Implementasi Lightweight Cryptography untuk Keamanan Data pada Sistem Komunikasi LoRa

Jundi Abdillah Mundzir dan Wirawan
Departemen Teknik Elektro, Institut Teknologi Sepuluh Nopember (ITS)
e-mail: wirawan@ee.its.ac.id

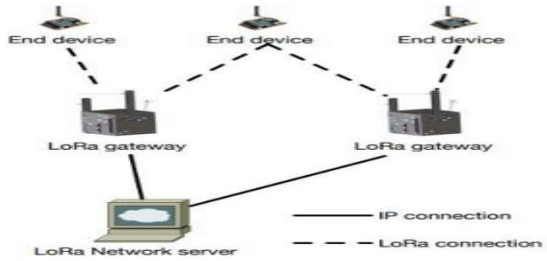
***Abstrak*—Seiring berkembangnya sistem komunikasi LoRa, muncul tantangan yang harus dihadapi berkaitan dengan keamanan data yang ditransmisikan. Akan tetapi, upaya implementasi keamanan data menghadapi tantangan terkait keterbatasan *resource* pada *constrained device* LoRa, misal kemampuan komputasi dan memori. Oleh karena itu, algoritma *lightweight cryptography* diperlukan pada *constrained device* untuk menjaga keamanan data selama proses transmisi menggunakan LoRa karena tidak menggunakan banyak *resource* dan ringan secara komputasi. Penelitian ini mengusulkan kombinasi algoritma *Advanced Encryption Standard* (AES) dan *Message Authentication Code* (MAC). Untuk proses enkripsi dan dekripsi pesan digunakan algortima kriptografi simetris *block cipher* tipe AES-128 dan AES-256, sedangkan untuk proses autentikasi pesan digunakan algoritma HMAC-SHA3 dan AES-CMAC. Kombinasi antara algoritma enkripsi dan autentikasi diharapkan mampu menjamin tiga aspek keamanan data, yaitu *confidentiality, integrity,* dan *authentication.* Berdasarkan hasil penelitian menunjukkan bahwa kombinasi algoritma *lightweight cryptography* yang diusulkan telah mampu menjaga tiga aspek keamanan data yaitu *integrity, authentication,* dan *confidentiality.* Semua algoritma *lightweight cryptography* yang diusulkan juga dapat diterapkan pada *constrained device* kelas 0. Kombinasi algoritma AES 128 dan CMAC memiliki total waktu komputasi dan sumber daya yang paling efisien.**

Kata Kunci*—AES, CMAC, HMAC, LoRa, *Constrained Device.

I. PENDAHULUAN

LORA merupakan salah satu media transmisi nirkabel yang banyak diterapkan pada *Internet of Things*. LoRa dirancang untuk dapat mentransmisikan data jarak jauh dengan konsumsi daya dan *bit rate* yang rendah. Salah satu tantangan implementasi LoRa pada *Internet of Things* adalah pengembangan sistem keamanan pada transmisi datanya. Hal ini karena tidak adanya proses enkripsi pada *payload* data [1]. Selain itu, terdapat beberapa ancaman keamanan pada komunikasi *point to pont* LoRa seperti *man in the middle attack* dan *eavesdropping.* Serangan-serangan ini sangat membahayakan karena dapat mencuri dan memanipulasi data yang ditransmisikan melalui LoRa.

Untuk membuat sistem kemanan data pada perangkat IoT perlu memperhatikan kapasitas perangkat. Hal ini karena salah satu karakteristik perangkat IoT adalah kapasitas *resource* yang terbatas atau sering dikenal sebagai *constrained device*. Keterbatasan *resource* dapat berupa terbatasnya ukuran memori, keterbatasan daya serta keterbatasan kemampuan komputasi. Hal ini menyebabkan terbatasnya kemampuan perangkat untuk dieksplorasi [2]. Oleh karena itu, diperlukan algoritma kriptografi ringan *(lightweight cryptography)* yang dapat diimplementasikan pada perangkat IoT untuk menjaga keamanan data selama proses transmisi menggunakan LoRa.



Gambar 1. Topologi LoRa.

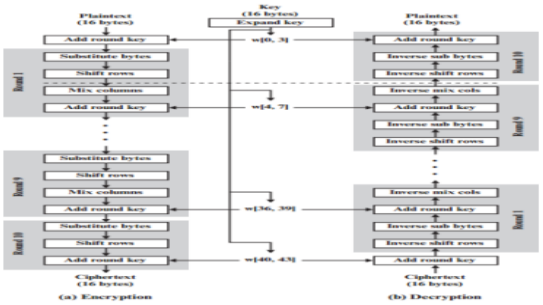
Pada penelitian ini penulis ingin mengkaji implementasi beberapa metode kemanan data dengan memanfaatkan algoritma *lightweight cryptography* pada proses transmisi data menggunakan LoRa. Algoritma kriptografi yang akan digunakan pada penilitian ini meliputi algoritma enkripsi AES dan algoritma autentikasi HMAC dan CMAC. Aspek keamanan yang akan diuji pada penilitian ini meliputi parameter *integrity*, *authentication* dan *confidentiality* [3]. Selain itu, peneliti juga akan mengkaji *overhead analysis* dari pengaruh penerapan algoritma *lightweight cryptography* yang berbeda pada *constrained device* IoT kelas 0.

