Jurnal Tugas Akhir Fakultas Informatika Februari - 2021

Penerapan Keamanan Komunikasi pada Jaringan LoRa(*Long Range*) Menggunakan Algoritma *Advanced Encryption Standard*(AES) dan *Message Authentication Code*(MAC)

Putri Apriyanti Windya[1], Vera Suryani[2], Aulia Arif Wardana[3]

[1,2,3]Fakultas Informatika, Universitas Telkom, Bandung
[1]putriawindya@students.telkomuniversity.ac.id, [2] verasuryani@telkomuniversity.ac.id,
[3]auliawardan@telkomuniversity.ac.id

Abstrak

***Internet of Things* (IoT) merupakan suatu hal yang populer saat ini. Penggunaan IoT semakin meningkat setiap tahun khususnya penggunaan LoRa, begitu juga dengan pengembangan perangkat IoT. Salah satu karakteristik perangkat IoT yaitu *resource* yang terbatas. Perangkat ini sering disebut sebagai *constrained device* IoT. Seiring dengan meningkatnya penggunaan LoRa, aspek keamanan komunikasi pada jaringan LoRa juga harus diperhatikan. Akan tetapi, keterbatasan *resource* yang dimiliki oleh perangkat IoT menjadi tantangan dalam memilih metode *security* yang sesuai. Oleh karena itu, untuk mengatasi masalah tersebut dibutuhkan sebuah metode *security* yang sesuai yaitu pemanfaatan algoritma AES dan MAC. Jenis AES yang digunakan pada penelitian ini yaitu AES128 dan AES256. Sedangkan Algoritma MAC yang digunakan adalah *Hash-based Message Authentication Code* (HMAC). Berdasarkan hasil analisis keamanan yang telah dilakukan, metode ini mampu menjamin aspek *confidentiality, integrity* dan *authentication.* Selain itu, penelitian ini juga melakukan analisis *overhead* pada *constrained devices* IoT kelas 0 dan kelas 2. Hasil analisis *overhead* menunjukkan bahwa metode ini cocok untuk diterapkan pada kelas 0 dan kelas 2.**

Kata kunci : *Long Range, constrained devices, CIA, AES, HMAC*

Abstract

Internet of Things (IoT) is a popular thing nowadays. The use of IoT is increasing every year especially the use of LoRa, as well as the development of IoT *device*s. One of the characteristics of IoT *device*s is limited resources. This *device* is often referred to as IoT-constrained *device*s. Along with the increasing use of LoRa, the communication security aspect of the LoRa network must also be considered. However, the limited resources possessed by IoT *device*s are a challenge in choosing the appropriate security method. Therefore, to overcome this problem an appropriate security method is needed namely the use of AES algorithms and MAC. Variants of AES algorithm used in this research are AES128 and AES256. Meanwhile, the MAC algorithm used is Hash-based Message Authentication Code (HMAC). Based on the results of the security analysis that has been done, this methode is able to guarantee aspects of confidentiality, integrity and authentication. In addition, this study also performs overhead analysis on IoT constrained *device*s class 0 and class 2. The results of the overhead analysis show that this method is suitable to be implemented on IoT constrained *device*s class 0 and class 2.

Keywords: *Long Range, constrained devices, CIA, AES, HMAC*

1. Pendahuluan

Latar Belakang

Seiring berjalannya waktu, penggunaan *Internet of Things* (IoT) dalam kehidupan sehari-hari semakin meningkat. *Internet of Things* (IoT) merupakan sebuah paradigma dimana segala sesuatu yang dikategorikan sebagai "*Things*" saling terhubung dan berkomunikasi melalui jaringan internet. "*Things*" yang dimaksudkan pada IoT merupakan segala sesuatu yang dapat melakukan atau memiliki *sensing* atau aktuasi serta memilki kemampuan untuk diprogram. Melalui eksploitasi dari "*Things*" pada IoT, informasi dapat dikumpulkan serta perubahan dapat terjadi kapan saja dan dimana saja[1]. Gartner Inc. meramalkan bahwa Penggunaan IoT akan meningkat menjadi 5,8 miliar *endpoint* pada akhir tahun 2020. Peningkatan tersebut terjadi mulai dari *smart home, smart city* hingga *Industrial* IoT[2]. Selain itu, Statista Research Department juga memprediksi bahwa pada tahun 2025, penggunaan IoT akan mencapai angka 1,6 triliyun[3].

Long Range(LoRa) merupakan salah satu jenis *wireless connectivity* yang populer pada jaringan IoT[4]. LoRa memiliki jangkauan frekuensi radio sekitar 15 Km. LoRa memiliki beberapa *frequency plan* yang telah ditetapkan yaitu mulai dari 433MHz hingga 925MHz sesuai dengan negara yang ditempati[5]. Pada IoT, LoRa dapat diaplikasikan pada *smart city, smart parking, streetlight control,* dan lainnya[6][7]. Semakin banyak sektor yang dapat disisipi oleh LoRa juga berdampak pada bervariasinya pengembangan perangkat atau *device* pada IoT. Salah satu karakteristik perangkat IoT adalah terbatasnya *resource* yang disediakan oleh perangkat tersebut

atau sering dikenal sebagai *constrained device.* Keterbatasan *resource* dapat berupa terbatasnya ukuran memori, keterbatasan daya serta keterbatasan *code complexity.* Hal ini menyebabkan terbatasnya kemampuan perangkat untuk dieksplorasi[8]. Akan tetapi, meningkatnya penggunaan LoRa juga berdampak pada keamanan komunikasi pada LoRa. Tak jarang ditemui ancaman terhadap *security* pada LoRa ketika digunakan. Ancaman tersebut dapat berupa *spoofing identity*, *tempering with data*, *information disclosure* dan sebagainya[9]. Sehingga hal tersebut mengharuskan pengguna atau pengembang untuk senantiasa meningkatkan aspek *security* pada penggunaan LoRa.

Keterbatasan *resource* pada perangkat IoT juga menjadi tantangan dalam pemilihan metode *security* yang sesuai. Salah satunya harus sesuai dengan kapasitas penyimpanan dan komputasi yang tersedia pada *constrained device.* Selain itu, parameter *security* dalam IoT juga harus tercapai. *Security* parameter dapat berupa *integrity, authentication* dan *confidentiality*[10]. Oleh karena itu, dibutuhkan sebuah metode keamanan komunikasi untuk menangani masalah. Adapun metode yang digunakan yaitu pemanfaatan algoritma AES dan MAC untuk mengamakan data selama proses transmisi. *Advanced Encryption Standard*(AES) adalah algoritma enkripsi simetris yang dikembangkan oleh *National Institute of Standard and Technology* (NIST) pada tahun 2001. AES memiliki tiga ukuran key yang berbeda yaitu 128 bits, 192 bits dan 256 bits[11]. AES dikenal cocok untuk diterapkan pada perangkat IoT yang memiliki sumber daya terbatas[12][13]. *Message Authentication Code*(MAC) adalah sebuah prosedur untuk memverifikasi bahwa pesan yang diterima berasal dari sumber yang otentik[11, p. 383]. Salah satu algoritma MAC yang sering digunakan yaitu *Hash-based Message Authentication Code*(HMAC). HMAC adalah jenis MAC yang menggunakan *cryptographic hash function.* HMAC dapat digunakan dengan *cryptographic hash function* iteratif apa pun misalnya MD5, SHA1, SHA128, SHA256 maupun SHA512[14].

Topik dan Batasannya

Berdasarkan latar belakang masalah yang telah dipaparkan, maka rumusan masalah yang diangkat oleh penulis pada penelitian ini yaitu bagaimana cara meningkatkan keamanan komunikasi pada jaringan *Long Range*(LoRa) menggunakan algoritma *Advanced Encryption Standard*(AES) dan Message Authentication Code(MAC) dengan mempertimbangkan keterbatasan sumber daya pada perangkat IoT yang digunakan serta bagaimana *security analysis* dan *overhead analysis* dari penerapan algoritma AES dan MAC pada penelitian ini. Pada proses enkripsi dan dekripsi pesan, jenis algoritma AES yang digunakan yaitu AES128 dan AES256. Untuk proses MAC, algoritma MAC yang digunakan yaitu *Hash-based Message Authentication Code*(HMAC). Variasi HMAC yang digunakan yaitu HMAC-MD5, HMAC-SHA1, HMAC-SHA256, dan HMAC-SHA512. Metode yang digunakan pada penelitian ini diterapkan pada *constrained devices* IoT kelas 0 dan 2. Adapun parameter keamanan yang diukur yaitu *confidentiality, integrity* dan *Authentication. Overhead analysis* pada penelitian ini diukur dari segi penggunaan *Flash,* penggunaan memori, lama waktu enkripsi dan dekripsi pesan, waktu tunggu kedatangan pesan selanjutnya, lama waktu proses generate MAC serta waktu yang butuhkan untuk verifikasi MAC.

Tujuan

Tugas akhir ini bertujuan untuk meningkatkan keamanan komunikasi pada jaringan *Long Range*(LoRa) dengan mempertimbangkan parameter kemananan yaitu *confidentiality, integrity* dan *authentication.* Adapun keberhasilan parameter *confidentiality* dilihat dari apakah *plaintext* pesan yang dikirim dapat dilihat oleh penyerang atau tidak. Sedangkan keberhasilan parameter *authentication* dan *integrity* dibuktikan dengan kecocokan nilai MAC antara pengirim dan penerima pada proses verifikasi MAC. Selain itu, penelitian ini juga bertujuan untuk mengetahui variasi metode yang diusulkan sesuai atau tidak untuk diterapkan pada *constrained devices* IoT kelas 0 dan 2 melalui proses *overhead analysis.*

Organisasi Tulisan

Buku tugas akhir ini disusun menjadi lima bagian yaitu Bab 1 – Pendahuluan yang berisikan latar belakang, topik dan batasannya, serta tujuan penelitian. Bab 2 – Studi terkait berisi studi literatur atau penelitian terdahulu yang dijadikan sebagai acuan dalam penelitian ini. Bab 3 – Perancangan Sistem, bab ini berisi alur sistem yang dikerjakan pada penelitian ini. Bab 4 – Implementasi dan Evaluasi menjelaskan tentang pengimplementasian sistem dan analisis dari program yang telah dibuat. Bab 5 – Penutup yang berisi kesimpulan dari hasil analisis dan saran.

2. Studi Terkait

2.1. Penelitian Terkait

Terdapat beberapa penelitan terkait metode penerapan algoritma AES, MAC, dan keamanan *constrained Device* IoT pada jaringan LoRa yang dijadikan sebagai acuan dan bahan kajian pada penelitian ini. Pada penelitian yang berjudul *Securing Communication in Class-0 IOT Devices*, Kavita Mital [15] menerapkan

algoritma AES128 untuk mengamankan data selama proses transmisi melalui jaringan IoT. Penelitian ini membuktikan bahwa data hanya dapat diakses oleh penerima yang memiliki *secret key* saja.

K. Tsai, dkk[12] pada penelitiannya yang berjudul *AES-128 Based Secure Low Power Communication for LoRaWAN IoT Environments*, menerapkan Algoritma AES128 untuk mengamankan komunikasi pada jaringan LoRaWAN. Penulis melakukan simplifikasi algoritma AES untuk mengurangi konsumsi daya. Penilitian ini terbutki dapat mengurangi konsumsi daya sebesar 26,2% pada saat proses enkripsi. Selain itu , metode yang diusulkan juga dapat menahan tiga serangan, termasuk serangan *known-key*, *replay attack*, and *eavesdropping attack*, dan secara praktis berguna untuk digunakan di lingkungan IoT LoRaWAN.

Amjad Iqbal, dkk[16] melakukan penelitian untuk mengamankan system komunikasi *Remote Micro-grids* pada perangkat LoRa ESP32 menggunakan algoritma AES. Pada penelitian ini, selain menerapkan algoritma AES, peneliti juga menerapkan MAC sebagai penjamin aspek *authentication* pada *smart grid* yang dibangun. Hasil akhir dari penelitian ini menunjukkan bahwa aspek *confidentiality* dan *authentication* telah berhasil dicapai. Selain itu, penulis mengungkapkan bahwa konsumsi energi hanya berkisar 5mW.

A. Fauzan, dkk [17] melakukan analisis *overhead* terhadap penerapan metode *digital signature* untuk protocol MQTT pada tiga kelas *constrained devices* IoT. Peneliti menggunakan algoritma AES128, AES192, dan AES256 pada proses enkripsi data. Dari hasil penelitian, peneliti membuktikan bahwa algoritma AES dapat diterapkan pada ketiga kelas *constrained devices* IoT.

S.M. Suhail Hussain, dkk[18] dalam penelitiannya menerapkan algoritma *Hash-based Message Authentication Code* (HMAC) dan *Advanced Encryption Standard - Galois Message Authentication Code* (AES-GMAC) untuk mengamankan *Generic ObjectOriented Substation Events* (GOOSE) *messages*. Peneliti menggunakan metode ini karena metode *digital signature* memiliki nilai komputasi yang sangat tinggi, sehingga sangat sulit untuk diterapkan pada GOOSE. Hasil dari penelitian ini menunjukkan bahwa metode yang digunakan dapat menjamin keamanan dari GOOSE dibuktikan dengan tidak terlihatnya *plaintext* ketika data dicapture menggunakan wireshark.

Berdasarkan peneltian terkait yang telah dikerjakan sebelumnya, penelitian ini menggunakan penelitian [12], [15], [17] sebagai referensi dalam penerapan AES pada LoRa, karena pada penelitian [12] AES terbukti dapat diterapkan pada jaringan LoRA dan pada dua penelitian lainnya penerapan AES terbukti cocok untuk diterapkan pada *constrained devices* IoT . Untuk menjamin aspek *authentication*, Penelitian ini menggunakan penelitian [16], [18] sebagai referensi karena kedua penelitian tersebut dapat menjamin aspek *authentication* dari pesan yang dikirimkan menggunakan modul LoRa.

