Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) DOI: 10.25126/jtiik.202296766
Vol. 9, No. 7, Desember 2022, hlm. 1647-1654 p-ISSN: 2355-7699
Akreditasi KEMENRISTEKDIKTI, No. 36/E/KPT/2019 e-ISSN: 2528-6579

IMPLEMENTASI ALGORITME CLEFIA 128-BIT UNTUK PENGAMANAN MODUL KOMUNIKASI LORA

Muhammad Fadhli Iman[1], Ari Kusyanti*[2], Rakhmadhany Primananda[3]

[1,2,3]Universitas Brawijaya, Malang
Email: [1]fadhlimuhammadiman@gmail.com, [2]ari.kusyanti@ub.ac.id, [3]rakhmadhany@ub.ac.id
*Penulis Korespondensi

(Naskah masuk: 08 Desember 2022, diterima untuk diterbitkan: 26 Desember 2022)

Abstrak

LoRa, yang berarti "*Long Range*" adalah sebuah sistem komunikasi nirkabel jarak jauh, yang dipromosikan oleh *LoRa Alliance*. LoRa merupakan sebuah protokol pengiriman data berdaya rendah yang menggunakan spektrum radio. LoRa memiliki tujuan agar dapat digunakan pada sebuah perangkat bertenaga baterai yang tahan lama, di mana konsumsi energi adalah yang paling utama. Namun, pada dasarnya modul komunikasi LoRa tidak memiliki sistem keamanan untuk melindungi pesan saat melakukan transfer data, dan hal itu menyebabkan modul LoRa sangat rentan terhadap serangan yang dilakukan oleh pihak yang tidak bertanggung jawab. Salah satu cara untuk mengamankan modul komunikasi LoRa adalah dengan menerapkan sistem keamanan yang dapat mencegah pihak yang tidak bertanggung jawab membaca pesan yang dikirim, yaitu dengan menggunakan metode enkripsi. Metode enkripsi yang akan digunakan adalah algoritme Clefia 128-bit. Dari hasil pengujian yang dilakukan pada penelitian, algoritme Clefia 128-bit terbukti dapat mencegah serangan *sniffing* dan *known-plaintext-attack*. Pada pengujian serangan *sniffing*, penyerang hanya mampu mendapatkan pesan yang masih berbentuk *ciphertext* sehingga pesan asli tidak dapat dibaca. Pada pengujian serangan *known-plaintext-attack*, penyerang gagal menemukan *key* asli yang digunakan pada algoritme Clefia 128-bit, sehingga penyerang gagal melakukan serangan ke sistem.

Kata kunci: *LoRa, Enkripsi, Algoritme Clefia 128-Bit*

IMPLEMENTATION OF CLEFIA 128-BIT ALGORITHM FOR LoRa COMMUNICATION MODULE SECURITY

Abstract

LoRa, which means "Long Range" is a long range wireless communication system, promoted by the LoRa Alliance. LoRa is a low-power data transmission protocol that uses the radio spectrum. LoRa aims to be used in a battery-powered device that lasts a long time, where energy consumption is the most important. However, basically the LoRa communication module does not have a security system to protect messages when transferring data, and that causes the LoRa module to be very vulnerable to attacks by irresponsible parties. One way to secure the LoRa communication module is to implement a security system that can prevent irresponsible parties from reading the messages sent, by using the encryption method. The encryption method that will be used is the 128-bit Clefia algorithm. From the results of the tests carried out in the study, the Clefia 128-bit algorithm is proven to be able to prevent sniffing and known-plaintext-attack attacks. In sniffing attacks test, attackers are only able to get messages that are still in the form of ciphertext so that the original message cannot be read. In the known-plaintext-attack attack test, the attacker failed to find the original key used in the 128-bit Clefia algorithm, so the attacker failed to attack the system.

Keywords: *LoRa, Encryption, Clefia 128-Bit Algorithm*

1. PENDAHULUAN

Di masa modern sekarang, IoT (*Internet of Things*) bertumbuh dengan sangat pesat. Banyak aplikasi IoT yang digunakan untuk berbagai sektor seperti industri, peternakan, pertanian, keamanan, dll. Pesatnya pertumbuhan IoT ini juga turut memicu akan perkembangan dari metode transmisi yang dapat digunakan, salah satunya adalah LoRa (Eridani, et al., 2019). Hampir semua perangkat IoT menggunakan baterai. Dengan estimasi bahwa perangkat IoT yang

akan digunakan mencapai 30 miliar pada tahun 2025 (Yousuf, et al., 2018), tentunya akan sangat dibutuhkan teknologi yang memiliki penggunaan baterai yang rendah untuk mengurangi kebutuhan energi, seperti LoRa.