II. TINJAUAN PUSTAKA

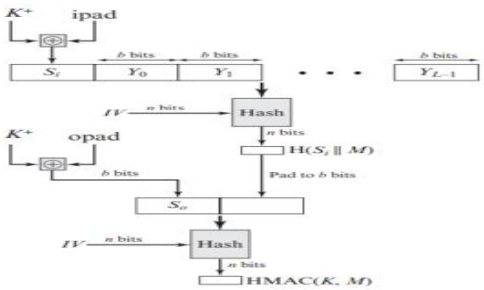
A. LoRa

LoRa atau "*Long Range*" merupakan sistem komunikasi *wireless* jarak jauh yang ditujukan untuk perangkat yang memiliki sumber daya terbatas. LoRa memiliki jangkauan komunikasi hingga 10 km dan memiliki *lifetime* baterai yang panjang, tetapi memiliki kecepatan data yang rendah [4]. *Bandwidth* LoRa berkisar antara 125 kHz sampai 500 kHz. Semakin besar *bandwidth* akan menyebabkan semakin tinggi kecepatan data, tetapi akan mengurangi jangkauan komunikasi. Daya transmisi LoRa berkisar antara -4 dBm sampai 20 dBm [5]. Jaringan LoRa dapat diaplikasikan pada *smart city*, *smart healthcare*, *smart farming* dan *environmental monitoring* . Pada Jaringan LoRaWAN terdapat beberapa komponen seperti *end device (*LoRa *node)*, LoRa *gateway*, dan LoRa *network server*. Topologi LoRa ditunjukkan oleh Gambar 1.

LoRa *node* atau *end device* merupakan sensor atau aktuator yang memiliki kemampuan komputasi terbatas dan beroperasi menggunakan daya baterai. LoRa *gateway* merupakan *intermediate device* yang berfungsi meneruskan *radio packets* yang datang dari LoRa *node* ke LoRa *network server* dan sebaliknya. LoRa *network server* berfungsi menerima semua paket yang berasal dari LoRa *gateway* serta



Gambar 2. Proses enkripsi dan dekripsi AES.



Gambar 3. Struktur HMAC.

bertanggung jawab atas pemrosesan paket dan analisis protokol [6].

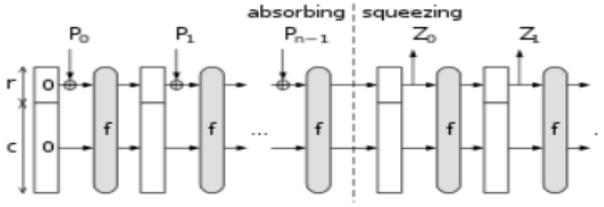
B. Ancaman Keamanan LoRa

Faktor keamanan menjadi salah satu tantangan yang harus diatasi dalam mendorong implementasi IoT secara luas. Risiko kemanan IoT dapat terjadi karena banyaknya entitas dan data yang terlibat sehingga dapat mengancam dan merugikan pengguna. Ancaman ini dapat berupa akses oleh orang yang tidak berwenang untuk mencuri data, memanipulasi data, dan menyalahgunakan informasi pribadi. Ancaman yang dapat memengaruhi entitas IoT sangat bervariasi, tergantung pada target serangan [4].

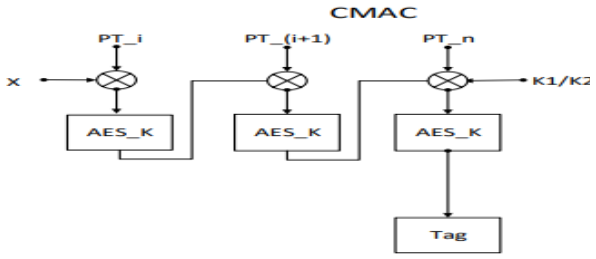
Pada sistem komunikasi LoRa terdapat beberapa ancaman serangan seperti *man in the middle attack* dan *eavesdropping*. *Man in the middle attack* merupakan serangan yang dilakukan oleh penyerang atau *attacker* dengan cara menyampaikan dan mengubah pesan saat dua pihak yang berwenang sedang berkomunikasi secara langsung[5]. *Eavesdropping attack* merupakan serangan yang pasif yang bertujuan untuk mengestrak data informasi yang ditransmisikan sehingga *plaintext* dapat dilihat. *Eavesdropping* dilakukan melalui proses *sniffing* terhadap data informasi (*payload*) yang dikirimkan melalui transmisi jaringan LoRa [3].

C. Constrained Device

Berdasarkan *resource*-nya, perangkat IoT dibagi menjadi dua yaitu *high-end IoT device* dan *low-end IoT device*. *High-end IoT device* merupakan perangkat IoT yang mampu menjalankan *operating system*, sedangkan *low-end IoT device* merupakan perangkat IoT yang kemampuan komputasinya dibatasi oleh *resource* yang tersedia pada perangkat. Pada *low-end IoT device* terdapat tiga kategori berdasarkan jumlah memori yang digunakan, yaitu kelas 0, kelas 1, dan kelas 2. Kelas 0 memiliki ukuran memori kurang dari 10 KiB dan ukuran flash kurang dari 100 KiB. Kelas 1 memiliki memiliki kapasitas memori sebesar 10 KiB dan flash 100 KiB. Kelas 2 memiliki kapasitas sumber daya terbesar daripada dua kelas sebelumnya. Kelas 2 memiliki



Gambar 4. Struktur *Sponge* SHA-3.



Gambar 5. Struktur AES-CMAC.

kapasitas memori sebesar 50 KiB dan flash sebesar 250 KiB [7]. Pada penelitian kali ini akan digunakan *constrained device* IoT kelas 2.

D. Advanced Encryption Standard

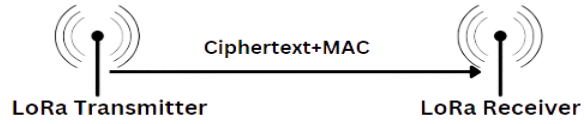
Advanced Encryption Standard (AES) merupakan *symmetric block chiper* yang mempunyai ukuran blok tetap. AES mendukung ukuran kunci 128, 192, dan 256 bits serta terdiri dari 10, 12, dan 14 *rounds* secara berturut-turut. Setiap *round* merupakan gabungan data dengan *round-key* yang diturunkan dari *encryption key*. Setiap *round* terdiri dari empat langkah pemrosesan yaitu *SubBytes*, *ShiftRows*, *MixColumns*, dan *AddRoundKey* [8], kecuali *round* terakhir yang hanya terdiri dari tiga proses yaitu *SubBytes*, *ShiftRows*, dan *AddRoundKey* [7]. Berikut ini adalah penjelasan dari masing-masing proses tersebut [8].