2.2. Dasar Teori

2.2.1. *Constrained Devices*

Internet of Things memiliki perangkat atau *device* yang sangat bervariasi. Apabila dilihat dari segi sumber daya (*resource*), perangkat IoT terbagi menjadi tiga kategori yaitu *high-end*, *middle-end* dan *Low-end* IoT *device*. *High-end* IoT *device* merupakan salah satu bagian dari *Single Board Computer* (SBC) yang memiliki sumber daya (*resource*) yang cukup, seperti memiliki *processing unit* yang kuat, memiliki RAM dengan kapasitas yang besar serta memiliki penyimpanan internal yang cukup besar sehingga mampu menjalankan sistem operasi seperti linux dan windows 10 IoT core. Salah satu perangkat *high-end* yang terkenal adalah Raspberry Pi. Sedangkan *middle-end* IoT *device* merupakan perangkat IoT yang memiliki sumber daya lebih kecil daripada *high-end* IoT *device* dimana kapasitas RAM sudah mencapai ratusan Kilobyte. Salah satu perangkat *middle-end* yaitu Arduino yun. Dan *low-end* IoT *device* merupakan perangkat IoT yang sangat terbatas dari segi sumber daya. *Low-end device* tidak dapat menjalankan sistem operasi seperti yang mampu dijalankan oleh *high-end* dan *middle-end* IoT *device* [19], [20]. Menurut *Internet Engineering Task Force* (IETF) pada RFC7228, *low-end device* IoT atau yang dikenal sebagai *constrained devices* adalah perangkat yang memiliki keterbatasan pada RAM, Flash serta keterbatasan daya. Terdapat tiga kelas pada *constrained device* IoT yaitu *class 0, class 1* dan *class 2. Class 0* (C0) merupakan *device* yang memiliki *resource* yang paling kecil. C0 memiliki kapasitas RAM << 10 kiB dan *flash* << 100 kiB. *Class 1* (C1) memiliki sumber daya lebih besar dari C0. *Device* C1 memiliki kapasitas RAM sebesar 10 kiB dan *flash* sebesar 100 kiB. *Class 2* (C2) memiliki sumber daya paling besar daripada dua kelas sebelumnya. *Device* C2 memiliki kapasitas RAM sebesar 50 kiB dan *flash* sebesar 250 kiB[8].

2.2.2. *Advanced Encryption Standard* (AES)

Pada tahun 2001, *National Institute of Standard and Technology*(NIST) menetapkan sebuah standard kriftografi baru yaitu *Advanced Encryption Standar* atau lebih dikenal dengan AES. AES beroperasi pada blok 128-bit atau 16 karakter. AES memiliki tiga variasi panjang kunci(*key size*) yaitu 128 bit, 192 bit dan 256 bit. AES bersifat non-Fiestel karena AES selalu memiliki invers dengan panjang 128-bit. AES juga menggunakan kunci berulang yang dikenal sebagai ronde. Adapun yang diulang berkali-kali pada algoritma AES ini

merupakan *state* dimana *state* ini akan menjadi *output* ronde k dan menjadi *input* pada ronde selanjutnya (k + 1) [11], [21]. Terdapat 4 jenis transformasi pada algoritma AES, yaitu:

1. *SubBytes*, sebagai transformasi subsitusi.
2. *ShiftRows*, sebagai transformasi permutasi.
3. *MixColumn*, sebagai transformasi pengacakan.
4. *AddRoundKey*, sebagai transformasi penambahan key.

Algoritma AES pada tugas akhir ini berperan sebagai algoritma enkripsi yang berfungsi untuk melakukan enkripsi terhadap pesan yang dikirim kepada penerima. Variasi AES yang digunakan adalah AES128 dan AES256.

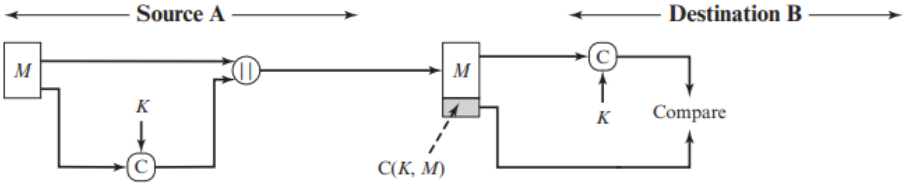
2.2.3. *Hash-based Message Authentication Code* (HMAC)

Message Authentication Code(MAC) adalah sebuah prosedur untuk memverifikasi bahwa pesan yang diterima berasal dari sumber yang otentik[11, p. 383]. MAC mengasumsikan bahwa dua entitas yang saling berkomunikasi akan berbagi *secret key* K yang sama. Ketika pengirim akan mengirim pesan kepada penerima, maka pengirim akan membangkitkan nilai MAC menggunakan fungsi[11, p. 388]:

$$\boldsymbol{MAC = C(K, M)} \quad (1)$$

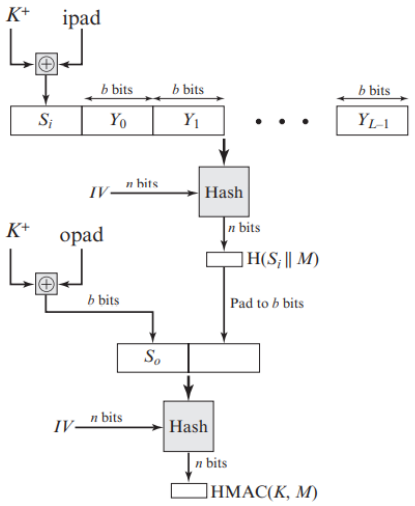
M = *Input message*
K = Kunci rahasia (*secret key*)
C = Fungsi MAC
MAC = Nilai MAC

Apabila proses generasi MAC berhasil dilakukan, maka pesan akan digabung dengan MAC sebelum dikirim kepada penerima. Setelah pesan tiba di sisi penerima, penerima akan memisahkan pesan dan MAC. Setelah pesan dan MAC dipisah, maka penerima akan membangkitkan nilai MAC dari pesan yang diterima kemudian membandingkan kedua nilai MAC tersebut. Jika diasumsikan bahwa hanya pengirim dan penerima saja yang mengetahui kunci rahasia, maka apabila nilai MAC pengirim dan penerima sama, penerima dapat memastikan bahwa pesan berasal dari pengirim yang otentik. Gambar 1 menunjukkan proses MAC secara visual.



Gambar 1 Proses MAC [11, p. 389]

Salah satu fungsi MAC yang sering digunakan yaitu *Hash-based Message Authentication Code*(HMAC). Menurut RFC2104, HMAC adalah jenis MAC yang menggunakan *cryptographic hash function.* HMAC dapat digunakan dengan *cryptographic hash function* iteratif apa pun misalnya MD5, SHA1, SHA128, SHA256 maupun SHA512[14]. Gambar 2 menujukkan struktur HMAC.



Gambar 2 Struktur HMAC [11, p. 396]

Berikut ini merupakan beberapa notasi yang digunakan pada struktur HMAC:
H = *Embedded hash function* (contohnya MD5, SHA-1, SHA-256, SHA-512)
IV = *Initial input value* untuk fungsi *hash*
M = Pesan yang akan dimasukkan kedalam fungsi
Yi = Blok ke-i dari pesan M, 0 … i … (L - 1)
L = Jumlah blok dari pesan M
b = Jumlah bit dalam sebuah blok
n = Panjang nilai *hash* yang dihasilkan oleh *hash function*
K = Kunci rahasia
K+ = K yang diisi dengan angka nol di kiri sehingga hasilnya adalah b-bit

Berdasarkan struktur HMAC pada Gambar 2, nilai ipad dan opad adalah sebagai berikut:
ipad = 00110110 (36 dalam heksadesimal) diulang sebanyak b/8 kali
opad = 01011100 (5C dalam heksadesimal) diulang sebanyak b/8 kali

sehingga menghasilkan persamaan HMAC:

$$\mathbf{HMAC}(\boldsymbol{K},\boldsymbol{M}) = \mathbf{H}[(\boldsymbol{K}^{+} \oplus \mathbf{opad}) \,||\, \mathbf{H}[(\boldsymbol{K}^{+} \oplus \mathbf{ipad}) \,||\, \mathbf{M}]] \quad (2)$$

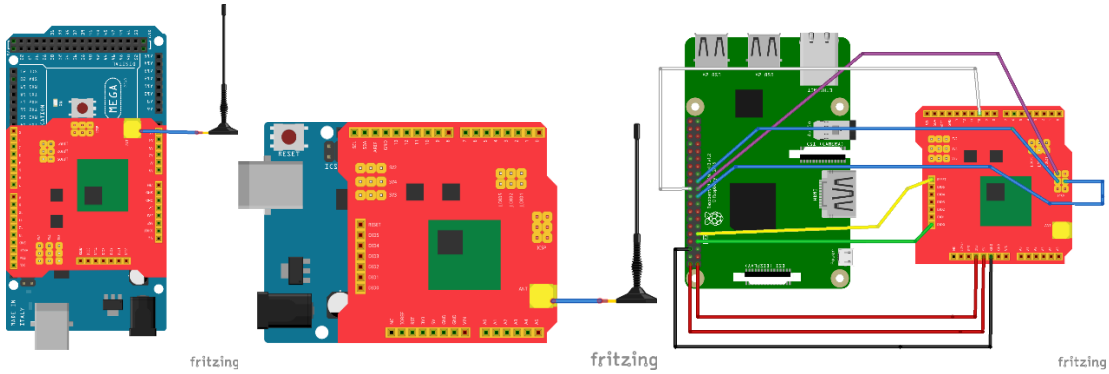
Posisi HMAC pada tugas akhir ini adalah sebagai penjamin aspek integritas dan *authentication*. Adapun variasi HMAC yang digunakan yaitu HMAC-MD5, HMAC-SHA1, HMAC-SHA256, HMAC-SHA512.

3. Sistem yang Dibangun

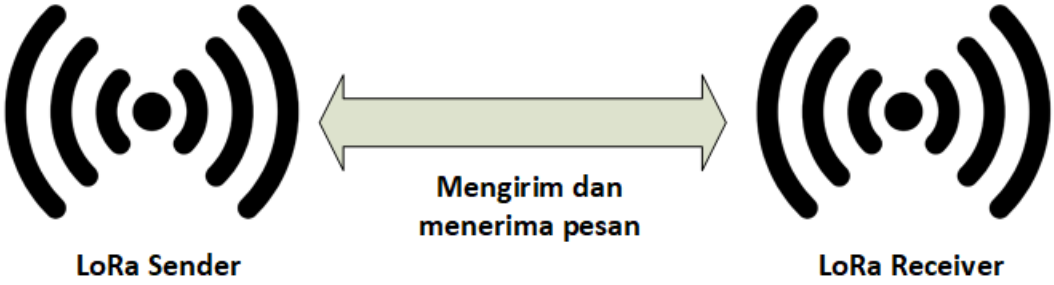
Sistem pengamanan komunikasi pada jaringan LoRa menggunakan algoritma AES dan MAC ini diterapkan pada *constrained devices* IoT kelas 0 dan kelas 2. Perangkat yang berperan sebagai perangkat kelas 0 adalah Arduino Uno sedangkan perangkat kelas 2 adalah Arduino Mega. Jumlah unit dari masing masing kelas adalah 2 unit, satu dari segi pengirim dan satu dari segi penerima. Sehingga pada TA ini hanya dapat menjalankan skema one to one. Proses enkripsi dan dekripsi pesan menggunakan algoritma AES128 dan AES256. Sedangkan proses MAC, penelitian ini menggunakan algoritma HMAC-MD5, HMAC-SHA1, HMAC-SHA256, dan HMAC-SHA512. Penelitian ini menggunakan skema *client* dan *server*. Perangkat pada masing-masing sisi dapat berupa perangkat kelas 0 maupun kelas 2 yang telah disandingkan dengan modul *LoRa*. Gambar 3 menujukkan *sketch device* yang digunakan sedangkan Gambar 4 merupakan arsitektur sistem yang digunakan pada penelitian ini. Detail perangkat serta *software* yang digunakan dipaparkan pada Tabel 1.