LoRa merupakan sebuah protokol pengiriman data yang menggunakan spektrum radio dan memiliki penggunaan daya yang rendah (Wixted, et al., 2016). Terdapat beberapa keunggulan yang dimiliki oleh LoRa, yaitu penggunaan daya baterai yang hemat, biaya rendah, dan dapat mengakomodasi jangkauan komunikasi hingga 2 km (Khutsoane, et al., 2017). Transmisi data yang tahan terhadap *noise*, tingkat konsumsi daya yang rendah, hingga sanggup mengakomodasi jarak komunikasi antar modul yang jauh menjadi keuntungan tersendiri yang dimiliki LoRa pada perkembangan teknologi seperti WSN (*Wireless Sensor Network*) (Susanto, et al., 2019). Namun, modul LoRa belum mempunyai pengamanan data yang sesuai. LoRa merupakan sebuah modul komunikasi, hal ini berarti LoRa akan berhubungan dengan banyak perangkat yang akan saling berkomunikasi secara *end-to-end*. Dengan tidak dimilikinya sistem keamanan, akan sangat berbahaya jika modul LoRa menerima serangan dari pihak yang tidak bertanggungjawab.

Pada penelitian sebelumnya dengan judul "*Exploring The Security Vulnerabilities of LoRa*" (Aras, et al., 2017), meneliti perihal potensi kerentanan keamanan pada LoRa. Penelitian ini melakukan analisis terhadap *network stack* LoRa dan mencari kemungkinan kerentanan yang ada pada perangkat LoRa terhadap berbagai jenis serangan menggunakan perangkat keras komersial. Pengujian serangan yang dilakukan pada penelitian ini adalah *compromising device* and *network keys*, *jamming attacks*, dan *replay attack*. Hasil dari penelitian ini menunjukkan bahwa transmisi jarak jauh LoRa memiliki kerentanan keamanan yang serius dan bisa dieksploitasi pihak-pihak yang tidak bertanggung jawab. Sehingga diperlukannya penggunaan sebuah algoritme kriptografi yang bisa memberikan keamanan di modul komunikasi LoRa.

Pada penelitian lainnya dengan judul "*Low-cost and Secure Communication System for Remote Micro-grids using AES Cryptography on ESP32 with LoRa Module*" (Iqbal & Iqbal, 2018), membahas mengenai pengimplementasian algoritme AES (*Advance Encryption Standard*) pada ESP32 dengan modul LoRa untuk mengamankan komunikasi nirkabel untuk *micro-grids*. Akan tetapi, terdapat kelemahan yang ada pada algoritme AES. Pada penelitian yang dilakukan oleh (Bogdanov, et al., 2011), terdapat serangan yang dapat membuktikan adanya kecacatan pada proses komputasi di dalam algoritme AES, yaitu serangan *Biclique Cryptanalysis*. Dari hasil percobaan serangan tersebut diketahui bahwa transformasi *round* pada AES tidak dirancang untuk memiliki ketahanan yang kuat terhadap beberapa jenis serangan untuk jumlah *round* yang lebih kecil. Dari hasil serangan tersebut mereka berhasil membagi *cipher* menjadi 3 bagian. Dan juga terdapat fakta bahwa ketika transformasi *MixColumns* algoritme AES dihilangkan pada *round* terakhir, hal itu dapat membantu untuk merancang serangan pada *round* lainnya.

Berdasarkan pemaparan latar belakang di atas, maka dibutuhkan solusi algoritme kriptografi yang bisa menutup celah keamanan tersebut, yaitu algoritme Clefia. Algoritme Clefia merupakan sebuah block cipher ringan yang dikembangkan oleh Sony. Algoritme ini mendukung ukuran blok 128-bit menggunakan kunci simetris dengan tiga jenis yang berbeda, yaitu 128-bit, 192-bit, dan 256-bit. Algoritme Clefia didesain atas dasar efisiensi akan tetapi tetap memerhatikan aspek keamanan. Selain itu, algoritme clefia telah ditetapkan sebagai algoritme yang memiliki standar untuk melakukan pervasive computing di mana berkriteria guna diterapkan di device yang mempunyai sumber daya terbatas lewat ISO/IEC 29192-2 (Permata & A'mas, 2016). Pada studi yang dilakukan oleh (Sugawara, et al., 2008) yang membandingkan kinerja ASIC antar algoritme *block cipher*, yaitu Clefia, AES, Camellia, SEED, CAST-128, MISTY1, dan TDEA. Dari pengujian efisiensi (Kbps/gates) yang dilakukan, didapatkan hasil bahwa algoritme Clefia lebih baik dibandingkan algoritme lainnya. Efisiensi dari algoritme Clefia menghasilkan nilai 268,63 Kbps/gates. Lebih tinggi dibandingkan AES yang menghasilkan nilai 175,72, Camellia dengan nilai 153,33, SEED dengan nilai 72,88, CAST-128 dengan nilai 27,46, MISTY1 dengan nilai 86,37, dan TDEA dengan nilai 145,10.