1. *SubBytes* merupakan subtitusi *byte* dengan menggunakan tabel subtitusi (s-box)
2. *ShiftRows* merupakan transposisi *byte* dengan cara menggeser baris blok data sesuai dengan *offsets* yang telah ditentukan.
3. *MixColumns* mengalikan setiap kolom dari blok data dengan polinomial modular dalam GF (28).
4. *AddRoundKey* merupakan proses untuk melakukan XOR antara *state* sekarang dengan *round key*.

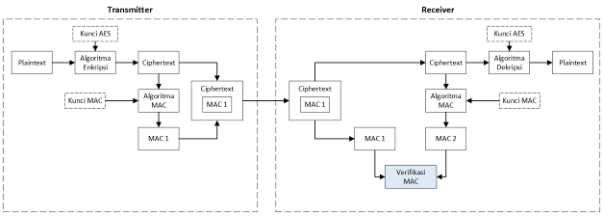
Proses dekripsi pada algoritma AES menggunakan transformasi byte pada invers cipher, yaitu *InvShiftRows*, *InvSubBytes*, *InvMixColumns*, dan *AddRoundKey*. *InvShiftRows* yaitu melakukan pergeseran hasil dari *Subbytes* secara *wrapping*. Pada *InvSubBytes* dilakukan konversi hasil dari *AddRoundKey* menggunakan nilai inverse s-box. Pada *InvMixColumns* dilakukan proses pengacakan data dengan melakukan perkalian antara matriks *public key* dengan matriks hasil *shiftrows* [4]. Gambar 2 menunjukkan proses enkripsi dan dekripsi AES.

E. Hash-based Message Authentication Code

Message Authentication Code (MAC) merupakan algoritma kriptografi yang digunakan untuk menjaga integritas dan autentikasi transmisi data [9]. Salah satu fungsi MAC yang sering digunakan adalah *Hash-based Message Authentication Code* (HMAC). HMAC merupakan mekanisme salah satu algoritma yang menjadi standar *The*



Gambar 6. Rancangan sistem komunkasi LoRa.



Gambar 7. Diagram blok sistem keamanan data.

National Institute of Standards and Technology (NIST). HMAC menghasilkan *Message Authentication Code* (MAC) dengan menggunakan operasi fungsi Hash yang dikombinasikan dengan *secret key* bersama [8]. Fungsi Hash berfungsi mengubah ukuran data yang acak ke dalam kumpulan data dengan panjang yang tetap[8]. Gambar 3 adalah struktur dari *Hash-based Message Authentication Code* (HMAC).

HMAC memiliki beberapa variasi tergantung fungsi Hash yang digunakan seperti SHA-1, SHA-2, dan SHA-3. *Secured Hash Algorithm* (SHA) dirancang untuk menjaga integritas data, dimana apabila terdapat sedikit saja perubahan pada data akan menghasilkan nilai hash yang jauh berbeda dari nilai aslinya[8]. Di antara beberapa versi SHA tersebut, SHA-3 merupakan standar terbaru yang diterbitkan oleh NIST pada tahun 2015. SHA-3 memiliki konstruksi struktur yang jauh berbeda bila dibandingkan dengan SHA-1 dan SHA-2 [8]. SHA-3 menggunakan konstruksi *sponge* seperti Gambar 4.

Konstruksi *sponge* terdiri dari modul *absorbing* dan modul *squeezing*. Sejumlah data akan diserap *(absorbed)* ke dalam *Hash state*, Kemudian data tersebut akan diperas *(squeezed)* keluar dari *Hash state* [10].

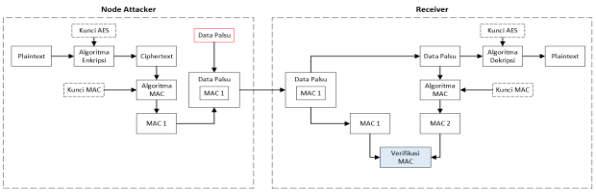
F. Cipher-based Message Authentication Code

Cipher-based Message Authentication Code (CMAC) merupakan algoritma kriptografi yang digunakan untuk autentikasi data. Pada algoritma CMAC sejumlah bit dari *plaintext* data dibagi ke dalam blok 128 bit. Apabila jumlah bit pada *plaintext* kurang dari 128 bit, *padding* akan ditambahkan untuk melengkapi jumlah bit agar menjadi 128 bit. *Plaintext* biasanya akan dienkripsi menggunakan algoritma AES dengan ukuran *secret key* 128 bit. Algoritma CMAC diimplementasikan dalam iterasi yang berurutan dengan *symmetric key* yang digunakan untuk menghasilkan dua *sub-key* [11]. Struktur AES-CMAC dapat dilihat pada Gambar 5.

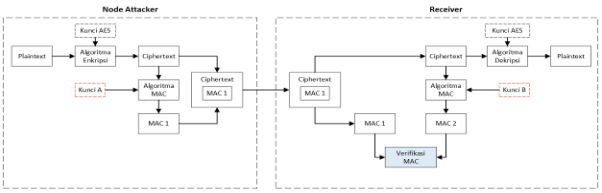
III. METODOLOGI

A. Perancangan Sistem

Pada penelitian ini akan dirancang sistem keamanan data pada transmisi LoRa dengan menggabungkan antara algoritma *Advanced Encryption Standard (*AES) dengan algoritma *Message Authentication Code* (MAC). Algoritma AES digunakan untuk proses enkripsi dan dekripsi data, sedangkan algoritma MAC digunakan untuk autentikasi data. Variasi algoritma AES yang akan digunakan yaitu AES-128



Gambar 8. Diagram blok pengujian *integrity*.