Tabel 1 *Hardware* dan *Software*

No.	Nama	Versi	Kuantitas
1.	Arduino Uno	R3	2
2.	Arduino Mega	2560	2
3.	Dragino LoRa HAT	-	2
4	Raspberry Pi	3 b+	1
5.	Dragino LoRa/GPS Hat for Rpi	1.4	1
6.	Arduino IDE	1.8.13	1



Gambar 3 *Sketch Devices*



Gambar 4 Arsitektur Sistem

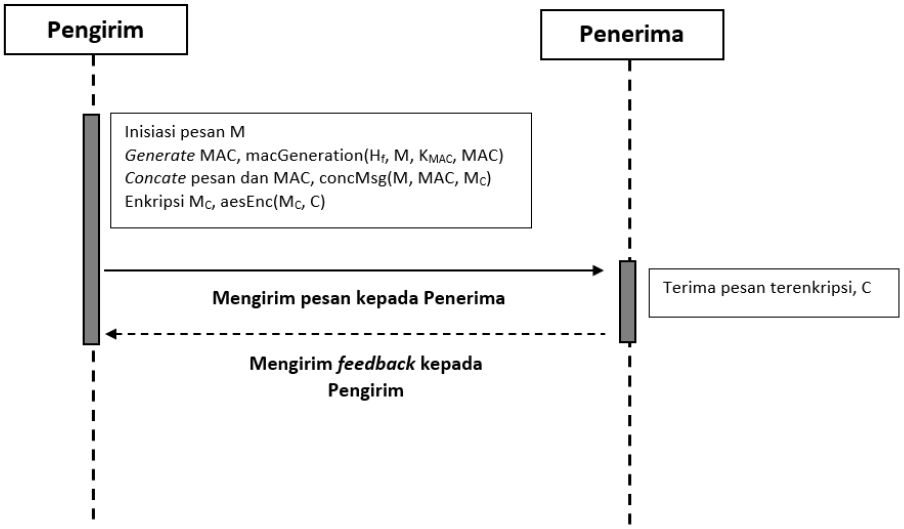
Terdapat dua fase sebelum pesan dikirim kepada penerima yaitu fase generasi MAC dan fase enkripsi pesan. Ketika pesan telah diterima oleh penerima, maka dua fase selanjutnya yang dilakukan yaitu fase dekripsi pesan dan fase verifikasi MAC. Penelitian ini menggunakan beberapa notasi yang berisi parameter yang berhubungan dengan pengirim(*sender*) dan penerima(*receiver*) untuk setiap fase yang ada pada pengirim dan penerima. Adapun notasi yang digunakan adalah sebagai berikut:

Tabel 2 Tabel Notasi

Notasi	Deskripsi
M	Pesan atau *plain text*
K_{MAC}	Kunci yang digunakan untuk membangkitkan nilai MAC
K_{ED}	Kunci yang digunakan untuk proses enkripsi dan dekripsi. Ukuran kunci menyesuaikan dengan algoritma AES yang digunakan
MAC	Nilai MAC yang telah di-*generate*
MAC_R	Nilai MAC yang telah di-*generate* di sisi penerima
HASH_SIZE	Panjang nilai MAC yang dihasilkan
H_f	*Hash function*
KEY_LENGTH	Panjang key untuk proses enkripsi dan dekripsi
M_C	Gabungan pesan dan MAC
C	*Cipher text* dari penggabungan pesan dan MAC
M_D	Hasil dekripsi, terdiri atas pesan dan MAC yang masih menyatu
concMsg()	Fungsi untuk mengabungkan pesan dan MAC
macGeneration()	Fungsi untuk membangkitkan nilai MAC
aesEnc()	Fungsi enkripsi menggunakan AES
aesDec	Fungsi dekripsi menggunakan AES
parseMsg()	Fungsi untuk *parsing* pesan dan MAC dari data yang telah didekripsi
macVerify()	Fungsi untuk memverifikasi nilai kesesuaian antara nilai MAC pengirim dan penerima

3.1. Alur Sistem Sisi *Client*/Pengirim

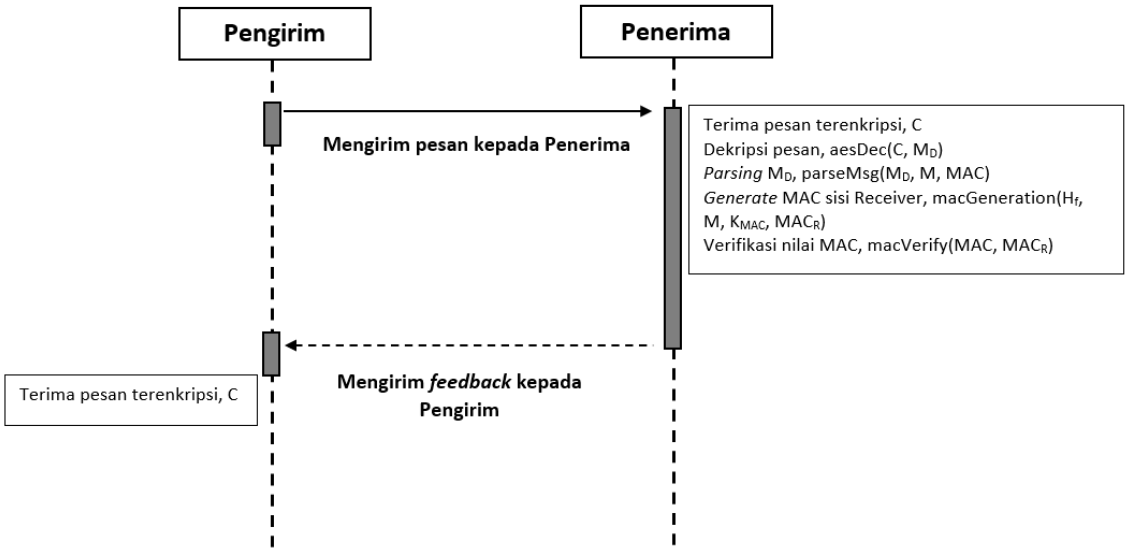
Sebelum pesan diterima oleh penerima, penerima melalui beberapa tahapan agar pesan dapat dikirim dengan aman. Tahap pertama pesan M dinisiasi terlebih dahulu. Setelah pesan berhasil diinisiasi, maka proses *Generate* MAC dilakukan. MAC yang dihasilkan dari proses generasi MAC memiliki ukuran sepanjang HASH_SIZE. Setelah MAC berhasil dibangkitkan, maka proses *concate* pesan dapat dilakukan. Proses *concate* pesan menghasilkan M_c. Kemudian, M_c dienkripsi menggunakan algoritma AES yang menghasilkan C sepanjang KEY_SIZE. Tahap terakhir pesan akan dikirim kepada penerima melalui jaringan LoRa. Apabila pesan berhasil diterima oleh penerima maka penerima akan mengirim pesan *feedback. Feedback* berfungsi untuk mengetahui apakah penerima/*server* telah menerima pesan dengan benar. Gambar 5 menunjukkan alur pengiriman pesan dari pengirim kepada penerima.



Gambar 5 Alur Sistem Sisi *Client*/Pengirim

3.2. Alur Sistem Sisi *Server*/Penerima

Setelah pesan diterima oleh penerima, penerima akan melalui beberapa tahapan agar pesan dapat melihat pesan asli M dan membuktikan bahwa pesan dikirim oleh pengirim yang seharusnya. Pada tahap pertama, pesan yang terenkripsi C akan didekripsi terlebih dahulu menjadi M_D. Setelah pesan berhasil didekripsi, maka M_D akan dipecah menjadi M dan MAC. Setelah itu, penerima akan melakukan proses generasi MAC_R. MAC_R yang dihasilkan dari proses generasi MAC memiliki ukuran sepanjang HASH_SIZE. Setelah MAC_R berhasil dibangkitkan, maka penerima melakukan proses verifikasi MAC dengan membandingkan nilai MAC dan MAC_R. Apabila nilai MAC dan MAC_R sama, maka proses verifikasi berhasil sebaliknya apabila berbeda, maka proses verifikasi gagal. Gambar 6 menunjukkan alur sistem dari sisi penerima.



Gambar 6 Alur Sistem Sisi *Server*/Penerima

3.3. Threat Model

Model penyerangan yang dilakukan pada penelitian ini adalah model penyerangan *eavesdropping attack*. *Eavesdropping* dilakukan melalui proses *sniffing* terhadap pesan (*payload)* yang dikirimkan melalui jaringan LoRa. Pada penelitian ini, peran *Sniffer* akan dilakukan oleh Raspberry pi 3 B+. Gambar 7 merupakan skema serangan *eavesdropping*.

Gambar 7 Skema Serangan *Eavesdropping*

3.4. Skenario Pengujian

Pengujian pada penelitian ini akan dilakukan sebanyak sepuluh kali. Skenario pengujian dari sistem peningkatan keamanan komunikasi yang dibangun dimulai dari pengujian terhadap parameter keamanan. Parameter keamanan yang pertama kali diuji adalah *confidentiality*. Pada tahap pertama, pengirim akan mengirim pesan tanpa dienkripsi terlebih dahulu kepada penerima. Selama proses ini berlangsung, *siniffer* melakukan *sniffing* terhadap pesan (*payload*) yang dikirim. Kemudian *sniffer* akan mencoba untuk mendapatkan pesan yang asli. Setelah tahap pertama berhasil dilakukan, maka pengirim akan melakukan proses generasi MAC dan enkripsi pesan terlebih dahulu sebelum dikirim kepada penerima. Selama proses transmisi berlangsung, *sniffer* melakukan *sniffing* terhadap pesan (*payload*) yang dikirim. Kemudian *sniffer* juga akan mencoba untuk mendapatkan pesan asli. Parameter keberhasilan dari dua tahap pengujian parameter keamanan *confidentiality* dipaparkan pada Tabel 3.

Tabel 3 Parameter Keberhasilan Pengujian *Confidentiality*

No.	Mode	Kondisi Seharusnya
1.	Tanpa metode keamanan	*Sniffer* dapat melihat *plain text*
2.	Enkripsi dan MAC	*Sniffer* hanya melihat *Cipher text*

Parameter keamanan yang diuji selanjutnya adalah *integrity* dan *authentication.* Pada pengujian ini, pengirim akan melakukan proses generasi MAC dan enkripsi pesan terlebih dahulu sebelum dikirim kepada penerima. Setelah pesan diterima oleh penerima, maka penerima akan melakukan dekripsi pesan dan membandingkan nilai MAC yang diterima dari pengirim dengan nilai MAC yang dibangkitkan di sisi penerima. parameter keberhasilan pengujian parameter keamanan *integrity* dipaparkan pada Tabel 4. Sedangkan keberhasilan pengujian parameter keamanan *authentication* dipaparkan pada Tabel 5.

Tabel 4 Parameter Keberhasilan Pengujian *Integrity*

No.	Pesan	Kondisi Seharusnya
1.	Tidak dimodifikasi setelah proses generasi MAC berhasil	*Match*
2.	Dimodifikasi setelah proses generasi MAC berhasil	*Unmatch*

Tabel 5 Parameter Keberhasilan Pengujian *Authentication*

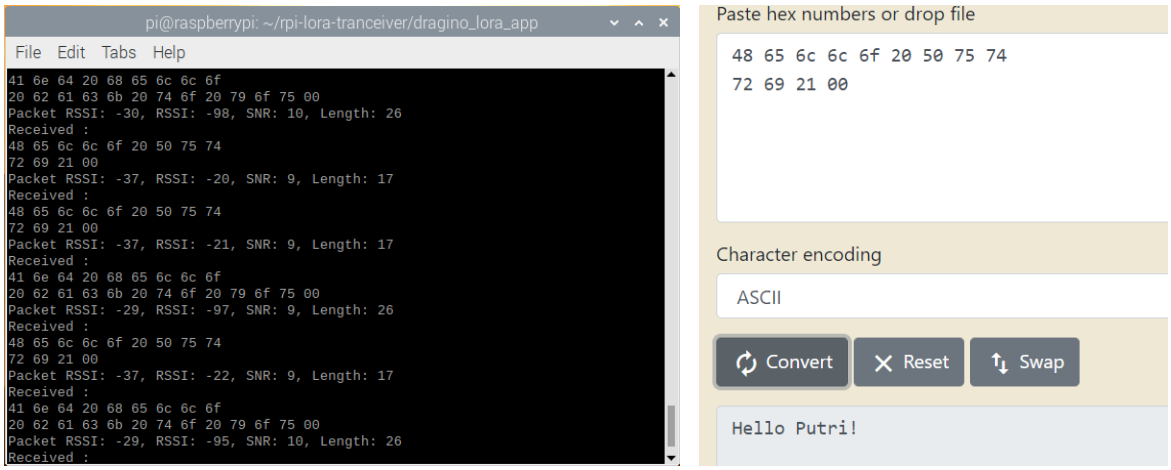
No.	Key	Kondisi Seharusnya
1.	Sama	*Match*
2.	Berbeda	*Unmatch*

Apabila pengujian terhadap parameter keamanan telah berhasil dilakukan, maka analisis *overhead* dapat dilakukan. Adapun yang dianalisis yaitu penggunaan *Flash*, penggunaan memori, lama waktu enkripsi dan dekripsi pesan, waktu tunggu kedatangan pesan selanjutnya, lama waktu proses generate MAC serta waktu yang butuhkan untuk verifikasi MAC. Pada penelitian ini, peneliti menggunakan Arduino IDE untuk mengetahui penggunaan *flash* dan memori serta *library* micros() untuk mendapatkan nilai waktu dari komponen *overhead* yang memiliki satuan dalam bentuk waktu. Adapun nilai yang diambil untuk dari komponen *overhead* yang memiliki satuan dalam bentuk waktu adalah nilai rata-rata dari sepuluh kali pengujian yang dilakukan.

