temperature*, *humidity*, *encrypted temp*, *encrypted hum*, dan *hash*. Proses pada sistem perangkat IoT diawali dengan terhubung melalui *WiFi* dengan *SSID* dan *Password* yang telah ditentukan. Setelah terkoneksi ke jaringan WiFi, selanjutnya sistem perangkat IoT melakukan koneksi ke broker MQTT yaitu HiveMQ untuk dapat melakukan pengiriman data. Untuk data yang dikirimkan hanya mencakup ID, *Encrypted Temp*, *Encrypted Hum* yang telah dienkripsi, dan nilai checksum SHA-256 secara bersamaan ke *subscriber*.

Gambar 8. *Publisher* mengirimkan data ke *Subscriber* melalui protokol *MQTT*

- Sistem keamanan di sisi *subscriber* mampu melakukan validasi data untuk memverifikasi integritas data dari setiap data-data yang dikirimkan oleh *publisher*. Seperti yang dapat dilihat pada gambar 9, pada *subscriber dashboard* menunjukkan data yang telah dikirimkan oleh *publisher* sebelumnya, diterima oleh *subscriber*. Dan dari setiap data-data yang dikirimkan tersebut, di validasi menggunakan algoritma SHA-256 dengan membandingkan nilai checksum yang dihitung ulang dengan nilai checksum yang diterima oleh subscriber. pada gambar tersebut menunjukkan, dari hasil validasi data yang dilakukan menginformasikan *output Data Valid* yang dimana berarti tidak ada manipulasi data saat pengiriman data dari *publisher* ke *subscriber* berlangsung, dan proses dekripsi dapat dilanjutkan ketika data yang dikirimkan tersebut dinyatakan valid serta menampilkan hasil validasi data seperti yang ditunjukkan pada gambar 9 berikut.



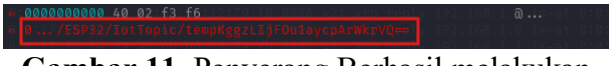
Gambar 9. Data yang diterima dan divalidasi di sisi *Subscriber*

2. Pengujian Efektivitas sistem keamanan yang dikembangkan melalui penerapan *checksum* SHA-256, teknik *salt*, serta algoritma AES-128.

   Pada tahap pengujian ini, diawali dengan melakukan serangan *Arp Spoofing* yang merupakan salah satu teknik dari tipe serangan MITM (Olazabal et al., 2022). Setelah melalui proses tahapan *Arp spoofing*, maka dilanjutkan dengan menggunakan *MITM Tools* untuk dapat melakukan manipulasi data, yang dapat dilihat pada gambar berikut.

Gambar 10. *Flows Panel* yang menunjukkan Koneksi *TCP* ke *Broker HiveMQ*

Pada gambar diatas merupakan *flows panel* pada *tools MITM Proxy*, yang berisi daftar lalu lintas yang berhasil di *intercept*. Dalam panel tersebut, terdapat satu koneksi TCP yang berhasil dicegat. Yang dimana komunikasi tersebut adalah milik target yang dapat dilihat dari Alamat IP tujuan dan *port*nya yaitu 52.59.167.5:1883 yang merupakan server broker MQTT yaitu HiveMQ. Pada satu koneksi TCP yang terdapat pada *flow panel* tersebut, berisi topik yang dikirimkan *publisher* ke broker MQTT yang berhasil di *intercept* oleh penyerang seperti yang dapat dilihat pada gambar 11.



Gambar 11. Penyerang Berhasil melakukan *Intercept*

Gambar diatas menunjukkan, penyerang berhasil melakukan *intercept* pada salah satu topik yang dikirimkan oleh target seperti yang ditunjukkan pada gambar diatas, tahap selanjutnya penyerang akan melakukan manipulasi data dengan mengubah data *temperature* yang telah terenkripsi tersebut. seperti yang dapat dilihat pada gambar 12 berikut.