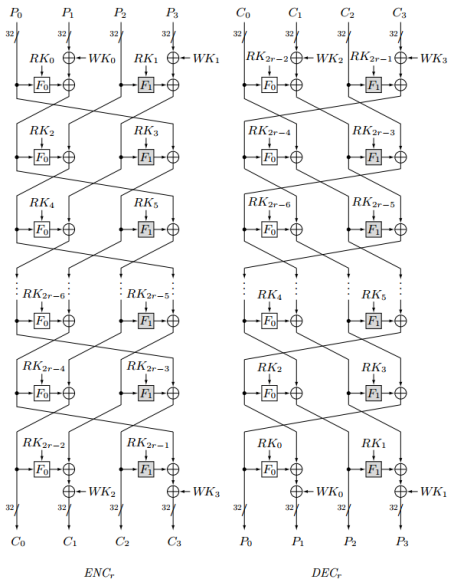
Penelitian ini akan menggunakan modul LoRa yang digunakan sebagai alat komunikasi antara *node* dan *gateway*, dalam hal ini adalah Arduino Uno. Pada penelitian ini akan digunakan sensor GPS yang akan dipasang pada *node* untuk memperoleh data. Perlindungan informasi pada GPS bisa sangat penting, terutama jika digunakan oleh militer. Jika seseorang yang tidak bertanggung jawab meretas informasi yang ada pada GPS, hal ini dapat menyebabkan masalah karena informasi yang ada pada GPS bisa sangat sensitif. Selain itu, mayoritas GPS berkomunikasi dengan stasiun yang tetap, yang menerima dan memanfaatkan data secara *real time* melalui transmisi nirkabel. Oleh karena itu, pengamanan data harus dilakukan untuk mencegah serangan-serangan yang tidak diinginkan (Azzaz & Krimil, 2018). Nantinya, data tersebut akan dienkripsi menggunakan algoritme Clefia 128-bit.

Berdasarkan hal di atas, penelitian ini akan difokuskan pada penerapan algoritme enkripsi pada komunikasi antar modul LoRa. Algoritme enkripsi yang akan diterapkan adalah algoritme Clefia dengan varian 128-bit.

2. DASAR TEORI

2.1. CLEFIA 128-BIT

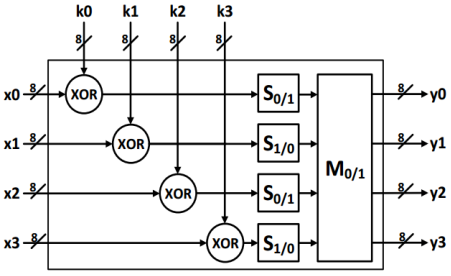
Algoritme Clefia merupakan sebuah *blockcipher* 128-bit yang menggunakan kunci simetris dengan panjang 128, 192, dan 256 bit, yang juga kompatibel dengan algoritme AES. Clefia sendiri terdiri dari dua bagian, yaitu *data processing* di mana proses enkripsi/dekripsi dilakukan, dan *key scheduling* di mana kunci yang diperlukan untuk setiap *round* enkripsi/dekripsi akan dihitung. Clefia menggunakan strukur Feistel yang menggunakan empat jalur data, dan lebar dari setiap jalur datanya adalah 32 bit. Selain itu Clefia memiliki jumlah *round* yang bervariasi tergantung panjang kuncinya. 18 *round* untuk kunci 128-bit, 22 round untuk 192 bit, dan 26 *round* untuk 256 bit (Sony Corporation, 2007). Gambar 1 di bawah ini menguraikan bagaimana proses enkripsi dan dekripsi dilakukan oleh algoritme Clefia 128-bit.



Gambar 1. Proses enkripsi dan dekripsi Algoritme CLEFIA 128-bit

Data Processing

Bagian data processing menggunakan *Generalized Feistel Network* (GFN) empat cabang, yang membagi blok input 128-bit menjadi empat kata dengan masing-masing 32-bit. Kata yang telah ditentukan (ke-1 atau ke-3) bersama dengan sub-kunci *round* akan dimasukkan ke fungsi F0 atau F1. Pada fungsi ini, kedua input akan di-XOR dan hasilnya akan dikombinasikan dengan *S-box* yang telah ada (S0 atau S1) dan matriks difusi M0 atau M1. Proses tadi akan menghasilkan XOR dengan kata kedua atau keempat. Pada *round* pertama dan terakhir, akan dilakukan XOR tambahan dari hasil *whitening key* yang sesuai (Pyrgas & Kitsos, 2019). Gambar 2 menunjukkan bagaimana Fungsi F_0/F_1 dilakukan.



Gambar 2. Fungsi F_0/F_1

Key Scheduling

Key scheduling juga menggunakan struktur GFN yang sama untuk melakukan perhitungan *intermediate key*, dengan menggabungkan kunci asli dengan konstanta CON_i. Kemudian, *round key* (RK_i) akan dibuat dengan melakukan XOR antara *intermediate key* (L), *secret key* awal, dan nilai konstanta CONi. *Intermediate key* (L) akan di-*expand* oleh fungsi *DoubleSwap* (Σ), yang merupakan permutasi *bit-wise* (Pyrgas & Kitsos, 2019). Gambar 3 merupakan langkah-langkah pada proses *key scheduling*.