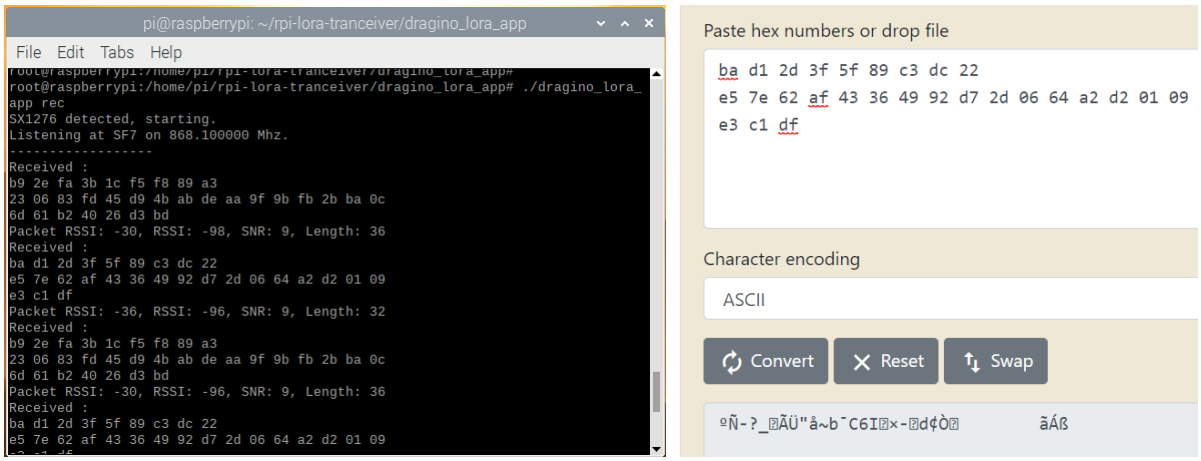
4. Evaluasi

4.1. *Security Analysis* : *Confidentiality*

Berdasarkan pengujian yang telah dilakukan terhadap seluruh tahapan skenario pengujian parameter keamanan *confidentiality* menggunakan skenario *eavesdropping*, maka hasil yang didapat adalah sebagai berikut:



Gambar 8 Hasil Proses *Sniffing* pada Pengujian Tanpa Sistem Keamanan



Gambar 9 Hasil Proses *Sniffing* pada Pengujian Menggunakan Sistem Keamanan

Apabila merujuk pada Gambar 8 dan Gambar 9, maka pengujian terhadap parameter keamanan *confidentiality* telah berhasil dilakukan. Adapun detail keberhasilan pengujian parameter *confidentiality* dipaparkan pada Tabel 6.

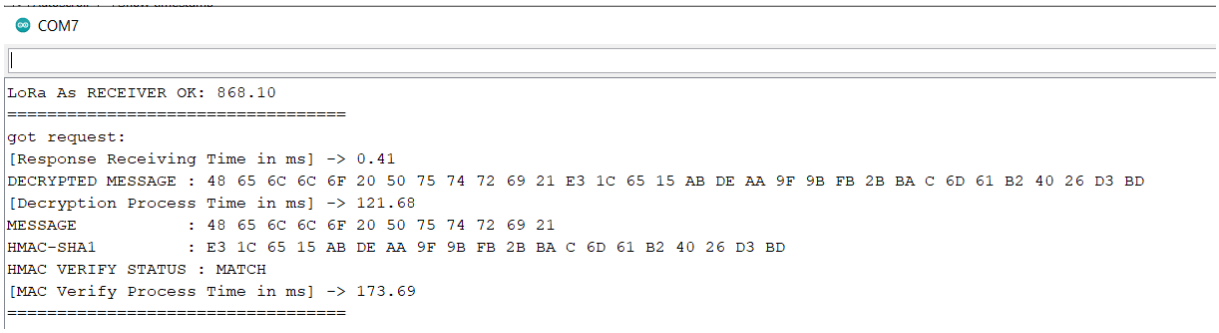
Tabel 6 Hasil Pengujian Parameter *Confidentiality*

No.	Mode	Kondisi Seharusnya	Kondisi Pengujian
1.	Tanpa metode keamanan	*Sniffer* dapat melihat *plain text*	*Sniffer* dapat melihat *plain text*
2.	Enkripsi dan MAC	*Sniffer* hanya melihat *Cipher text*	*Sniffer* hanya melihat *Cipher text*

Tabel 6 menunjukkan bahwa aspek *confidentiality* telah terjamin menggunakan metode yang diusulkan pada penelitian ini. Hal ini dibuktikan melalui kesesuaian antara kondisi seharusnya dan kondisi ketika dilakukan pengujian.

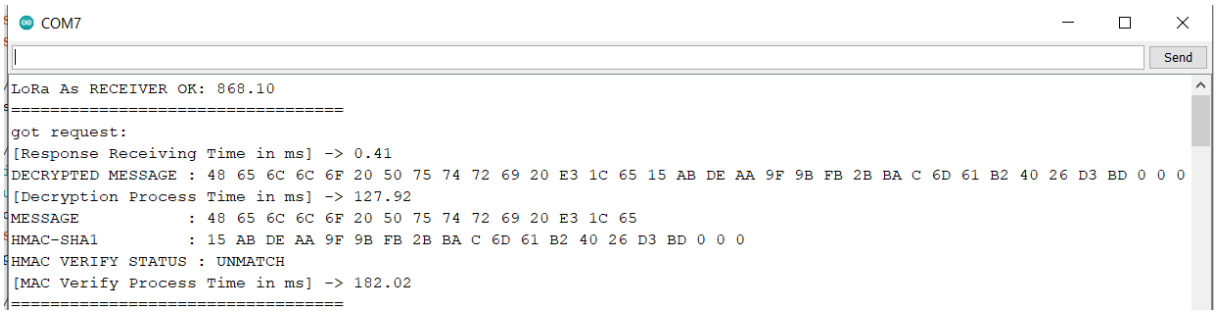
4.2. *Security Analysis* : *Integrity*

Berdasarkan pengujian yang telah dilakukan terhadap seluruh tahapan skenario pengujian parameter keamanan *integrity*, maka hasil yang didapat adalah sebagai berikut:



Gambar 10 Hasil Pengujian *Integrity* Apabila Pesan Tidak Dimodifikasi Setelah Proses Generasi MAC Berhasil

Pesan asli yang seharusnya dikirimkan adalah “Hello Putri!”, akan tetapi setelah MAC berhasil dibangkitkan, pesan dimodifikasi menjadi “Hello Putri AW!”. Oleh sebab itu, hasil verifikasi MAC disisi penerima tidak berhasil diverifikasi (*Unmatch*).



Gambar 11 Hasil Pengujian *Integrity* Apabila Pesan Dimodifikasi Setelah Proses Generasi MAC Berhasil

Apabila merujuk pada Gambar 10 dan Gambar 11, maka pengujian terhadap parameter keamanan *integrity* telah berhasil dilakukan. Adapun detail keberhasilan pengujian parameter *integrity* dipaparkan pada Tabel 7.

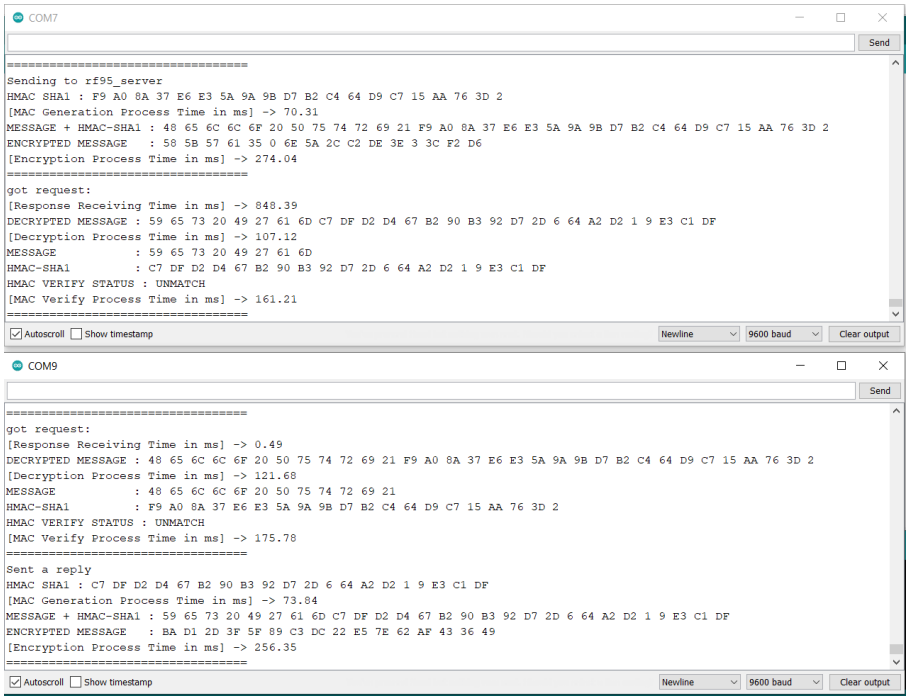
Tabel 7 Hasil Pengujian Parameter Keamanan *Integrity*

No.	Pesan	Kondisi Seharusnya	Kondisi Pengujian
1.	Tidak dimodifikasi setelah proses generasi MAC berhasil	*Match*	*Match*
2.	Dimodifikasi setelah proses generasi MAC berhasil	*Unmatch*	*Unmatch*

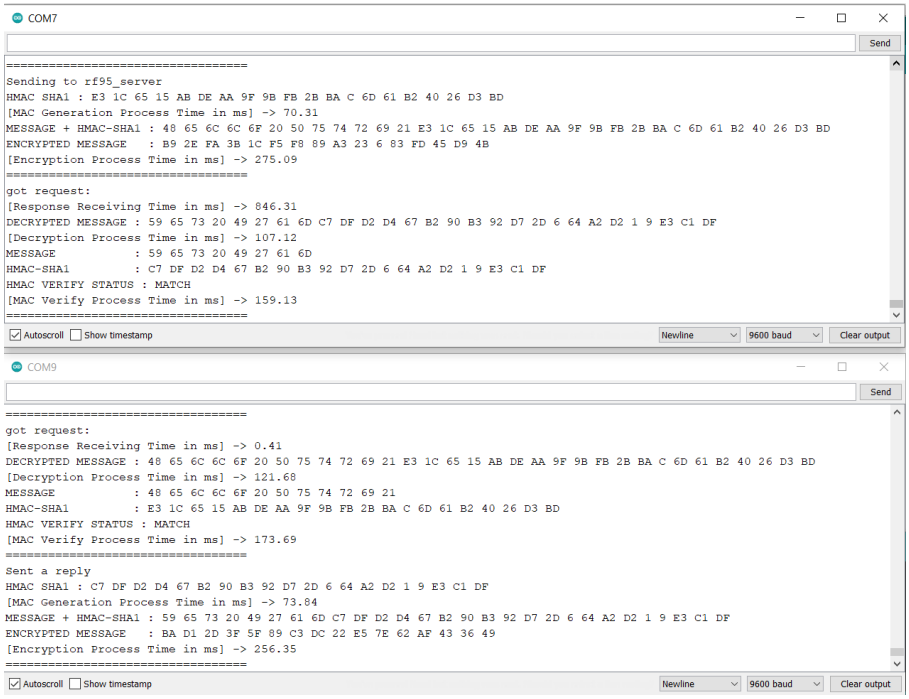
Tabel 7 menunjukkan bahwa aspek *integrity* telah terjamin menggunakan metode yang diusulkan pada penelitian ini. Hal ini dibuktikan melalui kesesuaian antara kondisi seharusnya dengan kondisi pengujian.

4.3. *Security Analysis* : *Authentication*

Berdasarkan pengujian yang telah dilakukan terhadap seluruh tahapan skenario pengujian parameter keamanan *authentication*, maka hasil yang didapat adalah sebagai berikut:



Gambar 12 Hasil Pengujian *Authentication* Apabila Kunci Rahasia Pengirim dan Penerima Berbeda



Gambar 13 Hasil Pengujian *Authentication* Apabila Kunci Rahasia Pengirim dan Penerima Sama

Apabila merujuk pada Gambar 12 dan Gambar 13, maka pengujian terhadap parameter keamanan *authentication* telah berhasil dilakukan. Adapun detail keberhasilan pengujian parameter *authentication* dipaparkan pada Tabel 8.

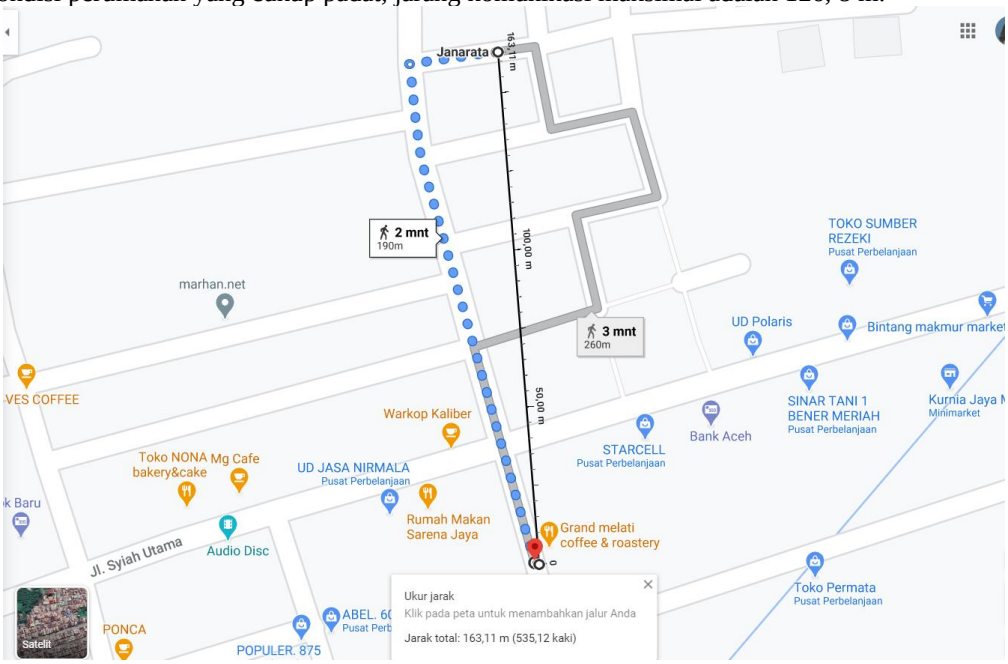
Tabel 8 Hasil Pengujian Parameter *Authentication*

No.	Key	Kondisi Seharusnya	Kondisi Pengujian
1.	Sama	*Match*	*Match*
2.	Berbeda	*Unmatch*	*Unmatch*