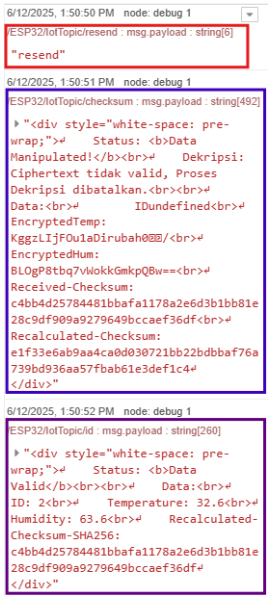
Gambar 12. Topik IoT yang dikirimkan oleh target berhasil dimanipulasi oleh penyerang

Pada gambar diatas, menunjukkan penyerang telah berhasil melakukan serangan *data tampering* menggunakan *tools MITM Proxy*. Disini data *temperature* yang terenkripsi tersebut, berhasil diubah yang sebelumnya nilai *ciphertext*-nya seperti KggzLIjF0u1aycpArWkrVQ== menjadi KggzLIjF0u1aDirubah, yang diakibatkan dari manipulasi data yang dilakukan oleh penyerang. Tetapi dengan adanya sistem keamanan Validasi Data yang dilakukan di sisi *subscriber*, dapat mencegah hal tersebut. Seperti yang dapat dilihat pada gambar 13 berikut.

Gambar 13. Serangan *Data Tampering* yang dilakukan secara manual dapat dideteksi dan dicegah dari validasi data yang dilakukan di *Subscriber*

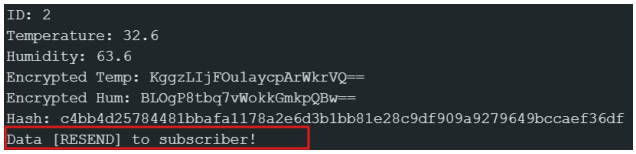
Pada gambar diatas menunjukkan, sistem keamanan yang telah dibangun di sisi *subscriber* tersebut dapat mendeteksi serta mencegah serangan *data tampering* yang dilakukan secara manual, dengan menghentikan proses dekripsi untuk tidak dilanjutkan dan diteruskan ke *subscriber*. Dari Validasi Data menunjukkan status *Data Manipulated*! Dengan rincian data *Encryptedtemp* KggzLIjFOu1aDirubah0/ (data yang diubah oleh penyerang), *Encryptedhum* BLOgP8tbq7vWokkGmkpQBw== , serta data *Received Checksum* dan data *Recalculated Checksum*. Dan juga *subscriber* secara langsung

melakukan *request* pengiriman ulang data ke *publisher* ketika terjadi manipulasi data. seperti yang dapat dilihat pada gambar 14 berikut.



Gambar 14. *Log Data* pada *Subscriber*

Gambar diatas menunjukkan, *subscriber* telah berhasil menerima ulang data yang telah di *request* sebelumnya ke *publisher*. Dapat dilihat dari tanda garis berwarna merah, *subscriber* secara langsung mengirimkan sebuah *payload* yaitu *resend*, ketika terjadi manipulasi data seperti pada tanda garis berwaran biru. Lalu *subscriber* menerima data yang telah di *request* yang dapat dilihat pada tanda garis berwarna ungu. Walaupun terjadi manipulasi data yang diakibatkan dari serangan *data tampering*, sistem pada *subscriber* dapat menerima kembali data asli secara utuh dari *publisher*. Seperti yang dapat dilihat pada gambar 15, *publisher* melakukan *resend data* yang telah di *request* sebelumnya oleh *subscriber*.



Gambar 15. *Log Serial Monitor Publisher*

Gambar diatas menunjukkan, *publisher* telah melakukan *resend data* yang asli kembali ke *subscriber* berdasarkan data terakhir atau data yang di *request* oleh *subscriber*.

3.2 Evaluasi Hasil Pengujian Sistem

Evaluasi hasil pengujian sistem dilakukan setelah melewati tahap pengujian fungsional dan pengujian efektivitas. Berikut adalah hasil analisis data pengujiannya.

1. Pengujian Fungsional pengembangan sistem perangkat IoT.

Hasil pengujian fungsional pengembangan sistem perangkat IoT dapat dilihat sebagai berikut.