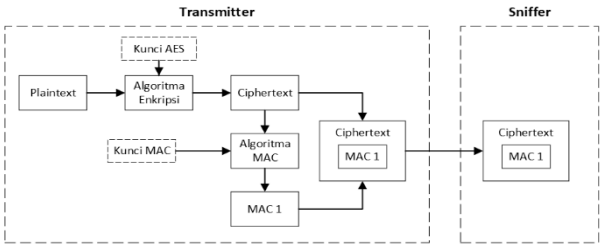
Gambar 9. Diagram blok pengujian *authentication.*

dan AES-256, sedangkan variasi algortima MAC yang akan digunakan adalah HMAC-SHA3 dan AES-CMAC. Variasi Metode keamanan dengan menggabungkan algoritma enkripsi AES dan algoritma autentikasi CMAC & HMAC diharapkan dapat meningkatkan keamanan data pada sistem komunikasi LoRa dengan menjaga aspek *confidentiality*, *integrity*, dan *authentication* pada data yang ditransmisikan.

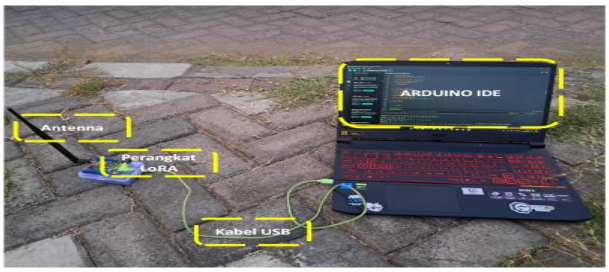
Sistem dibangun menggunakan skema komunikasi *point to point* antara LoRa *transmitter* dan LoRa *receiver*. Perangkat pada masing-masing sisi berupa *constrained device* kelas 0 yang telah disandingkan dengan modul LoRa. Untuk data yang ditransmisikan melalui LoRa berupa data teks dan gambar yang akan dienkripsi sehingga bentuk data tersebut bukan lagi *plaintext* melainkan *ciphertext*. Kemudian dari *ciphertext* tersebut akan dilakukan perhitungan nilai MAC untuk memastikan integritas pesan. Selanjutnya *ciphertext* dan nilai MAC akan dikirimkan secara bersamaan melalui LoRa. Skema sistem pada penelitian ini diilustrasikan oleh Gambar 6.

LoRa transmitter berperan sebagai pengirim data sedangkan LoRa receiver berperan sebagai penerima data. Pada sisi pengirim (LoRa *transmitter*) terdapat beberapa tahapan agar pesan dapat dikirimkan dengan aman. Tahap pertama yaitu melakukan enkripsi pesan berupa *plaintext*. *Plaintext* akan dienkripsi menggunakan algoritma AES sehingga menghasilkan *ciphertext* yang tidak bisa dibaca oleh pihak ketiga. Selanjutnya, dilakukan proses pembangkitan MAC dari *ciphertext* yang sudah dihasilkan sebelumnya. Setelah nilai MAC 1 berhasil dibangkitkan, dilakukan proses menggabungkan *ciphertext* dan nilai MAC 1 menjadi satu data. Data hasil proses penggabungan ini selanjutnya disebut dengan *payload* data dan akan dikirim kepada penerima (LoRa *receiver)* melalui transmisi LoRa.

Gambar 7 menunjukkan diagram blok sistem keamanan data. Pada sisi penerima (LoRa *receiver*), ada beberapa tahapan yang dilakukan agar *receiver* dapat mengetahui *plaintext* pesan dan memastikan keaslian pesan. Pada tahap awal, *payload* data yang diterima oleh *receiver* akan dilakukan proses untuk memisahkan *ciphertext* dengan nilai MAC 1 yang diterima dari *transmitter*. Pada *ciphertext* akan dilakukan proses dekripsi menggunakan algoritma AES sehingga didapatkan *plaintext* yang dapat dibaca oleh penerima. Selain itu, pada *ciphertext* juga akan dilakukan proses pembangkitan MAC pada sisi *receiver* sehingga akan dihasilkan nilai MAC 2. Kemudian untuk mengetahui keaslian pesan yang diterima, dilakukan proses verifikasi



Gambar 10. Diagram blok pengujian *confidentiality*.



Gambar 11. Implementasi perangkat.

MAC dengan membandingkan nilai MAC 1 yang dibangkitkan pada *transmitter* dengan nilai MAC 2 yang dibangkitkan pada sisi *receiver*. Apabila proses verifikasi nilai MAC 1 dan MAC 2 menunjukan nilai yang sama, maka dapat dipastikan bahwa pesan yang diterima merupakan pesan asli dari pengirim. Sebaliknya apabila nilai MAC 1 dan MAC 2 berbeda, maka dapat dipastikan bahwa pesan yang diterima oleh *receiver* adalah palsu.

B. Pengujian Keamanan

Pengujian keamanan dilakukan untuk memastikan sistem telah memenuhi mekanisme keamanan *confidentiality*, *integrity*, dan *authentication*. Diagram blok pengujian *integrity* ditunjukkan oleh Gambar 8.

Integrity atau integritas data diuji dengan melihat kemampuan sistem untuk mendeteksi modifikasi data yang dilakukan oleh *node attacker*. Pengujian integritas dilakukan dengan menggunakan skenario dimana data yang dikirimkan diubah setelah proses generate MAC berhasil sehingga data yang diterima *receiver* tidak sesuai dengan data asli. Keberhasilan deteksi integritas dinilai berdasarkan kemampuan sistem untuk mendeteksi adanya perubahan pada data. Diagram blok pengujian *authentication* dapat dilihat pada Gambar 9.