(Generating L from K)
Step 1. $L \leftarrow GFN_{4,12}(CON_0^{(128)}, \ldots, CON_{23}^{(128)}, K_0, \ldots, K_3)$

(Expanding K and L)
Step 2. $WK_0|WK_1|WK_2|WK_3 \leftarrow K$
Step 3. For $i = 0$ to 8 do the following:
$T \leftarrow L \oplus (CON_{24+4i}^{(128)} \mid CON_{24+4i+1}^{(128)} \mid CON_{24+4i+2}^{(128)} \mid CON_{24+4i+3}^{(128)})$
$L \leftarrow \Sigma(L)$
if i is odd: $T \leftarrow T \oplus K$
$RK_{4i}|RK_{4i+1}|RK_{4i+2}|RK_{4i+3} \leftarrow T$

Gambar 3. *Key Scheduling*

Enkripsi

Hal pertama yang dilakukan adalah melakukan proses *initial state*. *Initial state* berfungsi sebagai tahapan awal sebelum menjalankan algoritme CLEFIA 128-bit. Pada tahapan ini, dilakukan inisialisasi terhadap 4 nilai, yaitu *plaintext* yang memiliki panjang 128 bit, *key* awal yang juga memiliki panjang 128 bit, *constant value* yang berjumlah 60 buah dengan masing-masing memiliki panjang 32 bit, dan 8 bit *S-Boxes* (S_0 dan S_1) 16 x 16 dalam bentuk heksadesimal.

Selanjutnya adalah proses *key scheduling*. *Key scheduling* digunakan untuk membentuk 3 jenis *key*, yaitu *white key*, *intermediate key*, dan *round key*. *White key* merupakan bilangan yang sama dengan *key* awal. *White key* akan digunakan pada awal dan akhir dari proses enkripsi. *Intermediate key* merupakan bilangan dengan panjang 128 bit, yang nantinya akan digunakan dalam proses pembuatan *round key*. Untuk men-*generate intermediate key* menggunakan rumus permutasi GFN dengan 128 bit key yang telah dibagi jadi 4 blok (per 32 bit) sebagai input dan nilai konstanta yang telah ditentukan sebelumnya sebagai *round key*. Nantinya dibutuhkan permutasi sebanyak 12 *round* untuk men-*generate intermediate key* sepanjang 128 bit, seperti yang terlihat di Persamaan (1).

$$L \leftarrow GFN_{4,12}(CON_{(0)}^{(128)}, \dots, CON_{23}^{(128)}, K_0, \dots, K_3 \quad (1)$$

Setelah nilai *intermediate key* berhasil didapatkan, maka langkah selanjutnya adalah pembuatan *round key*. Pada algoritme CLEFIA dengan jenis 128 bit, *round key* yang dibutuhkan sebanyak 36 buah. *Round key* dibuat dengan melakukan XOR antara *intermediate key*, *key* awal, dan *constant value* indeks ke-24 sampai indeks ke-59. Nantinya, *round key* akan berisi 36 buah bilangan dengan panjang masing-masing sebesar 32 bit. Alur proses pembuatan *round key* dapat dilihat pada Persamaan (2) dan (3).

For i = 0 to 8 do the following
$T \leftarrow L \oplus (CON_{24+4i}^{(128)} | CON_{24+4i+1}^{(128)} | CON_{24+4i+2}^{(128)} | CON_{24+4i+3}^{(128)}$
$L \leftarrow \sum(L)$
if i is odd: $T \leftarrow T \oplus K$
$RK_{4i} | RK_{4i+1} | RK_{4i+2} | RK_{4i+3} \leftarrow T$ (2)

Double Swap (∑)

$X_{128} \mapsto Y_{128}$

Y = X[7-63] | X[121-127] | X[0-6] | X[64-120] (3)

Untuk *i* 0 sampai 8, 128 bit *intermediate key* yang telah dibagi menjadi 4 blok akan melakukan perhitungan XOR dengan nilai konstanta yang dimulai dari indeks ke-24. Nantinya di setiap putaran, 4 blok *intermediate key* akan melakukan XOR dengan 4 indeks dari nilai konstanta. Setelah itu langkah selanjutnya adalah melakukan fungsi *double swap* terhadap nilai *intermediate key* yang telah di-XOR dengan nilai konstanta. *Double swap* sendiri merupakan proses menukar posisi bit pada *intermediate key*. Pada perhitungan ini, jika *i* bernilai ganjil, maka hasil perhitungan *intermediate key* dan nilai konstanta akan di-XOR lagi dengan nilai *key* awal. Selanjutanya jika 8 putaran telah selesai, maka akan terbentuk *round key* sebanyak 36 buah.