Table 1. Pengujian Fungsional

No	Pengujian Fungsional	Hasil Pengujian	
		Berhasil (✓)	Gagal (✗)
1	Sistem perangkat IoT mampu mengirimkan sebuah data sensor *temperature*, *humidity* yang telah dienkripsi, serta nilai checksum SHA-256 ke *subscriber* melalui protokol MQTT.	Berhasil (✓)	
2	Sistem keamanan di sisi *subscriber* mampu melakukan Validasi Data untuk memverifikasi integritas data dari setiap data-data yang dikirimkan oleh *publisher*.	Berhasil (✓)	

Pada tabel diatas, menunjukkan hasil pengujian fungsional pengembangan sistem perangkat IoT telah berhasil dilakukan. Sistem perangkat IoT dapat mengirimkan sebuah data sensor seperti *temperature*, *humidity*, yang telah dienkripsi, serta nilai *checksum* SHA-256 ke *subscriber* melalui protokol MQTT yang digunakan yaitu HiveMQ. Dan sistem keamanan di sisi *subscriber* mampu melakukan Validasi Data untuk memverifikasi integritas data dari setiap data-data yang dikirimkan oleh *publisher*.

2. Pengujian Efektivitas sistem keamanan yang dikembangkan melalui penerapan *checksum* SHA-256, teknik *salt*, serta algoritma AES-128.

Pada pengujian ini, dilakukan pengiriman topik data *temperature*, *humidity*, dan *checksum* secara bersamaan sebanyak 50 kali ke *subscriber*, dengan total jumlah pengiriman topik datanya sebanyak 150 dari topik *temperature*,

humidity, dan *checksum*. Dari jumlah topik data yang dikirimkan ke *subscriber* tersebut, dilakukan 20 kali serangan *data tampering* pada topik data *temperature*. Dari hasil pengujian ini, didapatkan rata-rata persentase keberhasilan pengujian dari sistem yang telah dibangun. Yang dapat dilihat sebagai berikut:

- Hasil perhitungan rata-rata pada keberhasilan deteksi *tampering*, dilakukan dengan membagi jumlah serangan *data tampering* dengan jumlah pada sistem keamanan yang berhasil mendeteksi dan mencegah serangan *data tampering*. seperti perhitungan berikut: $\frac{20}{20} = 1$. Hasil perhitungan menunjukkan sistem keamanan di *subscriber* 100% berhasil mendeteksi dan mencegah semua serangan *data tampering* yang dilakukan sebanyak 20 kali.
- Hasil perhitungan rata-rata dari data yang diterima oleh *subscriber*. Dari jumlah keseluruhan topik yang dikirimkan adalah 150 pengiriman. Dengan kegagalan data yang tidak diterima oleh *subscriber* berjumlah 9 topik, dari 3 pengiriman data yaitu pada data ke 17, 21, dan 31. Dan sisanya yang berjumlah 141 topik yang berhasil diterima oleh *subscriber*. Berikut adalah perhitungannya: $\frac{topik\ data\ yang\ diterima}{jumlah\ topik\ data\ yang\ dikirimkan} = \frac{141}{150} = 0.94$ atau 94% untuk nilai rata-rata Tingkat keberhasil data yang diterima oleh *subscriber*.

3.3 Perbandingan Dengan Penelitian Lain

Perbandingan dengan penelitian lain, yang terdapat beberapa perbedaan mendasar. Seperti pendekatan autentikasi menggunakan JSON Web Token (JWT) yang dikombinasikan dengn TLS(Mangkurat, n.d.) mampu memberikan lapisan keamanan yang lebih kuat terutama dalam autentikasi pengguna dan enkripsi pada komunikasi saat transmisi data dilakukan, namun memiliki keterbatasan dalam hal konsumsi sumber daya yang cukup tinggi pada perangkat dengan kapasitas terbatas(Gerez et al., 2019).

Penelitian lainnya yang menggunakan *honey sensor* efektif dalam mendeteksi adanya manipulasi data melalui penyisipan data palsu atau *manipulate data*, tetapi pendekatan ini hanya bersifat deteksi dan tidak secara langsung mencegah serangan(Kumar Shrivastava et al., 2019).

Adapun pada penelitian dengan judul Implementasi Mekanisme *end-to-end security* pada IoT *Middleware* . Penelitian ini menggunakan pendekatan *event-driven* yang mampu mendukung interoperabilitas berbagai macam sensor pada lingkungan IoT. Penggunaan prokol keamanan TLS/SSL digunakan untuk menutup celah keamanan pada skema komunikasi *middleware*, menawarkan jaminan kerahasiaan yang lebih tinggi. Tetapi hasil penelitan menunjukkan mekanisme TLS/SSl terbatas dalam penerapan, terlebih pada perangkat dengan keterbatasan sumber daya(Pramukantoro et al., 2019).