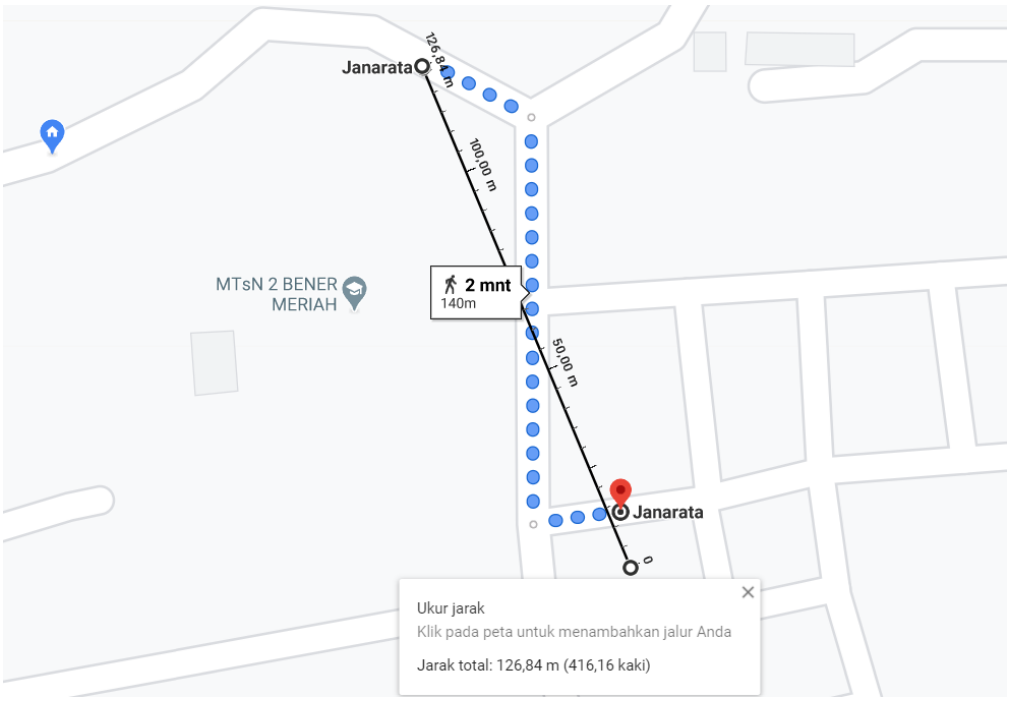
Tabel 8 menunjukkan bahwa aspek *authentication* telah terjamin menggunakan metode yang diusulkan pada penelitian ini. Hal ini dibuktikan melalui kesesuaian antara kondisi seharusnya dan kondisi ketika dilakukan pengujian.

4.4. Jarak Penggunaan LoRa

Berdasarkan pengujian yang telah dilakukan, penggunaan modul LoRa pada *constrained devices* IoT kelas 0 dan kelas 2 masih mampu berkomunikasi dengan jarak maksimal 163,11 m pada permukaan lingkungan yang datar dan perumahan atau pertokoan yang padat. Sedangkan pada permukaan lingkungan datar menuju lembah dengan kondisi perumahan yang cukup padat, jarang komunikasi maksimal adalah 126, 8 m.



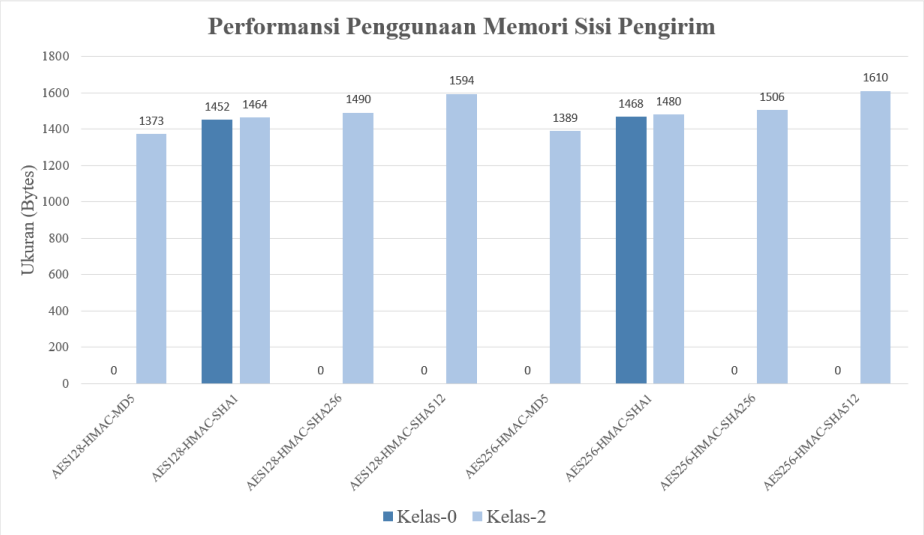
Gambar 14 Jarak Komunikasi Maksimal LoRa pada Permukaan Lingkungan yang Datar



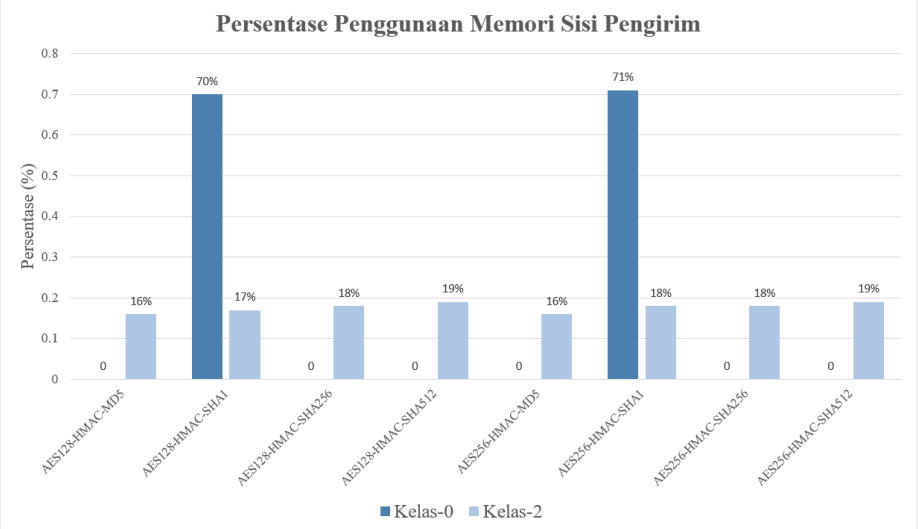
Gambar 15 Jarak Komunikasi Maksimal LoRa pada Permukaan Lingkungan yang Datar Menuju Lembah

4.5. *Overhead Analysis* : Penggunaan Memori

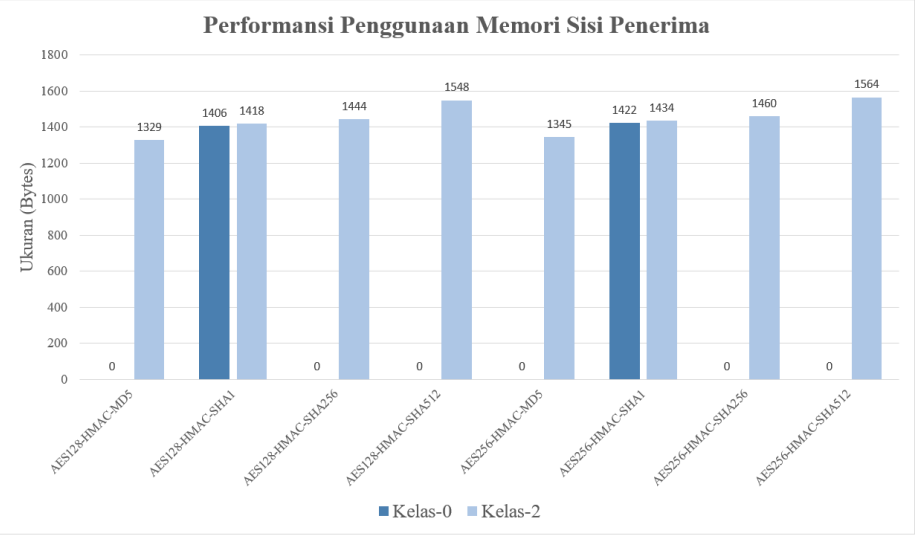
Analisis *Overhead* penggunaan memory bertujuan untuk mengetahui seberapa besar memori dinamis perangkat yang digunakan untuk penerapan AES dan MAC. Perangkat kelas 0 dalam hal ini Arduino Uno memiliki memori sebesar 2048 Bytes. Apabila merujuk pada Gambar 16, Gambar 17, Gambar 18, dan Gambar 19, Perangkat kelas 0 hanya dapat menerapkan metode AES128-HMAC-SHA1 dan AES256-HMAC-SHA1 karena penggunaan memorinya berkisar antara 1452-1468 Bytes (70% - 71%), Sedangkan untuk metode lainnya memori yang dihabiskan sudah melebihi 72% yang mengakibatkan perangkat terlalu sering melakukan *oneself restart* sehingga tidak cocok untuk diterapkan pada perangkat kelas 0. Sebaliknya perangkat kelas 2 dapat menerapkan semua metode karena ukuran memorinya empat kali lebih besar daripada kelas 0 yaitu sebesar 8192 Bytes. Apabila merujuk pada Gambar 16, Gambar 17, Gambar 18, dan Gambar 19, penggunaan memori terus meningkat sesuai dengan metode yang digunakan. Peningkatan ukuran dari masing-masing metode HMAC dengan algoritma enkripsi yang berbeda adalah sebesar 16Bytes. Hal ini disebabkan oleh berubahnya ukuran kunci dari algoritma AES128 sebesar 16 Bytes menjadi AES256 sebesar 32 Bytes.



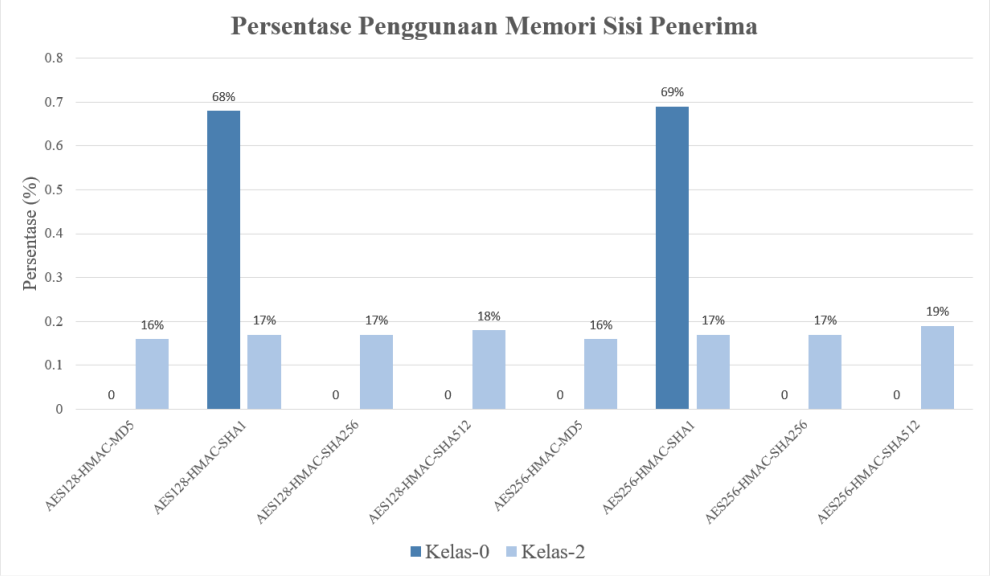
Gambar 16 Penggunaan Memori Sisi Pengirim



Gambar 17 Presentase Penggunaan Memori Sisi Pengirim



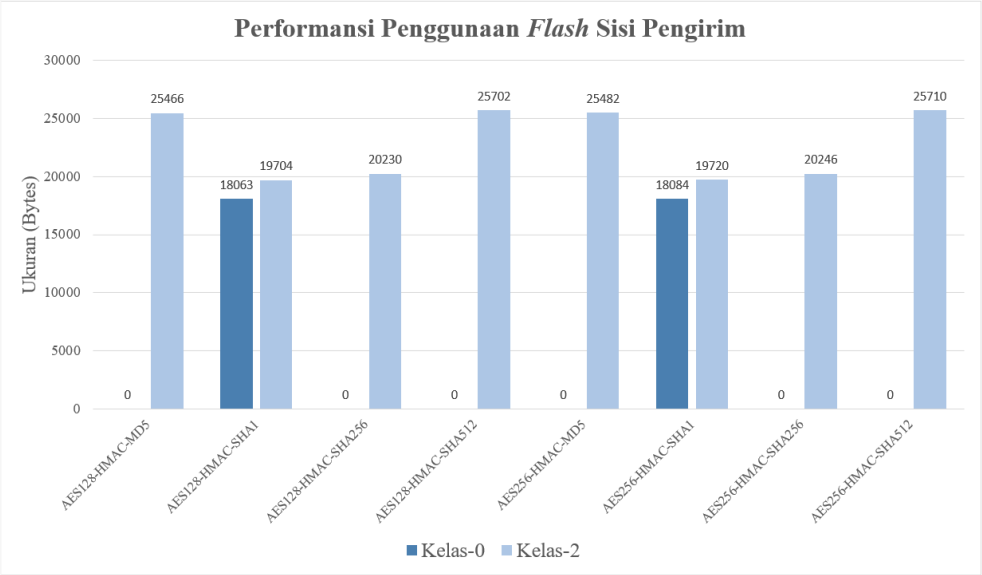
Gambar 18 Penggunaan Memori Sisi Penerima