Setelah proses *key scheduling* selesai, maka proses enkripsi bisa dilakukan. Sebelum melakukan proses enkripsi, *plaintext* dengan ukuran 128 bit akan dibagi menjadi 4 blok dengan masing-masing berukuran 32 bit. Langkah pertama yang harus dilakukan sebelum melakukan enkripsi adalah *initial whitening key*. *Initial whitening key* adalah proses perhitungan XOR antara 2 *white key* pertama dan *plaintext* blok ke-2 dan ke-4, seperti yang ditunjukkan pada Persamaan (4).

$$T_0 | T_1 | T_2 | T_3 \leftarrow P_0 | (P_1 \oplus WK_0) | P_2 | (P_3 \oplus WK_1) \quad (4)$$

Setelah melakukan *initial whitening key*, Proses selanjutnya yaitu melakukan pre-enkripsi dengan menggunakan struktur permutasi GFN sebanyak 18 putaran. Dengan *plaintext* hasil perhitungan *initial whitening key* dan *round key* sebagai masukan, seperti pada Persamaan (5).

$$T_0 | T_1 | T_2 | T_3 \leftarrow GFN_{4,r}(RK_0, \dots, RK_{2r-1}, T_0, T_1, T_2, T_3) \quad (5)$$

Setelah melalui 18 putaran, maka akan didapatkan *ciphertext* sementara. Selanjutnya akan dilakukan proses *final whitening key* dengan melakukan perhitungan XOR antara 2 *white key* terakhir dan *ciphertext* blok ke-2 dan ke-4. Setelah itu, hasil akhir *ciphertext* dengan panjang 128 bit akan didapatkan. Seperti yang ditunjukkan pada Persamaan (6).

$$C_0 | C_1 | C_2 | C_3 \leftarrow T_0 | (T_1 \oplus WK_2) | T_2 | (T_3 \oplus WK_3) \quad (6)$$

2.2. LoRa

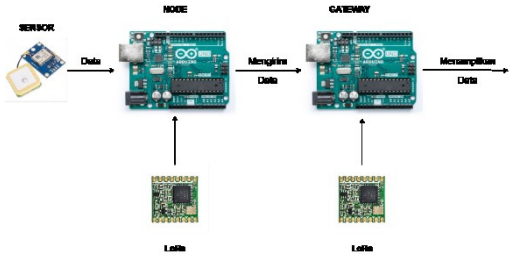
LoRa, yang berarti "*Long Range*" adalah sebuah sistem komunikasi nirkabel jarak jauh, yang dipromosikan oleh LoRa Alliance. Sistem ini memiliki tujuan agar dapat digunakan pada sebuah perangkat bertenaga baterai yang tahan lama, di mana konsumsi energi adalah yang paling utama. LoRa memiliki dua lapisan yang berbeda, yaitu yang pertama adalah lapisan fisik yang menggunakan teknik modulasi *Chirp Spread Spectrum* (CSS), yang dikembangkan oleh Semtect, memungkinkan komunikasi jarak jauh, daya rendah, dan throughput yang rendah. Lapisan ini beroperasi pada pita ISM 433-, 868-, atau 915-MHz, tergantung wilayahnya. Dan yang kedua adalah lapisan MAC (LoRaWAN). LoRaWAN menyediakan mekanisme kontrol akses menengah, yang memungkinkan banyak *end-device* untuk berkomunikasi dengan *gateway* menggunakan modulasi LoRa. Sementara modulasi LoRa adalah hak milik (*proprietary*), LoRaWAN adalah sebuah *open-standard* yang dikembangkan oleh LoRa Alliance.

3. PERANCANGAN

3.1. Perancangan Sistem

Perancangan sistem ialah sebuah tahap yang dilaksanakan untuk membuat rancangan sistem baik dari segi model dan arsitekturnya, hal apa saja yang diperlukan oleh sistem pada proses pembuatannya mencakup deskripsi sistem, dan perancangan pengujian. Langkah-langkah kerja sistem disesuaikan seperti arsitektur yang dirancang sebelumnya. Perancangan sistem akan dilakukan sebelum diterapkannya algoritme Clefia 128-bit di modul komunikasi LoRa. Perancangan sistem dibuat dengan berpatokan terhadap analisa kebutuhan yang dilaksanakan pada sub-bab sebelumnya. Di tahapan ini, akan dilakukan perancangan yang melingkupi alur sistem yang dirancang serta *library* yang akan dipakai. Penelitian ini bersifat implementatif, data yang akan dipakai berasal dari sensor yang digunakan. Terdapat 3 tahapan pada perancangan ini sebelum diterapkannya algoritme Clefia 128-bit, yaitu perancangan umum, *node* dan *gateway*. Pada perancangan sistem, membutuhkan Arduino Uno sebagai *node*, Arduino Uno sebagai *gateway*, GPS Ublox Neo 6M sebagai sensor, dan dua modul LoRa yang digunakan oleh *node* dan *gateway* sebagai modul komunikasi.