Perbandingan dengan sistem yang dilakukan pada penelitian ini menawarkan keseimbangan dengan mengkombinasikan checksum SHA-256, teknik salt, dan AES-128 sehingga dapat menjaga integritas sekaligus kerahasiaan data. Sistem keamanan yang diterapkan pada *publisher*, dan terutama di *subscriber* yang menerapkan validasi data untuk menjaga integritas data, ketika data sampai di *subscriber*. Sistem yang dibuat memastikan jika data telah dirubah/dimanipulasi disaat proses transmisi data dilakukan, secara langsung data tersebut tidak diberikan ke pada *subscriber*, melainkan proses sistem keamanan di *subscriber* akan melakukan permintaan ulang ke *publisher* terkait data yang telah dimanipulasi, yang memastikan meskipun adanya perubahan data/manipulasi data sistem keamanan pada *subscriber* menjamin data yang sampai pada *subscriber* adalah data yang asli tanpa adanya data yang dirusak akibat manipulasi data.

4 PENUTUP

4.1. Kesimpulan

Berdasarkan hasil dari penelitian ini, Sistem keamanan yang dibangun mampu mengirimkan sebuah data sensor *temperature*, *humidity* yang telah dienkripsi, serta nilai checksum SHA-256 ke *subscriber* melalui protokol MQTT. Sistem keamanan di sisi *subscriber* mampu melakukan Validasi Data untuk memverifikasi integeritas data dari setiap data-data yang dikirimkan oleh *publisher*. Dan juga sistem keamanan yang dibangun berhasil mendeteksi dan mencegah seluruh jumlah serangan *data tampering* melalui

pengujian yang telah dilakukan. Dengan nilai rata-rata hasilnya adalah $\frac{20}{20} = 1$ atau bila dalam bentuk persentase adalah 100%. Serta nilai rata-rata untuk data yang berhasil diterima oleh *subscriber* adalah 94%, dari seluruh pengiriman topik data selama pengujian berlangsung.

4.2. Saran

Namun, sistem keamanan yang dibangun ini masih belum sempurna, karena hanya mampu mencegah serangan di sisi *subscriber*. Untuk dapat sepenuhnya mencegah serangan *data tampering* diperlukan sistem keamanan tambahan seperti TLS, SSL ataupun menggunakan *port* 8883 pada MQTT agar komunikasi dilakukan secara privasi.

Berdasarkan hasil uji dari sistem keamanan di *subscriber*, terdapat beberapa data yang tidak diterima subscriber yang terjadi pada pengiriman data ke 17, 21, dan 31. Maka dari itu, diperlukan sistem konfirmasi penerimaan data pada *subscriber*, agar *publisher* dapat mengetahui data yang dikirim ulang ke *subscriber* telah diterima. Dan juga sistem *resend* atau kirim ulang data pada *publisher* dapat membatasi jumlah *resend data* misalnya tiga kali pengiriman ulang data jika belum terdapat konfirmasi dari *subscriber*.

Pada sistem keamanan di *subscriber* memberikan informasi terkait adanya manipulasi data, dengan menampilkan dashboard yang berisikan informasi *data valid* atau *data manipulated*, tetapi ini belum cukup karna *subscriber* hanya mengetahui ketika membuka *dashboard* tersebut melalui *node-red*. Dibutuhkan pemberian informasi melewati notifikasi agar subscriber secara langsung dapat mengetahui yang terjadi.

Penggunaan metode *checksum SHA-256* dan enkripsi *AES-128* menambah beban komputasi pada perangkat IoT dengan sumber daya terbatas seperti ESP32. *SHA-256* membutuhkan proses kompresi 64 putaran(Anugrah et al., 2019) dengan menggunakan *message schedule* serta parameter yang telah ditentukan dalam algoritma *SHA-256*.