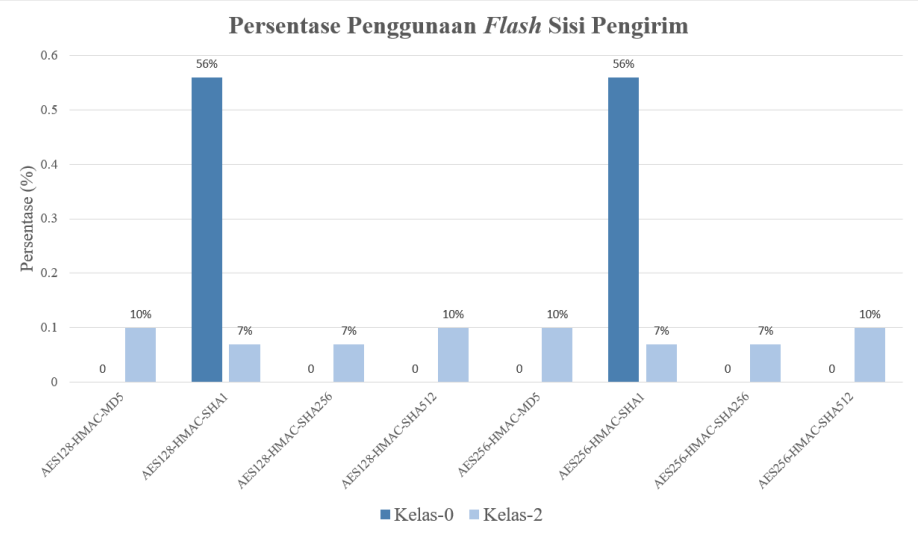
Gambar 19 Presentase Penggunaan Memori Sisi Penerima

4.6. *Overhead Analysis* : Penggunaan *Flash*

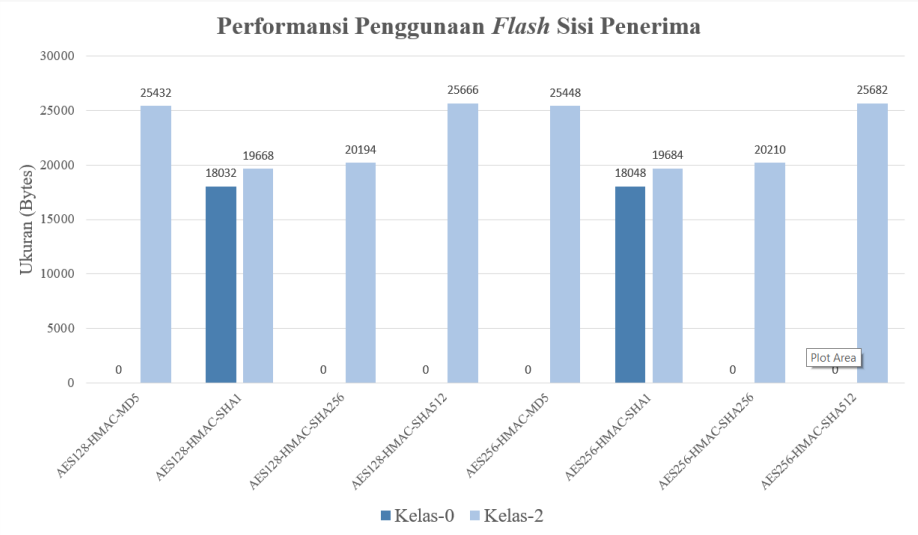
Analisis overhead penggunaan *Flash* berfungsi untuk mengetahui seberapa besar penyimpanan yang digunakan untuk penerapan AES dan MAC. Perangkat kelas 0 dalam hal ini Arduino Uno memiliki penyimapan sebesar 32256 Bytes. Apabila merujuk pada Gambar 20, Gambar 21, Gambar 22, dan Gambar 23, Perangkat kelas 0 hanya dapat menerapkan metode AES128-HMAC-SHA1 dan AES256-HMAC-SHA1 karena penggunaan *flash* berkisar antara 18063-18084 Bytes(56%). Sedangkan untuk metode lainnya seperti AES128-HMAC-MD5 dan AES256-HMAC-MD5, *flash* yang dihabiskan sudah melebihi 73% yang mengakibatkan perangkat terlalu sering melakukan *oneself restart* sehingga tidak cocok untuk diterapkan pada perangkat kelas 0. Sebaliknya perangkat kelas 2 dapat menerapkan semua metode karena ukuran *flash* delapan kali lebih besar daripada kelas 0 yaitu sebesar 253952 Bytes. Pada penelitian ini, metode yang menggunakan *flash* paling banyak adalah AES128-HMAC-MD5, AES256-HMAC-MD5, AES128-HMAC-SHA512 dan AES256-HMAC-SHA512 yaitu sebesar 10% dari total *flash* pada kelas 2.



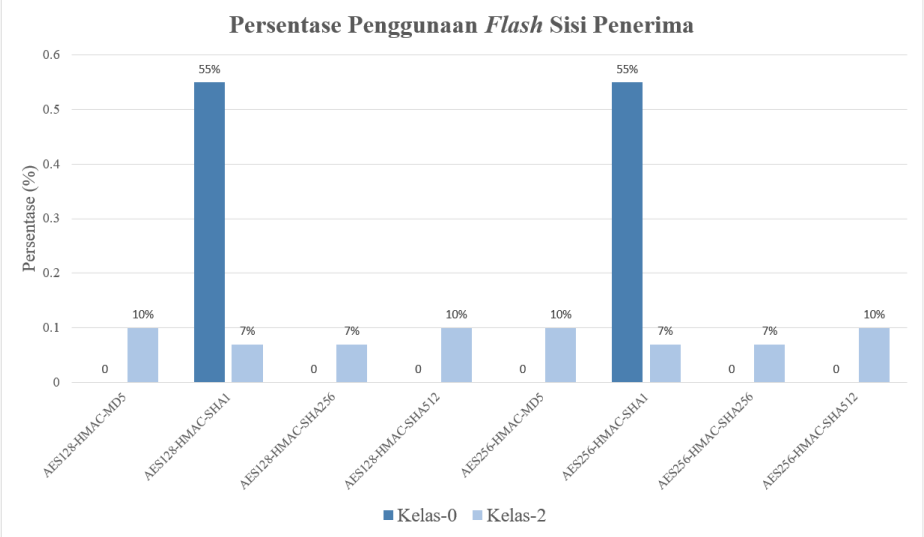
Gambar 20 Penggunaan *Flash* Sisi Pengirim



Gambar 21 Presentase Penggunaan *Flash* Sisi Pengirim



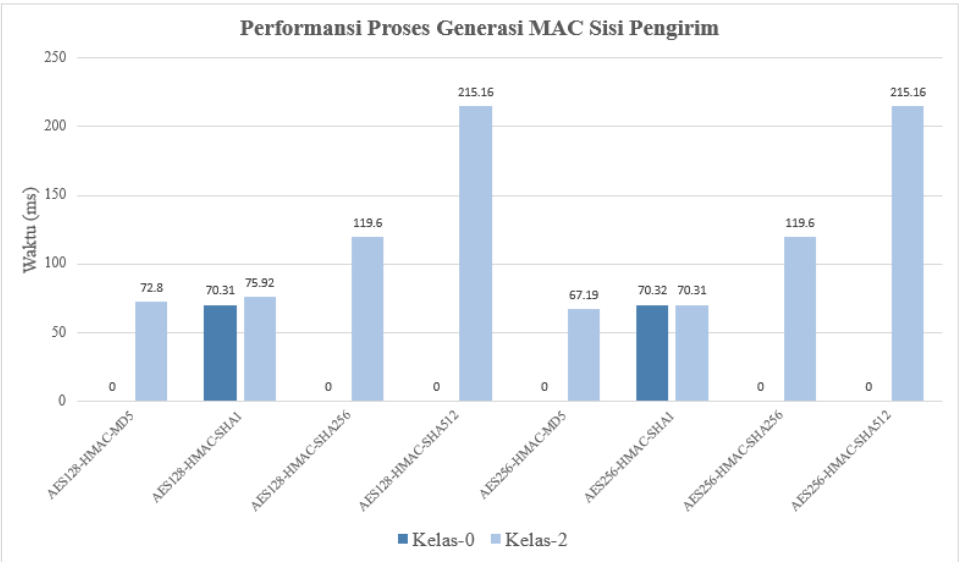
Gambar 22 Penggunaan *Flash* Sisi Penerima



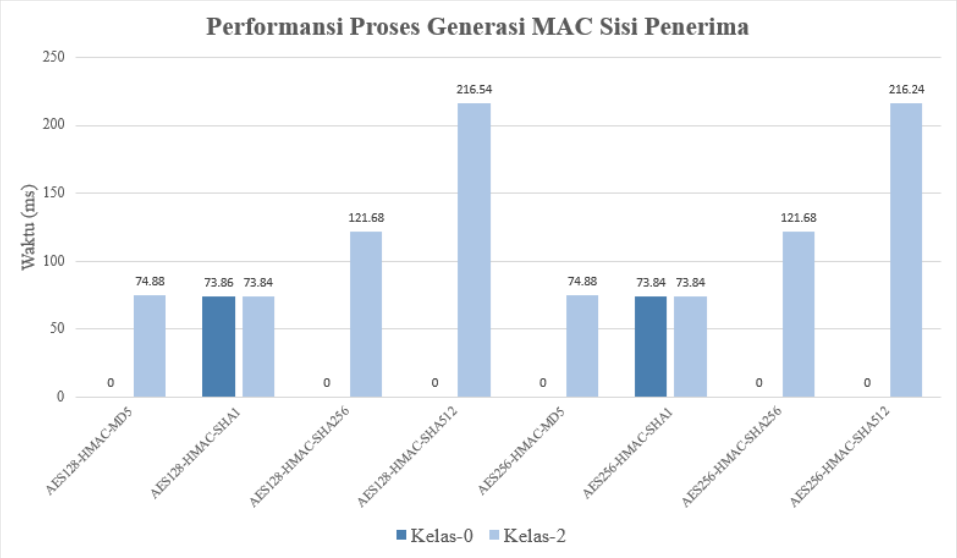
Gambar 23 Presentase Penggunaan *Flash* Sisi Penerima

4.7. *Overhead Analysis* : Proses MAC *Generation*

Pengujian ini dilakukan dengan menghitung berapa lama waktu yang dibutuhkan untuk membangkitkan nilai MAC dari pesan yang dikirimkan. Berdasarkan Gambar 24 dan Gambar 25, lama waktu proses generasi MAC terus meningkat sesuai dengan panjang ukuran MAC. Semakin kecil ukuran MAC yang dihasilkan maka waktu generasi semakin kecil. Dari segi pengirim, metode yang memiliki waktu generasi MAC paling kecil apabila dibandingkan dengan tujuh variasi metode lainnya yaitu AES256-HMAC-SHA1 untuk perangkat kelas 2 dan AES128-HMAC-SHA1 untuk perangkat kelas 0 dengan nilai 70.31 ms. Sebaliknya dari sisi penerima, metode AES256-HMAC-SHA1 menghasilkan waktu generasi MAC paling kecil untuk semua kelas *device* selama 70.31 ms.



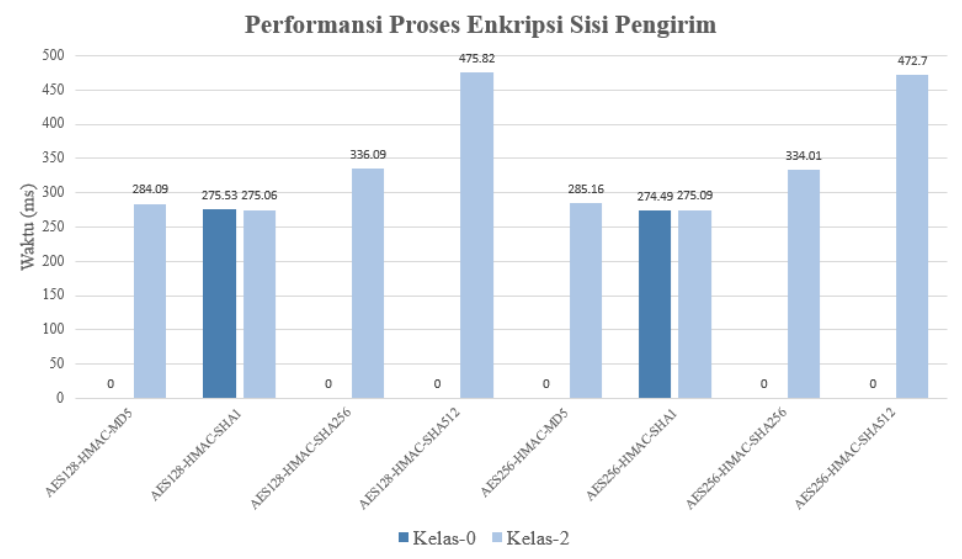
Gambar 24 Performansi Proses MAC *Generation* Sisi Pengirim



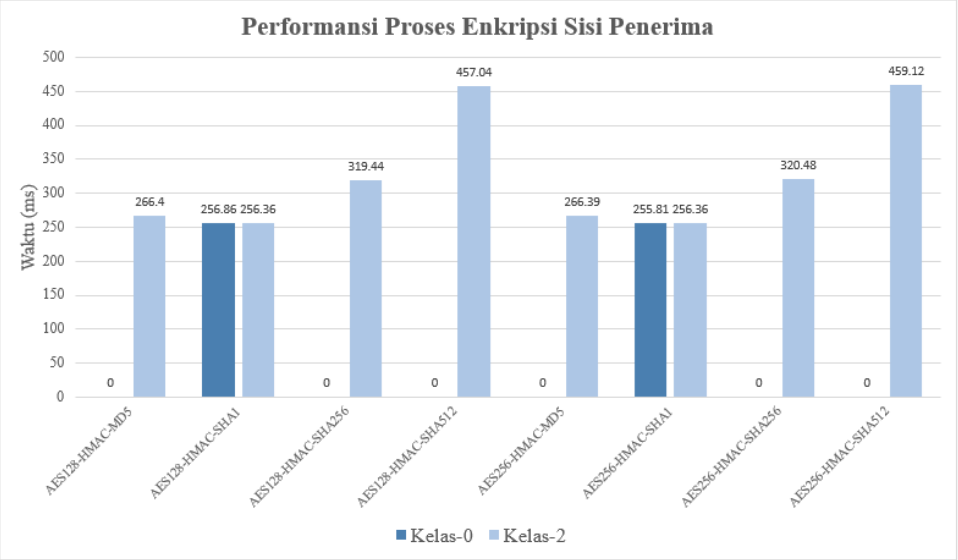
Gambar 25 Performansi Proses MAC *Generation* Sisi Penerima

4.8. *Overhead Analysis* : Proses Enkripsi

Pengujian ini dilakukan dengan menghitung berapa lama waktu yang dibutuhkan untuk mengenkripsi pesan yang dikirimkan. Berdasarkan Gambar 26 dan Gambar 27, lama waktu enkripsi pesan terus meningkat sesuai dengan panjang pesan, ukuran kunci dan ukuran blok yang dihasilkan oleh variasi AES yang digunakan. Semakin kecil ukuran kunci dan panjang pesan dari variasi AES yang digunakan maka waktu enkripsi semakin singkat. Dari segi pengirim, metode yang memiliki waktu enkripsi paling singkat yaitu AES128-HMAC-SHA1 karena ukuran kunci dan panjang pesan lebih kecil daripada metode lainnya. Akan tetapi pada sisi penerima, terjadi perubahan dimana metode AES256-HMAC-SHA1 menghasilkan waktu menghasilkan waktu enkripsi paling singkat untuk perangkat kelas 0 dengan selisi 1.04 ms daripada AES128-HMAC-SHA1 pada kelas 0.