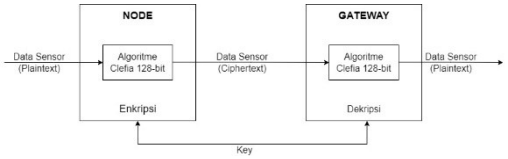
Gambar 4 menunjukkan bagaimana sistem akan dibuat.



Gambar 4. Gambaran umum sistem

3.2. Perancangan Keamanan

Perancangan keamanan merupakan sebuah tahapan yang dilakukan untuk merancang keamanan yang akan diimplementasikan ke sistem yang akan dibuat. Proses enkripsi akan ditambahkan pada *node* dengan menggunakan algoritme Clefia 128-Bit dan proses dekripsi akan ditambahkan pada sisi *gateway* sehingga pesan asli dapat dibaca. Hanya *gateway* yang sah yang dapat membaca pesan asli yang dikirimkan oleh *node*.



Gambar 5. Perancangan Keamanan Sistem

Berdasarkan Gambar 5, perancangan keamanan meliputi data dari sensor yang berupa *plaintext* akan diproses oleh *node* untuk dienkripsi menggunakan algoritme Clefia 128-bit. Setelah proses enkripsi berhasil dilakukan, data yang sudah berbentuk *ciphertext* akan dikirim ke *gateway* menggunakan modul LoRa. Setelah data berhasil diterima oleh *gateway*, data yang sudah berbentuk *ciphertext* tadi akan didekripsi menggunakan kunci (*key*) yang sudah ditentukan. Setelah proses dekripsi berhasil, data kemudian akan ditampilkan.

3.2. Perancangan Pengujian

Di tahap ini penulis melakukan beberapa pengujian pada sistem yang dibuat.

Yang pertama adalah pengujian *test vector*. Pengujian *Test Vector* dilakukan dengan cara memberikan masukan *plaintext* dan *key* yang disesuaikan dengan jurnal rujukan pada algoritme Clefia 128-bit. Dengan demikian, keluaran dari algoritme harus sama dengan keluaran yang dihasilkan oleh jurnal rujukan.

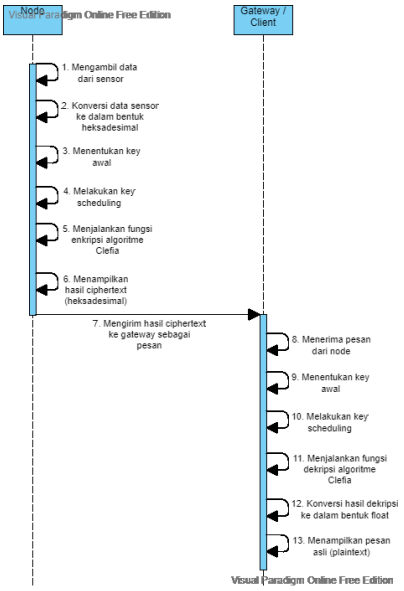
Lalu selanjutnya adalah pengujian kinerja. Pengujian ini dilaksanakan lewat cara mengukur waktu yang diperlukan sistem untuk lakukan enkripsi dan dekripsi memakai algoritme Clefia 128-bit dan algoritme AES 128 dan dibandingkan antara satu sama lain. Perhitungan waktu enkripsi dimulai saat *node* melakukan enkripsi hingga *node* menghasilkan *ciphertext*. Selain itu, pengujian dekripsi dimulai saat *gateway* melakukan dekripsi hingga *gateway* berhasil mendekripsinya.

Yang ketiga adalah pengujian serangan pasif. Pengujian ini melakukan serangan pasif dengan menggunakan metode serangan *sniffing* dengan menambahkan *client* sebagai *gateway* yang tidak sah. Modul LoRa akan lakukan *broadcast* di modul yang berfrekuensi sama.

Dan terakhir adalah pengujian serangan aktif. Pengujian ini melakukan serangan aktif dengan memakai metode serangan *known-plaintext-attack* (KPA). KPA adalah serangan pada kriptografi dengan kondisi penyerang memiliki sampel untuk *plaintext* dan memiliki hasil *ciphertext*-nya. Penyerang akan melakukan proses pencarian *key* dari sebagian *plaintext* dan *ciphertext* yang telah didapatkan.

4. HASIL DAN PEMBAHASAN

Bagian ini akan menjelaskan bagaimana hasil implementasi algoritme CLEFIA 128-bit ke dalam sistem yang telah dibuat. Sebelum melakukan enkripsi pada *node* sensor, langkah pertama adalah mendapatkan data dari sensor. Sensor yang akan digunakan adalah GPS Ublox Neo 6M. Sensor ini akan mengambil nilai *latitude* dan *longitude* yang bisa menunjukkan lokasi. Sebelum mendapatkan nilai sensor akan melakukan *looping* untuk mendapatkan nilai *latitude* dan *longitude* yang *fix*. Selanjutnya nilai *latitude* dan *longitude* tadi akan diubah ke dalam bentuk 32 bit heksadesimal untuk dijadikan sebagai *plaintext*. Setelah *plaintext* didapatkan, maka akan dilakukan *initial state* untuk menentukan *key* awal. Selanjutnya setelah semua hal yang dibutuhkan telah didapatkan, maka akan dilakukan proses enkripsi pada nilai *plaintext*. Setelah *ciphertext* didapatkan, selanjutnya akan dimasukkan ke dalam *struct* untuk dikirim ke *gateway* dengan menggunakan modul komunikasi LoRa. Data akan dikirim dalam bentuk *bytearray*.