Authentication atau autentikasi diuji dengan melihat kemampuan sistem dalam memverifikasi sumber data yang sah. Pengujian autentikasi dilakukan dengan menggunakan skenario dimana *node attacker* berusaha mengubah nilai MAC dengan menggunakan kunci autentikasi yang berbeda dari aslinya. Kunci yang berbeda menyebabkan nilai MAC yang dihasilkan juga berbeda dari aslinya. MAC hasil pembangkitan di *node attacker* ini kemudian digabungkan dengan data asli menjadi satu *payload* data untuk dikirimkan ke *receiver*. Selanjutnya *receiver* akan menerima *payload* data yang dikirim oleh *attacker node*. *Receiver* yang sudah diimplementasikan metode keamanan kriptografi kemudian memverifikasi keaslian data yang dikirimkan oleh *node attacker*. Proses verifikasi dilakukan dengan membandingkan nilai MAC yang dibangkitkan di *receiver* dengan nilai MAC yang dikirimkan oleh *node attacker*.

Keberhasilan autentikasi dinilai berdasarkan kemampuan sistem untuk mendeteksi perubahan kunci dan validasi keaslian data. Hasil verifikasi MAC menunjukkan MATCH berarti kunci autentikasi antara penerima dan pengirim sama, sedangkan hasil verifikasi MAC menunjukkan UNMATCH berarti kunci autentikasi antara pengirim dan penerima berbeda.

Gambar 12. Hasil proses *sniffing* pada pengujian tanpa metode keamanan.

Gambar 13. Hasil proses *sniffing* pada pengujian metode keamanan.

Pengujian *confidentiality* bertujuan untuk memastikan bahwa data yang ditransmisikan melalui komunikasi LoRa tidak dapat dilihat *plaintext*-nya oleh penyerang sehingga kerahasian data akan terjamin. Pada pengujian ini akan ada *node sniffer* yang berperan untuk menyadap data yang ditransmisikan. Diagram blok pengujian *confidentiality* ditunjukkan oleh Gambar 10.

Aspek *confidentiality* atau kerahasiaan data diuji dengan melihat kemampuan algoritma enkripsi AES-128 dan AES-256 dalam menjaga data agar tidak dapat dibaca oleh pihak yang tidak berwenang. Pengujian *confidentiality* dilakukan dengan cara menangkap data yang sedang ditransmisikan oleh *transmitter* menggunakan *node sniffer* yang tidak memiliki kunci enkripsi. Data yang tertangkap oleh *sniffer* dianalisis untuk melihat apakah data tersebut dapat didekripsi tanpa kunci yang benar. Keberhasilan pengujian ini dinilai berdasarkan kemampuan *node sniffer* untuk membaca isi pesan yang sebenarnya.

C. Pengujian Overhead

Urutan Pengujian *overhead* dilakukan pada perangkat LoRa setelah diimplementasikan berbagai algoritma *lightweight cryptography*. Overhead yang dimaksud merujuk pada beban tambahan yang dikenakan pada perangkat akibat penggunaan algoritma kriptografi yang berbeda. Parameter *overhead* yang diujikan pada tugas akhir ini yaitu penggunaan *flash*, penggunaan memori, lama waktu enkripsi dan dekripsi pesan, waktu proses pembangkitan MAC, dan waktu verifikasi MAC.

IV. HASIL DAN PEMBAHASAN

A. Implementasi Perangkat

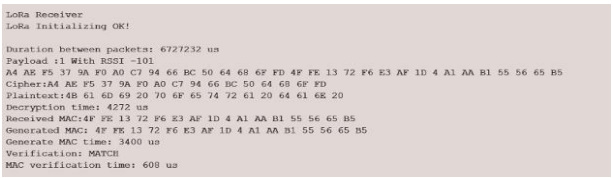
Komponen perangkat yang digunakan terdiri dari perangkat LoRa yang sudah terhubung ke antenna dan Arduino IDE. Berikut merupakan dokumentasi perangkat

Tabel 1.
Hasil Pengujian *Confidentiality*

No.	Metode	Kondisi Pengujian
1.	Tanpa Metode Keamanan Kriptografi	*Sniffer* dapat melihat *plaintext*
2.	AES 128 + CMAC	*Sniffer* hanya melihat *ciphertext*
3.	AES 256 + CMAC	*Sniffer* hanya melihat *ciphertext*
4.	AES 128 + HMAC	*Sniffer* hanya melihat *ciphertext*
5.	AES 256 + HMAC	*Sniffer* hanya melihat *ciphertext*

Tabel 1.
Hasil Pengujian *Authentication*

No.	Metode Keamanan	Kunci MAC	Kondisi Verifikasi MAC Pengujian
1.	AES 128 + CMAC	Sama	Match
		Berbeda	Unmatch
2.	AES 128 + HMAC	Sama	Match
		Berbeda	Unmatch
3.	AES 256 + CMAC	Sama	Match
		Berbeda	Unmatch
4.	AES 256 + HMAC	Sama	Match
		Berbeda	Unmatch



Gambar 11. Hasil pengujian *authentication* apabila kunci sama.

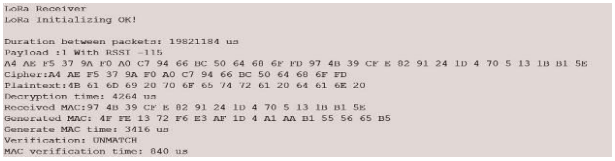
yang digunakan selama proses pengujian. Gambar 11 menunjukkan kondisi pengujian, dimana selama proses pengujian perangkat LoRa diletakkan sejajar dengan tanah. Hal ini untuk memastikan bahwa sinyal yang dipancarkan dan diterima tidak terhalang oleh objek lain dan untuk menjaga konsistensi dalam pengujian.

B. *Hasil Pengujian Confidentiality*

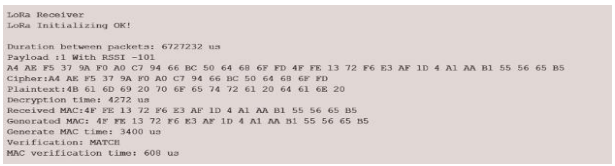
Terdapat dua skenario pada pengujian *confidentiality*, yaitu tanpa metode keamanan dan menggunakan metode keamanan. Gambar 12 dan Gambar 13 merupakan hasil pengujian *confidentiality*.