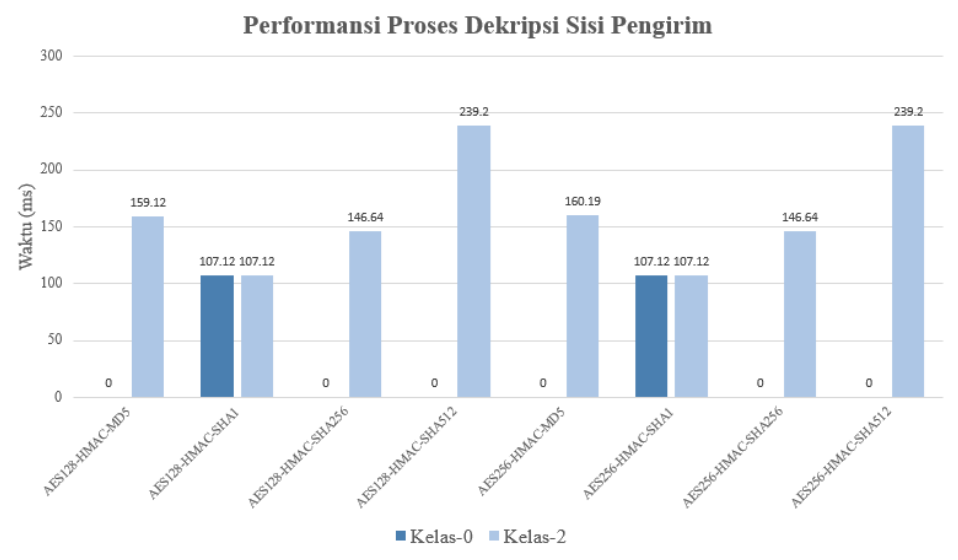
Gambar 26 Performansi Proses Enkripsi Pesan Sisi Pengirim



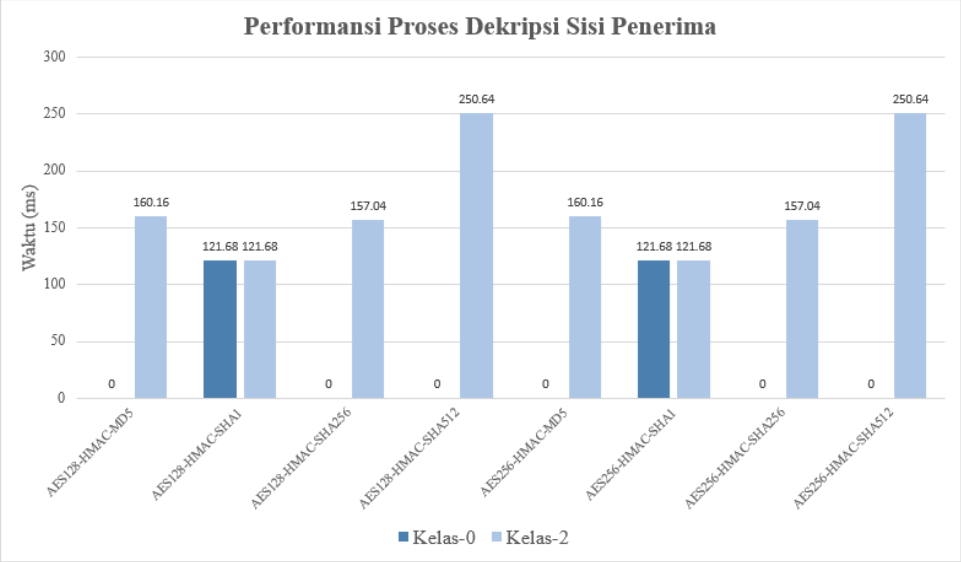
Gambar 27 Performansi Proses Enkripsi Pesan Sisi Penerima

4.9. *Overhead Analysis* : Proses Dekripsi

Analisis *overhead* proses dekripsi dilakukan dengan menghitung berapa lama waktu yang dibutuhkan untuk mendekripsi pesan yang diterima. Berdasarkan Gambar 28 dan Gambar 29, lama waktu dekripsi juga sama seperti waktu enkripsi pesan dimana waktu terus meningkat sesuai dengan panjang pesan, ukuran kunci, dan ukuran blok yang dihasilkan dari variasi AES yang digunakan. Pada kondisi nyata semakin kecil ukuran kunci dan panjang pesan dari variasi AES yang digunakan maka waktu dekripsi semakin singkat akan tetapi ada kemungkinan kecil bahwa AES128 dan AES256 tidak memiliki selisih waktu tergantung pada perangkat yang digunakan. Dari segi pengirim, metode yang memiliki waktu enkripsi paling singkat yaitu AES128-HMAC-SHA1 dan AES256-HMAC-SHA1 karena panjang pesan lebih kecil daripada metode lainnya. Hal tersebut juga berlaku pada sisi penerima, dimana metode AES128-HMAC-SHA1 dan AES256-HMAC-SHA1 menghasilkan waktu menghasilkan waktu dekripsi paling singkat untuk semua kelas *device.*



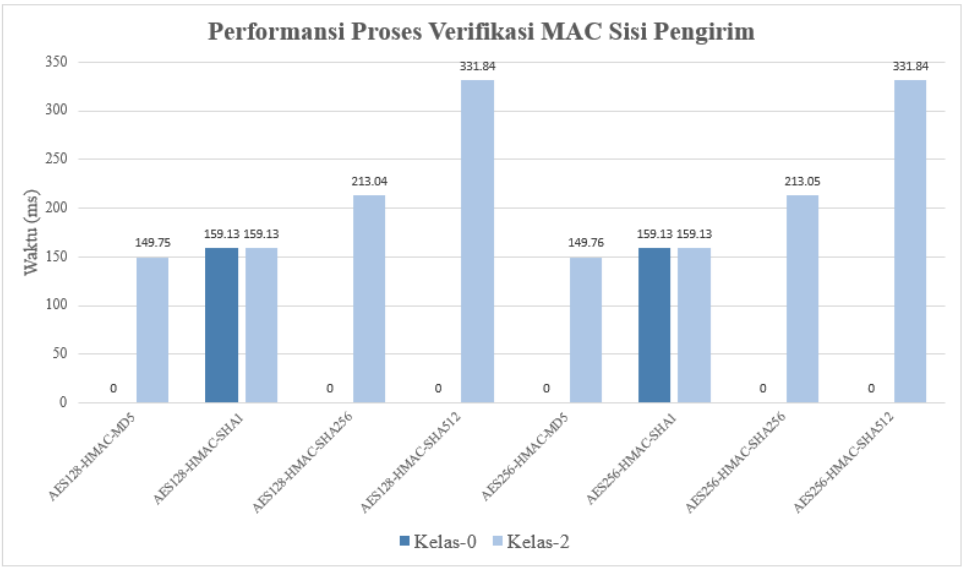
Gambar 28 Performansi Proses Dekripsi Pesan Sisi Pengirim



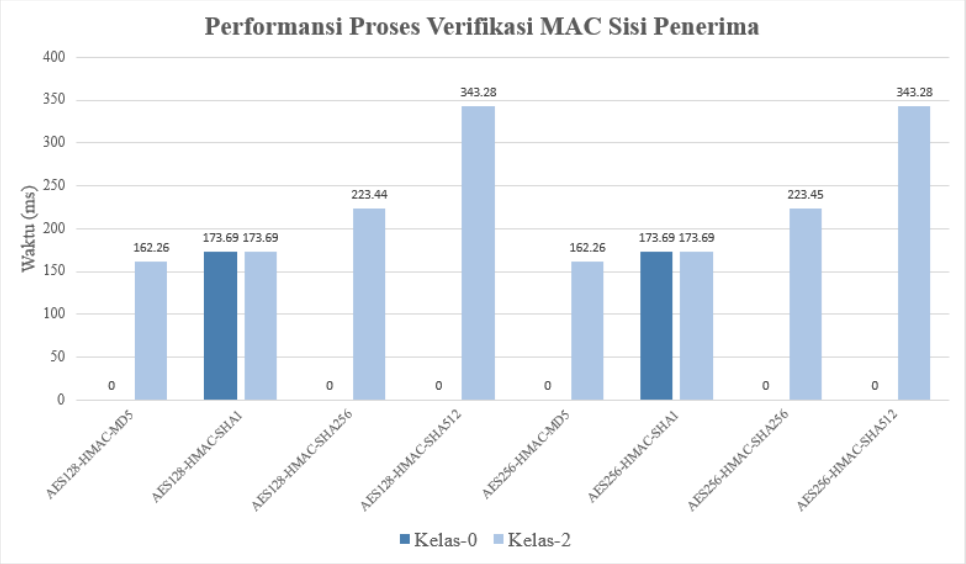
Gambar 29 Performansi Proses Dekripsi Pesan Sisi Penerima

4.10. *Overhead Analysis* : Proses Verifikasi MAC

Pengujian ini dilakukan dengan menghitung berapa lama waktu yang dibutuhkan untuk memverifikasi kesesuaian nilai MAC pengirim dengan nilai MAC yang dibangkitkan pada sisi penerima. Berdasarkan Gambar 30 dan Gambar 31, lama waktu proses generasi MAC terus meningkat sesuai dengan panjang pesan dan ukuran MAC. Semakin kecil ukuran pesan dan MAC yang dihasilkan maka waktu generasi semakin kecil. Dari sisi pengirim, metode yang memiliki waktu verifikasi MAC paling kecil untuk semua kelas *device* yaitu AES128-HMAC-MD5. Hal tersebut juga berlaku di sisi penerima.



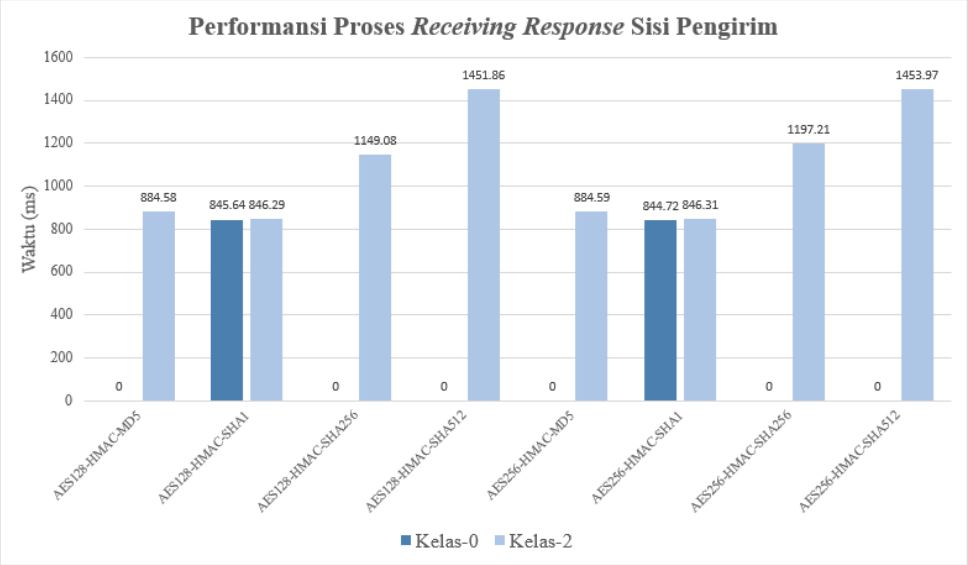
Gambar 30 Performansi Proses Verifikasi MAC Sisi Pengirim