Gambar 6. *Sequence Diagram* Proses Algoritme Clefia 128-bit Pada Sistem

```
D:\know-plain-text-attack-master>python find_key.py -i 80B66D8CF272A6291A0360EDE6530973
-o 3ED8865942CAEB0A 128
[+] I find the key --->  0x552100601581370296559154295204 62
```

Gambar 7. Hasil pengujian serangan aktif

Pada *node gateway* akan dilakukan *initial state* juga untuk menentukan *key awal* yang akan digunakan pada proses dekripsi. Setelah pesan diterima dari *node* sensor, maka akan dilakukan proses dekripsi dengan menggunakan kunci yang sama dengan proses enkripsi sebelumnya. Setelah proses dekripsi selesai, maka pesan yang masih berbentuk bilangan heksadesimal akan dikonversikan ke dalam bentuk *float* agar nilai *latitude* dan *longitude* dapat dibaca. Gambar 6 menunjukkan *sequence diagram* hasil dari implementasi algoritme Clefia 128-bit ke dalam sistem dan proses lengkap bagaimana sistem berjalan mulai dari pengambilan data sensor oleh *node* hingga data diterima oleh *gateway*.

Pengujian *Test Vector*

Pengujian *test vector* dimulai dari memasukkan *key* awal dan *plaintext* yang merujuk dari referensi rujukan yang digunakan. Setelah itu, proses *key scheduling* dijalankan untuk men-generate *white key*, *intermediate key*, dan 36 *round key*. Hasil *key* dari proses *key scheduling* sebelumnya lalu dipakai guna menjalankan proses enkripsi dan dekripsi sebanyak 18 *round*.

Dari hasil *test vector* yang didapatkan, algoritme Clefia 128-bit yang dibuat oleh penulis menghasilkan nilai yang valid, dapat dilihat pada tabel 1.

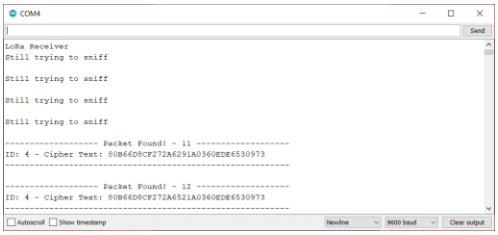
Tabel 1. Pengujian *Test Vector*

Tipe	Hasil *Test Vector* pada Sistem	Hasil *Test Vector* pada *Paper*
Plaintext	0001020304050607 08090a0b0c0d0e0f	0001020304050607 08090a0b0c0d0e0f
Key	ffeeddccbbaa99887 766554433221100	ffeeddccbbaa99887 766554433221100
Ciphertext	de2bf2fd9b74aacdf 1298555459494fd	de2bf2fd9b74aacdf 1298555459494fd

Pengujian Serangan Pasif

Pengujian serangan pasif menggunakan metode *sniffing* dengan menambahkan skenario bagaimana jika ada *gateway* palsu yang memiliki frekuensi modul LoRa yang sama dengan *node* sensor. Karena modul LoRa menggunakan metode *broadcast* untuk mengirim pesan, maka pada *node* sensor akan ditambahkan ID dengan tujuan pesan tidak dapat diterima pada *gateway* yang memiliki frekuensi yang sama namun memiliki ID yang berbeda. *Gateway* palsu akan menerapkan metode *brute force* dengan tujuan untuk mencari ID dari *node* sensor agar dapat menerima pesan yang ditujukan kepada *gateway* asli.

Gambar 8 merupakan hasil percobaan sniffing yang dilakukan oleh gateway palsu. Seperti yang ditampilkan pada gambar, *gateway* palsu berhasil mendapatkan ID dari *node* sensor sehingga *gateway* palsu berhasil membaca pesan yang dikirimkan oleh *node* sensor. Namun di sini *gateway* palsu hanya mendapatkan *ciphertext* saja yang berupa data sebesar 128 bit, tidak dengan pesan aslinya.



Gambar 8. Hasil pengujian serangan pasif

Pengujian Serangan Aktif

Pengujian ini menggunakan *known-plaintext-attack* sebagai metode serangannya. Metode *known-plaintext-attack* (KPA) bisa dimulai ketika penyerang sudah mendapatkan *ciphertext*, menggunakan sampel *plaintext*, dan panjang *key* untuk bisa mendapatkan *key* yang dicari. Nilai *ciphertext* yang dimasukkan pada pengujian ini adalah nilai yang didapat berdasarkan pengujian *sniffing* sebelum ini. Pada pengujian ini, penyerang tidak memiliki informasi algoritme kriptografi apa yang digunakan oleh korban.