Pada pengujian tersebut, *sniffer* berhasil menangkap data yang dikirimkan oleh *transmitter* melalui sistem komunikasi LoRa. Dalam proses pengujian ini, data yang ditransmisikan melalui LoRa berupa *byte* yang merepresentasikan teks sumpah pemuda. Setiap *payload* mengandung informasi 16 *byte* teks sumpah pemuda sehingga untuk *payload* 1 seharusnya berisi pesan “kami poetra dan ”. Untuk pengujian tanpa metode keamanan, *transmitter* hanya mengirimkan *plaintext* 16 *byte* untuk setiap payload tanpa adanya enkripsi dan penambahan nilai MAC. Sedangkan untuk pengujian dengan menggunakan metode keamanan, *transmitter* mengirimkan 16 *byte ciphertext* ditambah nilai MAC untuk setiap *payload.*

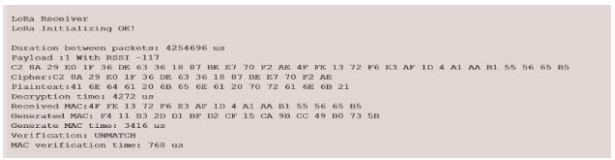
Selanjutnya data yang berhasil ditangkap oleh *sniffer* akan dilakukan proses *encoding* dari data *byte* ke teks. Terlihat dari hasil *encoding* bahwa *plaintext* hanya terlihat pada pengujian tanpa metode keamanan, sedangkan dengan metode keamanan yang diusulkan *byte* dikonversi menjadi karakter yang tidak terbaca. Hal ini membuktikan bahwa usulan metode keamanan kriptografi yang diterapkan pada sistem komunikasi LoRa dapat menjaga kerahasiaan data



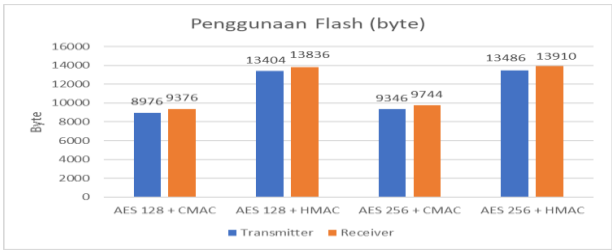
Gambar 15. Hasil pengujian *authentication* apabila kunci berbeda.



Gambar 16. Hasil pengujian *integrity* pada data apabila pesan tidak dimodifikasi.



Gambar 17. Hasil pengujian *integrity* pada data teks apabila pesan dimodifikasi.



Gambar 18. Hasil pengujian *flash* dalam *byte.*

yang dikirimkan. Tabel 1 memuat detail keberhasilan pengujian untuk semua metode keamanan yang diterapkan pada sistem komunikasi LoRa.

C. *Hasil Pengujian Authentication*

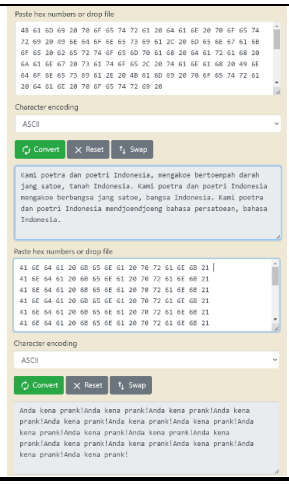
Terdapat dua skenario pada pengujian *authentication*, yaitu dengan kunci autentikasi yang sama antara pengirim dan penerima, serta dengan kunci autentikasi yang berbeda antara pengirim dan penerima. Gambar 14 dan Gambar 15 merupakan hasil pengujian *authentication*.

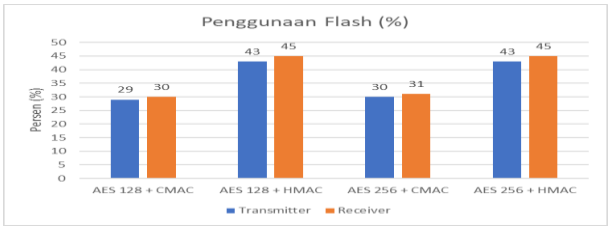
Berdasarkan hasil pengujian *authentication*, sistem telah berhasil mendeteksi apabila ada kiriman *payload* yang bukan berasal dari pihak yang berwenang. Hasil verifikasi UNMATCH menunjukkan bahwa pengirim memiliki kunci yang berbeda dari kunci penerima, sedangkan hasil verifikasi MATCH menunjukkan kunci yang sama digunakan anatara pengirim dan penerima. Detail keberhasilan pengujian aspek *authentication* pada metode keamanan yang diusulkan dapat dilihat pada Tabel 2.

D. *Hasil Pengujian Integirty*

Berdasarkan hasil pengujian *integrity*, semua metode keamanan yang diusulkan mampu mendeteksi adanya perubahan data yang terjadi. Gambar 16 merupakan hasil pengujian *integrity* untuk kondisi pesan tidak dimodifikasi setelah proses pembangkitan MAC berhasil dilakukan. Gambar 16 menunjukkan kondisi verifikasi MAC bernilai MATCH yang berarti sistem berhasil mendeteksi bahwa data yang dikirim tidak dilakukan modifikasi. Berikut merupakan hasil pengujian *integrity* untuk kondisi pesan yang dimodifikasi setelah proses pembangkitan MAC berhasil dilakukan.

Tabel 3.
Hasil Pengujian *Integrity*

No.	Pesan	Verifikasi MAC	Data yang diterima
1.	Tidak dimodifikasi setelah proses generasi MAC berhasil	Match	
2.	Dimodifikasi setelah proses generasi MAC berhasil	Unmatch	



Gambar 19. Hasil pengujian *flash* dalam persen.