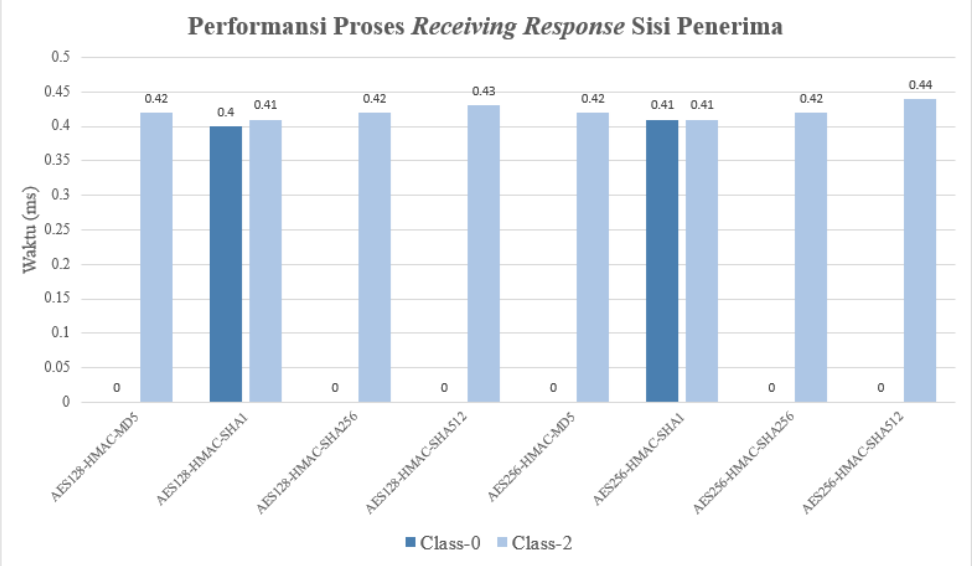
Gambar 31 Performansi Proses Verifikasi MAC Sisi Penerima

4.11. *Overhead Analysis* : *Receiving Response*

Analisis *Overhead response receiving time* dilakukan untuk mengetahui berapa lama pengirim dan penerima menunggu untuk pesan selanjutnya. Berdasarkan Gambar 32, lama waktu pengirim menerima *feedback* dari penerima terus meningkat sesuai dengan lama proses dekripsi pesan yang diterima hingga proses Verifikasi MAC dan proses inisiasi *feedback* hingga proses enkripsi *feedback* pada penerima. Semakin cepat proses tersebut, maka waktu tunggu semakin singkat. Pengirim memiliki waktu tunggu *feedback* lebih lama daripada penerima karena terdapat parameter *timeout* selama 5 detik. *Timeout* berfungsi untuk mengetahui apakah *server/*penerima tersedia atau tidak. Selain itu, apabila pesan telah tiba disisi penerima, maka penerima akan memproses terlebih dahulu sebelum mengirim *feedback* kepada pengirim pesan.



Gambar 32 Performansi Lama Waktu Penerimaan Pesan Sisi Pengirim



Gambar 33 Performansi Lama Waktu Penerimaan Pesan Sisi Penerima

Dari segi penerima berdasarkan Gambar 33, waktu tunggu pesannya lebih kecil daripada pengirim karena yang dihitung adalah lama waktu transmisi melalui medium LoRa ketika penerima bersedia menerima pesan sebaliknya apabila penerima tidak bersedia menerima pesan, maka waktunya tidak dihitung.

5. Kesimpulan

Long Range(LoRa) merupakan salah satu jenis *wireless connectivity* yang populer pada jaringan IoT. Penggunaan IoT semakin meningkat setiap tahun khususnya penggunaan LoRa, begitu juga dengan pengembangan perangkat IoT. Salah satu karakteristik perangkat IoT yaitu *resource* yang terbatas. Perangkat ini sering disebut sebagai *constrained device* IoT. Seiring dengan meningkatnya penggunaan LoRa, aspek keamanan komunikasi pada jaringan LoRa juga harus diperhatikan. Akan tetapi, keterbatasan *resource* yang dimiliki oleh perangkat IoT menjadi tantangan dalam memilih metode *security* yang sesuai. Oleh karena itu, untuk mengatasi masalah tersebut dibutuhkan sebuah metode *security* yang sesuai yaitu pemanfaatan algoritma AES dan MAC. Berdasarkan percobaan yang telah dilakukan, penggabungan metode AES dan MAC telah mampu mengamankan komunikasi antar perangkat pada jaringan LoRa dengan jaminan *confidentiality, integrity* dan *authentication.* Hal ini dibuktikan dengan, tidak berhasilnya *sniffer* mendapatkan pesan asli pada proses *sniffing* dan nilai MAC berhasil diverifikasi dengan benar di sisi penerima. Selain analisis terhadap keamanan (*Security Analysis*), penelitian ini juga melakukan analisis *overhead.* Adapun komponen yang dianalisis pada analisis *overhead* yaitu penggunaan *Flash*, penggunaan memori, lama waktu enkripsi dan dekripsi pesan, waktu tunggu kedatangan pesan selanjutnya, lama waktu proses *generate* MAC serta waktu yang butuhkan untuk verifikasi MAC. Berdasarkan hasil analisis *overhead*, semua variasi metode ini sangat cocok untuk diterapkan pada perangkat kelas 2. Sedangkan untuk perangkat kelas 0 hanya mampu menerapkan metode AES128-HMAC-SHA1 dan AES256-HMAC-SHA1 karena perangkat kelas 0 memiliki ukuran memori dan *flash* lebih kecil daripada perangkat kelas 2.

Pada penelitian ini, arsitektur yang dibangun masih bersifat lokal. Oleh sebab itu, untuk penelitian selanjutnya diharapkan dapat mengimplementasikan metode ini pada jaringan LoRaWAN agar data dapat diakses melalui jaringan internet.

Referensi

[1] R. Minerva, A. Biru, and D. Rotondi, "Towards a Definition of the Internet of Things (IoT)," *IEEE Internet Initiat.*, pp. 1–86, 2015, doi: 10.1111/j.1440-1819.2006.01473.x.

[2] Gartner, "Gartner Says 5.8 Billion Enterprise and Automotive IoT Endpoints Will Be in Use in 2020." https://www.gartner.com/en/newsroom/press-releases/2019-08-29-gartner-says-5-8-billion-enterprise-and-automotive-io.

[3] statista, "Global IoT end-user spending worldwide 2017-2025." https://www.statista.com/statistics/976313/global-iot-market-size/#:~:text=The global market for Internet,around 1.6 trillion by 2025. (accessed Jan. 20, 2021).

[4] E. Pasqua, "https://iot-analytics.com/5-things-to-know-about-the-lpwan-market-in-2020/," 2021. https://iot-analytics.com/5-things-to-know-about-the-lpwan-market-in-2020/ (accessed Jan. 20, 2021).

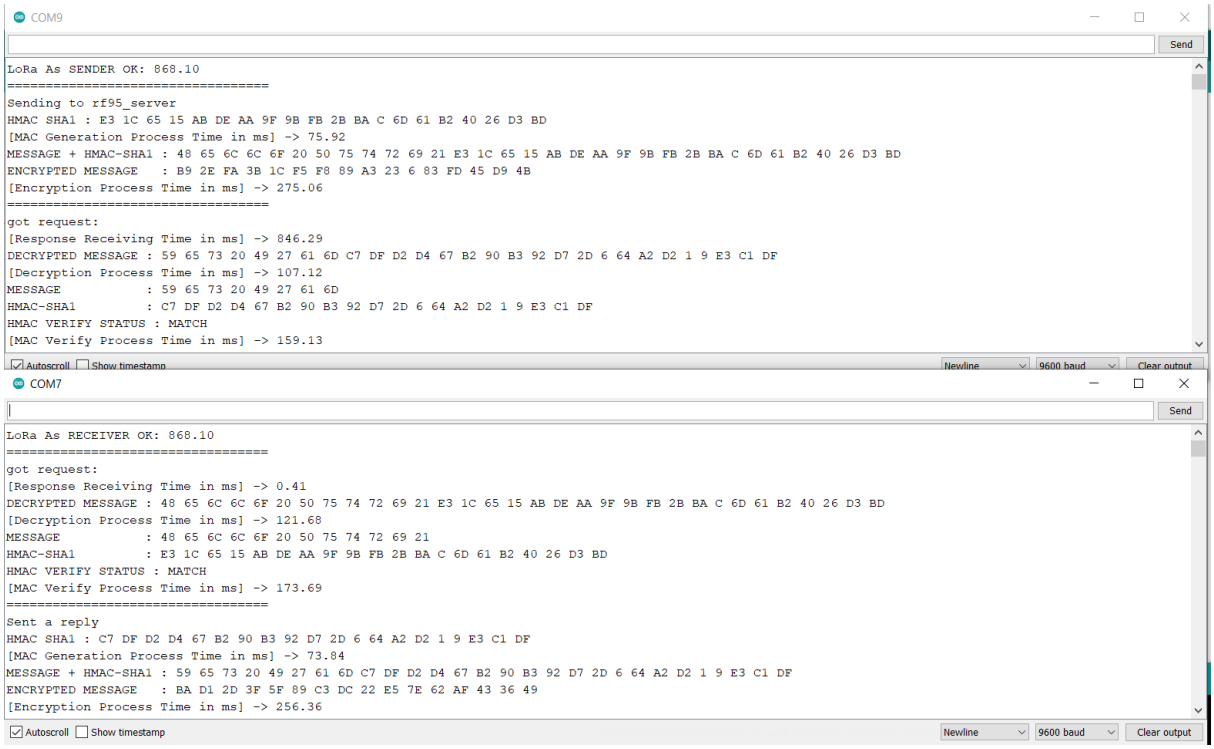
[5] Thethingsnetwork, "LoRaWAN Frequency Plans and Regulations by Country." https://www.thethingsnetwork.org/docs/lorawan/frequencies-by-country.html (accessed Jan. 20, 2020).
[6] S. Devalal and A. Karthikeyan, "LoRa technology-an overview," *2018 Second Int. Conf. Electron. Commun. Aerosp. Technol.*, no. Iceca, pp. 284–290, 2018.
[7] Semtech, "What is LoRa?," 2021. https://www.semtech.com/lora/what-is-lora (accessed Jan. 20, 2021).
[8] C. Bormann, M. Ersue, and A. Keranen, "Terminology for Constrained-Node Networks," *RFC7228*, 2014.
[9] R. Brian and D. Van Duren, *Practical Internet of Things Security*. Packt Publishing Ltd., 2016.
[10] R. Brian and D. Van Duren, *Practical Internet of Things Security*. Packt Publishing Ltd., 2016.
[11] W. Stallings, *CRYPTOGRAPHY AND NETWORK SECURITY PRINCIPLES AND PRACTICE SEVENTH EDITION GLOBAL EDITION British Library Cataloguing-in-Publication Data*. 2017.
[12] K. L. Tsai, Y. L. Huang, F. Y. Leu, I. You, Y. L. Huang, and C. H. Tsai, "AES-128 based secure low power communication for LoRaWAN IoT environments," *IEEE Access*, vol. 6, pp. 45325–45334, 2018, doi: 10.1109/ACCESS.2018.2852563.
[13] J. R. Naif, G. H. Abdul-Majeed, and A. K. Farhan, "Secure IOT System Based on Chaos-Modified Lightweight AES," *2019 Int. Conf. Adv. Sci. Eng. ICOASE 2019*, pp. 12–17, 2019, doi: 10.1109/ICOASE.2019.8723807.
[14] R. Krawczyk, H.; Bellare M.; Canetti, "HMAC: Keyed-Hashing for Message Authentication," *RFC2104*, 1997.
[15] K. Mittal, "Securing Communication in Class-0 IOT Devices," vol. 4, no. 5, pp. 90–95, 2016.
[16] A. Iqbal and T. Iqbal, "Low-cost and Secure Communication System for Remote Micro-grids using AES Cryptography on ESP32 with LoRa Module," *2018 IEEE Electr. Power Energy Conf. EPEC 2018*, pp. 1–5, 2018, doi: 10.1109/EPEC.2018.8598380.
[17] A. Fauzan, P. Sukarno, and A. A. Wardana, "Overhead Analysis of the Use of Digital Signature in MQTT Protocol for Constrained Device in the Internet of Things System," in *2020 3rd International Conference on Computer and Informatics Engineering (IC2IE)*, 2020, pp. 415–420, doi: 10.1109/IC2IE50715.2020.9274651.
[18] S. M. S. Hussain, S. M. Farooq, and T. S. Ustun, "Analysis and implementation of message authentication code (MAC) algorithms for GOOSE message security," *IEEE Access*, vol. 7, pp. 80980–80984, 2019, doi: 10.1109/ACCESS.2019.2923728.
[19] M. O. Ojo, S. Giordano, G. Procissi, and I. N. Seitanidis, "A Review of Low-End, Middle-End, and High-End Iot Devices," *IEEE Access*, vol. 6, no. 710583. IEEE, pp. 70528–70554, 2018, doi: 10.1109/ACCESS.2018.2879615.
[20] O. Hahm, E. Baccelli, H. Petersen, and N. Tsiftes, "Operating Systems for Low-End Devices in the Internet of Things: A Survey," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 720–734, 2016, doi: 10.1109/JIOT.2015.2505901.
[21] W. Stallings, *Network Security Essentials 4Th Edition*, 4th ed. Pearson Education, Inc., 2011.

Lampiran

```
Sketch uses 19668 bytes (7%) of program storage space. Maximum is 253952 bytes.
Global variables use 1418 bytes (17%) of dynamic memory, leaving 6774 bytes for local variables. Maximum is 8192 bytes.
```

```
Sketch uses 19704 bytes (7%) of program storage space. Maximum is 253952 bytes.
Global variables use 1464 bytes (17%) of dynamic memory, leaving 6728 bytes for local variables. Maximum is 8192 bytes.
```

Gambar 34 Penggunaan *flash* dan Memori Pengirim dan Penerima



Gambar 35 Hasil Komunikasi Pengirim dan Penerima