Gambar 7 merupakan hasil dari pengujian menggunakan metode KPA. Dari pengujian serangan ini, penyerang tidak dapat menemukan *key* yang asli. Sehingga penyerang gagal melakukan serangan ke sistem. Karena tidak menemukan *key* yang asli, maka penyerang tidak dapat menemukan *plaintext* secara utuh, dan penyerang tidak dapat melanjutkan serangannya untuk memanipulasi *gateway* yang dituju oleh *node* sensor.

5. KESIMPULAN

Implementasi algoritme CLEFIA 128-bit pada modul komunikasi LoRa berhasil dilakukan, sesuai dengan *paper* yang digunakan sebagai acuan. Pada implementasi algoritme Clefia 128-bit pada modul LoRa didapat beberapa kesimpulan:

Penerapan algoritme CLEFIA 128-bit berhasil memberikan keamanan pada komunikasi antar modul LoRa. Hal ini dapat dibuktikan dari dua pengujian yang telah dilakukan sebelumnya, yaitu pengujian serangan pasif (*sniffing*) dan serangan aktif (*known-plaintext-attack*). Pada pengujian serangan pasif, penyerang berhasil mendapatkan *ciphertext* hasil penerapan algoritme CLEFIA 128-bit namun gagal

untuk mendapatkan pesan asli. Pada pengujian serangan aktif, penyerang gagal untuk mendapatkan *key* sehingga penyerang gagal untuk mendapatkan pesan asli.

DAFTAR PUSTAKA

ARAS, E., RAMACHANDRAN, G. S., LAWRENCE, P. & HUGHES, D., 2017. *Exploring the Security Vulnerabilities of LoRa.* Exeter, IEEE, pp. 1-6.

AZZAZ, M. S. & KRIMIL, M. A., 2018. *A New Chaos-based Text Encryption to Secure GPS Data.* El Oued, IEEE.

BOGDANOV, A., KHOVRATOVIC, D. & RECHBERGER, C., 2011. *Biclique Cryptanalysis of the Full AES.* Berlin, Springer, Berlin, Heidelberg, pp. 344-371.

ERIDANI, D., WIDIANTO, E. D., AUGUSTINUS, R. D. O. & FAIZAL, A. A., 2019. *Monitoring System in Lora Network Architecture using Smart Gateway in Simple LoRa Protocol.* Yogyakarta, IEEE, pp. 200-204.

IQBAL, A. & IQBAL, T., 2018. *Low-cost and Secure Communication System for Remote Micro-grids using AES Cryptography on ESP32 with LoRa Module.* Toronto, IEEE, pp. 1-5.

KHUTSOANE, O., ISONG, B. & ABU-MAHFOUZ, A. M., 2017. *IoT devices and applications based on LoRa/LoRaWAN.* Beijing, IEEE, pp. 6107-6112.

PERMATA, A. D. & A'MAS, 2016. *IMPLEMENTASI ALGORITMA BLOCK CIPHER CLEFIA 128 BIT UNTUK PENGAMANAN APLIKASI KOMUNIKASI CHAT.* Malang, Politeknik Negeri Malang, pp. 179-183.

PYRGAS, L. & KITSOS, P., 2019. *A Very Compact Architecture of CLEFIA Block Cipher for Secure IoT Systems.* Kallithea, IEEE, pp. 624-627.

RINI, D. A., KUSYANTI, A. & AMRON, K., 2022. Implementasi Serangan Aktif Pada Algoritme Speck 128/128bit Berbasis Modul Komunikasi LoRa. *Jurnal IKRAITH-INFORMATIKA,* 6(2), pp. 91-97.

SONY CORPORATION, 2007. *The 128-bit Blockcipher CLEFIA, Algorithm Specification.* Japan: Sony Corporation.

SUGAWARA, T., HOMMA, N., AOKI, T. & SATOH, A., 2008. *High Performance ASIC implementations of the 128-bit block cipher.* Seattle, IEEE, pp. 2925-2928.

SUSANTO, A. R., BHAWIYUGA, A. & AMRON, K., 2019. Implementasi Sistem Gateway Discovery pada Wireless Sensor Network (WSN) Berbasis Modul Komunikasi LoRa. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer,* 3(2), pp. 2138-2145.

WIXTED, A. J.,dkk., 2016. *Evaluation of LoRa and LoRaWAN for wireless sensor networks.* Orlando, IEEE, pp. 1-3.

YOUSUF, A. M., ROCHESTER, E. M. & GHADERI, M., 2018. *A low-cost LoRaWAN testbed for IoT: Implementation and measurements.* Singapore, IEEE.

Halaman ini sengaja dikosongkan