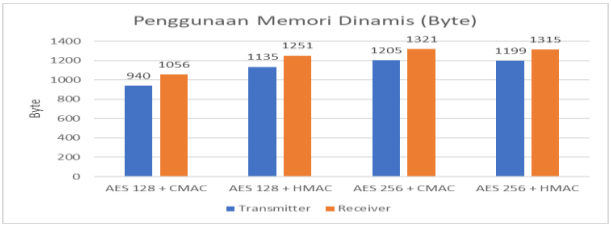
Gambar 17 menunjukkan kondisi verifikasi MAC bernilai UNMATCH yang berarti sistem berhasil mendeteksi bahwa data yang dikirim telah dilakukan modifikasi. Pada pengujian ini, data teks terdapat 16 *payload* data, sedangkan untuk data gambar terdapat 81 *payload* data. Masing-masing *payload* berisi 16 *byte* data informasi. Semua *byte* yang didapatkan dari hasil pengujian di atas kemudian dilakukan *encoding* untuk melihat informasi yang dikirim oleh penerima. Rangkuman keberhasilan sistem keamanan yang diusulkan untuk integritas data dapat dilihat pada Tabel 3.

E. Penggunaan Flash

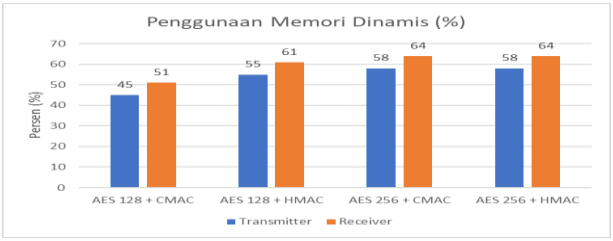
Penggunaan *flash* diukur untuk mengetahui seberapa besar memori *flash* yang diperlukan untuk menyimpan kode algoritma keamanan kriptografi yang diusulkan. Gambar 18 dan Gambar 19 adalah hasil pengukuran penggunaan *flash*. Dari hasil pengukuran, Algoritma AES 128 + CMAC memiliki penggunaan memori flash paling sedikit, sedangkan Algoritma AES 256 + HMAC memiliki penggunaan memori flash paling banyak. Hal ini menunjukkan bahwa Algoritma AES 128 + CMAC lebih efisien dalam hal kapasitas penyimpanan program, memungkinkan lebih banyak ruang untuk kode tambahan atau data lainnya.

F. Penggunaan Memori Dinamis

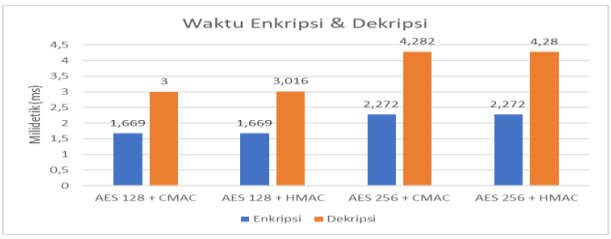
Pengukuran ini mengacu pada jumlah memori dinamis (RAM) yang digunakan selama eksekusi program. Gambar 20 dan Gambar 21 menunjukkan hasil pengukuran penggunaan memor dinamis. Hasil pengukuran menunjukkan bahwa Algoritma AES 128 + CMAC menggunakan kapasitas memori dinamis paling sedikit, sedangkan Algoritma AES 256 + CMAC menggunakan kapasitas memori dinamis paling banyak. Ini menandakan bahwa Algoritma AES 128 + CMAC lebih efisien dalam hal penggunaan memori dinamis selama eksekusi program.



Gambar 20. Hasil pengujian memori dinamis dalam *byte.*



Gambar 21. Hasil pengujian memori dinamis dalam persen.



Gambar 22. Hasil pengujian waktu enkripsi dan dekripsi.

G. Waktu Enkripsi dan Dekripsi

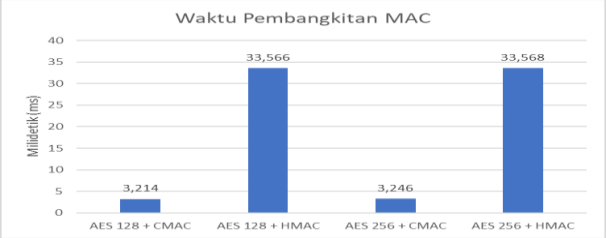
Pengukuran ini mengacu pada waktu yang diperlukan untuk mengenkripsi dan mendekripsi suatu pesan dengan panjang 16 *byte*. Nilai waktu enkripsi dan dekripsi diambil dari nilai rata-rata dalam satu kali perulangan pengiriman data yang terdiri dari 16 *payload*. Gambar 22 adalah hasil pengukuran dalam milidetik (ms).

Waktu enkripsi dan dekripsi yang cepat penting untuk memastikan responsivitas sistem. Algoritma AES 128 + CMAC menunjukkan waktu enkripsi dan dekripsi yang paling cepat, masing-masing 1,669 ms dan 3 ms. Sedangkan algoritma AES 256 + CMAC menunjukkan waktu enkripsi dan dekripsi yang paling lambat, yaitu masing-masing 2,272 ms dan 4,282 ms. Ini menunjukkan bahwa algoritma AES 128 + CMAC lebih efisien dalam hal waktu komputasi sehingga cocok untuk aplikasi dengan kebutuhan waktu pemrosesan yang cepat.

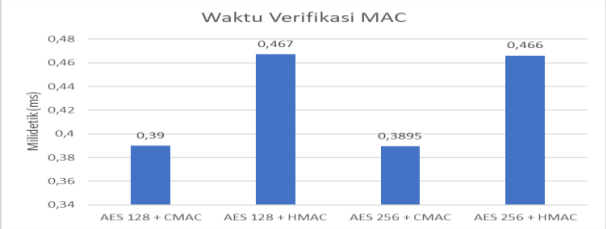
H. Waktu Pembangkitan MAC

Waktu pembangkitan MAC mengukur waktu yang diperlukan untuk menghasilkan kode autentikasi pesan (MAC) yang digunakan untuk memastikan integritas dan autentisitas pesan. Nilai waktu pembangkitan MAC diambil dari nilai rata-rata dalam satu kali perulangan pengiriman data yang terdiri dari 16 *payload.* Gambar 23 adalah hasil pengukuran dalam milidetik (ms).