5 DAFTAR PUSTAKA

Al-Mashhadani, M., & Shujaa, M. (2022). IoT Security Using AES Encryption Technology based ESP32 Platform. *International Arab Journal of Information Technology*, *19*(2), 214–223. https://doi.org/10.34028/iajit/19/2/8

Anugrah, Y., Hannats, M., Ichsan, H., & Kusyanti, A. (2019). *Implementasi Algoritme SHA-256 Menggunakan Protokol MQTT pada Budidaya Ikan Hias* (Vol. 3, Issue 4). http://j-ptiik.ub.ac.id

Gerez, A. H., Kamaraj, K., Nofal, R., Liu, Y., & Dezfouli, B. (2019). Energy and Processing Demand Analysis of TLS Protocol in Internet of Things Applications. *IEEE Workshop on Signal Processing Systems, SiPS: Design and Implementation*, *2018-October*, 312–317. https://doi.org/10.1109/SiPS.2018.8598334

Hue, A., Sharma, G., & Dricot, J. M. (2022). Privacy-enhanced mqtt protocol for massive iot. *Electronics (Switzerland)*, *11*(1). https://doi.org/10.3390/electronics11010070

Kelechi, A. H., Alsharif, M. H., Agbaetuo, C., Ubadike, O., Aligbe, A., Uthansakul, P., Kannadasan, R., & Aly, A. A. (2021). Design of a low-cost air quality monitoring system using arduino and thingspeak. *Computers, Materials and Continua*, *70*(1), 151–169. https://doi.org/10.32604/cmc.2022.019431

Khudhur, D. D., & Croock, M. S. (2021). Developed security and privacy algorithms for cyber physical system. *International Journal of Electrical and Computer Engineering*, *11*(6), 5379–5389. https://doi.org/10.11591/ijece.v11i6.pp5379-5389

Kumar Shrivastava, R., Mishra, S., E, A. V, Hota, C., & Member, S. (2019). Preventing data tampering in IoT networks. In *2019 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*.

Mangkurat, C. A. (n.d.). *RANCANG BANGUN SISTIM KEAMANAN PERANGKAT IoT DENGAN METODE AUTENTIKASI MENGGUNAKAN JSON WEB TOKEN PADA PROTOKOL MQTT*.

Marco TORCHIANO Doc Maximilian HILS, S. (2024). *Enhancing Network Interception with Mitmproxy An Open Source Solution*

for Transparent Proxy Mode on macOS and Linux Candidate Emanuele MICHELETTI.

Meylan, A., Cherubini, M., Chapuis, B., Humbert, M., Bilogrevic, I., & Huguenin, K. (2020). A Study on the Use of Checksums for Integrity Verification of Web Downloads. *ACM Transactions on Privacy and Security*, *24*(1). https://doi.org/10.1145/3410154

Nikolov Neven, & Nakov Ognyan. (2019). *Developed security and privacy algorithms for cyber physical system*. IEEE.

Olazabal, A. A., Kaur, J., & Yeboah-Ofori, A. (2022). Deploying Man-In-the-Middle Attack on IoT Devices Connected to Long Range Wide Area Networks (LoRaWAN). *ISC2 2022 - 8th IEEE International Smart Cities Conference*. https://doi.org/10.1109/ISC255366.2022.9922377

Pramukantoro, E. S., Andri Bakhtiar, F., Lutfi, A., Aji, B., Prasetya Dewa, D. H., & Komputer, F. I. (2019). *IMPLEMENTASI MEKANISME END-TO-END SECURITY PADA IoT MIDDLEWARE IMPLEMENTATION OF END TO END SECURITY IN IOT MIDDLEWARE*. *6*(3). https://doi.org/10.25126/jtiik.201961401

Rachmadini, D., & Puspasari, I. (2020). Implementasi dan Analisis Fitur Keamanan Protokol MQTT pada Telehealthcare. *Journal of Technology and Informatics (JoTI)*, *2*(1).

Rathod, U., Sonkar, M., & Chandavarkar, B. R. (2020, July 1). An Experimental Evaluation on the Dependency between One-Way Hash Functions and Salt. *2020 11th International Conference on Computing, Communication and Networking Technologies, ICCCNT 2020*. https://doi.org/10.1109/ICCCNT49239.2020.9225503

TLP:WHITE Compromise of U.S. Water Treatment Facility SUMMARY. (2021). www.fbi.gov/contact-us/field,