Hasil pengukuran menunjukkan bahwa Algoritma AES 128 + CMAC memiliki waktu pembangkitan MAC yang paling cepat, yaitu 3,214 ms. Sedangkan untuk waktu pembangkitan MAC yang paling lambat adalah algoritma AES 256 + HMAC dengan memakan waktu 33,568 ms.



Gambar 23. Hasil pengujian waktu pembangkitan MAC.



Gambar 24. Hasil pengujian waktu verifikasi MAC.

I. Waktu Verifikasi MAC

Pengujian ini mengukur waktu yang diperlukan untuk melakukan verifikasi bahwa MAC yang diterima sesuai dengan MAC yang dihasilkan oleh pengirim. Nilai waktu verifikasi MAC diambil dari nilai rata-rata dalam satu kali perulangan pengiriman data yang terdiri dari 16 *payload*. Gambar 24 menunjukkan hasil pengukuran dalam milidetik (ms).

Waktu verifikasi MAC yang cepat penting untuk memastikan integritas pesan yang diterima dengan cepat. Hasil pengukuran menunjukkan bahwa Algoritma autentikasi CMAC memiliki waktu verifikasi yang paling cepat daripada algoritma autentikasi HMAC.

V. KESIMPULAN

Penerapan algoritma *lightweight cryptography* pada sistem komunikasi LoRa berhasil mengamankan data dengan baik, terbukti dari kemampuannya dalam menjaga integritas, autentikasi, dan kerahasiaan data yang dikirimkan melalui jaringan komunikasi LoRa. Algoritma AES 128 menunjukkan kinerja waktu komputasi enkripsi dan dekripsi yang lebih cepat dibandingkan dengan AES 256. Algoritma CMAC juga lebih cepat dalam pembangkitan dan verifikasi MAC dibandingkan HMAC.

Penelitian ini menunjukkan bahwa metode keamanan yang diusulkan sangat sesuai untuk diterapkan pada perangkat dengan sumber daya terbatas seperti *constrained device* kelas 0, dimana kombinasi algoritma AES 128 + CMAC menggunakan paling sedikit sumber daya. Meskipun terdapat perbedaan dalam tingkat keamanan antara berbagai algoritma yang digunakan, penggunaan AES 128 + CMAC telah terbukti memberikan keseimbangan yang baik antara keamanan, efisiensi sumber daya dan waktu komputasi.

DAFTAR PUSTAKA

[1] R. I. Lestari, “Implementasi Pengamanan pada Jaringan LoRaWAN untuk Mengatasi Serangan Sniffing dengan Menggunakan Metode Digital Signature,” Departemen Informatika, Universitas Telkom, Bandung, 2020.

[2] P. A. Windya, “Penerapan Keamanan Komunikasi pada Jaringan LoRa(Long Range) Menggunakan Algoritma Advanced Encryption Standard(AES) dan Message Authentication Code(MAC),” Departemen Informatika, Universitas Telkom, Bandung, 2021.

[3] P. A. Windya, V. Suryani, and A. A. Wardana, “Sniffing prevention in lora network using combination of advanced encryption standard (aes) and message authentication code (mac),” in *2021 International Conference Advancement in Data Science, E-learning and Information Systems (ICADEIS)*, Nusa Dua, Bali, IEEE, Oct. 2021, pp. 1–5, doi: 10.1109/ICADEIS52521.2021.9702081.

[4] A. Rachmayanti and W. Wirawan, “Implementasi algoritma advanced encryption standard (AES) pada jaringan internet of things (IoT) untuk mendukung smart healthcare,” *J. Tek. ITS*, vol. 11, no. 3, pp. 217–222, 2022, doi: 10.12962/j23373539.v11i3.97042.

[5] N. Andiyani, A. Kusyanti, and R. A. Siregar, “Implementasi man in the middle attack pada algoritme Blake2s berbasis LoRa,” *Ikraith-Informatika*, vol. 6, no. 2, pp. 98–103, 2022.

[6] Q. Zhou, K. Zheng, L. Hou, J. Xing, and R. Xu, “Design and implementation of open LoRa for IoT,” *IEEE Access*, vol. 7, pp. 100649–100657, 2019, doi: 10.1109/ACCESS.2019.2930243.

[7] 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)*, Yogyakarta, IEEE, Sep. 2020, pp. 415–420, doi: 10.1109/IC2IE50715.2020.9274651.

[8] V. K. Sarker, T. N. Gia, H. Tenhunen, and T. Westerlund, “Lightweight security algorithms for resource-constrained iot-based sensor nodes,” in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, Dublin, Irlandia, IEEE, Jun. 2020, pp. 1–7, doi: 10.1109/ICC40277.2020.9149359.

[9] R. Dilli and P. C. S. Reddy, “Implementation of security features in manets using sha-3 standard algorithm,” in *2016 International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS)*, Bengaluru, India, IEEE, Oct. 2016, pp. 455–458, doi: 10.1109/CSITSS.2016.7779410.

[10] J. Li, L. Wu, and X. Zhang, “An efficient hmac processor based on the sha-3 hash function,” in *2017 IEEE 12th International Conference on ASIC (ASICON)*, Guiyang, China, IEEE, Oct. 2017, pp. 252–255, doi: 10.1109/ASICON.2017.8252460.

[11] S. J. Hussain Pirzada, A. Murtaza, M. N. Hasan, T. Xu, and L. Jianwei, “The Implementation of Aes-cmac authenticated encryption algorithm on fpga,” in *2019 IEEE 2nd International Conference on Computer and Communication Engineering Technology (CCET)*, Beijing, China, IEEE, Aug. 2019, pp. 193–197, doi: 10.1109/CCET48361.2019.8989202.