Implementasi dan Analisis *New Lightweight Cryptographic Algorithm* (NLCA) pada Perangkat Berdaya Komputasi Rendah

1st Khairunnisa Novitania Rahardja
Fakultas Informatika
Universitas Telkom
Bandung, Indonesia
khairahardja@student.telkomuniversity.ac.id

2nd Farah Afianti
Fakultas Informatika
Universitas Telkom
Bandung, Indonesia
farahafi@telkomuniversity.ac.id

***Abstrak* — Perkembangan teknologi memungkinkan *smart devices* untuk saling terhubung melalui *Internet of Things* (IoT), membuka berbagai peluang aplikasi dalam kehidupan sehari-hari. Namun, hal ini juga menimbulkan kekhawatiran terhadap keamanan data dan privasi, terutama karena perangkat IoT memiliki keterbatasan daya komputasi, memori, dan daya baterai. Untuk mengatasi ini, kriptografi ringan seperti *New Lightweight Cryptographic Algorithm* (NLCA) menjadi penting. NLCA menjanjikan kinerja lebih cepat dibandingkan dengan algoritma kriptografi ringan lainnya. Namun, belum diuji pada perangkat berdaya komputasi rendah. Dalam penelitian ini, data dengan panjang 32, 200, 400, dan 800 karakter diuji menggunakan simulator perangkat ESP32. Pada data sepanjang 200 karakter, NLCA memerlukan waktu 66,645 detik, sedangkan AES hanya membutuhkan 1,682 detik. Hasil penelitian menunjukkan bahwa NLCA memiliki rata-rata waktu eksekusi yang lebih lama dibandingkan dengan AES, dengan fungsi F menjadi faktor utama yang memperpanjang waktu eksekusi.**

Kata kunci*— Kriptografi ringan, *Internet of Things

I. PENDAHULUAN

Perkembangan teknologi informasi berkembang pesat dalam beberapa tahun ini. Saat ini, karena kemajuan teknologi yang berkembang pesat, smart devices dapat saling terhubung, berkomunikasi, dan berinteraksi melalui internet. Teknologi tersebut membuat kegiatan manusia menjadi lebih mudah dan efisien. *Internet of Things* (IoT) adalah teknologi inovatif yang berkembang pesat dengan berbagai aplikasi, fungsi, dan layanan dalam kehidupan sehari-hari dan dalam berbagai domain [1]. IoT merupakan jaringan antar perangkat fisik, kendaraan, bangunan, dan benda-benda lain yang disematkan dengan elektronik, perangkat lunak, sensor, aktuator, dan jaringan konektivitas yang memungkinkan objek-objek ini mengumpulkan dan bertukar data [2].

Semakin banyak perangkat IoT yang berkembang memunculkan banyak celah dalam keamanan data dan privasi. Masalahnya berkaitan dengan bagaimana data rentan terhadap pelanggaran dan pengawasan [3]. Perangkat IoT terhubung ke internet sehingga rentan terhadap serangan siber yang memengaruhi sistem komputer lainnya [3]. Maka dari itu, banyak perhatian yang perlu diberikan pada perangkat-perangkat ini, terutama dalam hal pemrosesan data [4]. Keamanan perangkat IoT sangat penting untuk mencegah pencurian informasi ketika perangkat IoT diretas. Area utama dalam bidang keamanan IoT meliputi *authentication, non-repudiation, access control, confidentiality, integrity*, dan *availability* [5]. Dengan bantuan kriptografi tradisional, semua tujuan ini dapat dipenuhi, namun metode ini membutuhkan alokasi sumber daya yang besar. Di sisi lain, perangkat IoT memiliki daya komputasi yang terbatas, memori yang terbatas, dan daya tahan baterai yang terbatas [4, 5]. Maka dari itu, dibutuhkan sebuah algoritma kriptografi yang ringan untuk dijalankan pada perangkat berdaya komputasi rendah.

Hingga saat ini, sudah banyak yang mengembangkan algoritma kriptografi ringan atau *Lightweight Cryptography* (LWC) yang menggunakan metode enkripsi simetri, seperti *Advanced Encryption Standard* (AES), *Data Encryption Standard* (DES), LED, dan Blowfish. Pada tahun 2021, Thabit dkk. [6] mengembangkan sebuah algoritma baru bernama *New Lightweight Cryptographic Algorithm* (NLCA). NLCA adalah *cipher* blok kunci simetris dan idenya terinspirasi oleh kombinasi metode arsitektur Feistel dan *Substitution Permutation* (SP) untuk meningkatkan kompleksitas enkripsi [6]. Pada ukuran file 50MB, NLCA menghasilkan waktu yang lebih cepat daripada algoritma DES, 3DES, AES, Blowfish, dan LED [6]. Namun, NLCA baru hanya diuji pada cloud network dan belum pernah diuji pada perangkat berdaya komputasi rendah seperti IoT. Oleh karena itu, diperlukan sebuah analisis terhadap performa NLCA pada perangkat berdaya komputasi rendah seperti IoT.

Penelitian ini bertujuan untuk mengetahui waktu eksekusi NLCA pada sebuah perangkat berdaya komputasi rendah serta membandingkan algoritma NLCA dengan algoritma enkripsi blok cipher lainnya.

Penyusunan penelitian dimulai dengan mengeksplorasi studi sebelumnya tentang kriptografi ringan dan IoT. Rencana metodologi akan diuraikan dalam bagian ketiga, disusul oleh presentasi hasil dan diskusi dalam bagian keempat, dan akhirnya kesimpulan disajikan dalam bagian kelima.

II. KAJIAN TEORI

Usman, et. al [7] (2017) mengusulkan algoritma enkripsi ringan bernama *Secure IoT* (SIT), berupa *cipher* blok 64-bit dengan kunci 64-bit. Arsitektur algoritma ini merupakan kombinasi dari feistel dan jaringan substitusi-permutasi, diimplementasikan pada mikrokontroler 8 bit. Hasilnya menunjukkan waktu eksekusi enkripsi dan dekripsi sekitar 0,188 dan 0,187 milidetik. Selain itu, ukuran kode algoritma SIT lebih kecil dibandingkan PRESENT dan AES. Implementasi ini menunjukkan hasil yang menjanjikan, menjadikan SIT sebagai kandidat yang layak untuk aplikasi IoT. Algoritma NLCA merujuk pada algoritma SIT, sehingga dapat dijadikan referensi untuk memperdalam algoritma NLCA.

Buhari, et. al [8] (2019) mengevaluasi kinerja AES dan Blowfish pada empat jenis data: gambar, audio, video, dan teks. Evaluasi dilakukan dengan memperhatikan waktu enkripsi dan *throughput*. Mereka mengembangkan prototipe menggunakan JAVA dan Netbeans IDE7.1.2. Hasil penelitian menunjukkan bahwa Blowfish lebih efisien daripada AES. Namun, waktu enkripsi untuk Blowfish terkadang berkurang dengan peningkatan ukuran data, dan efisiensinya meningkat dengan kompleksitas tipe data. Penelitian ini [8] menjadi gambaran untuk membandingkan dua algoritma kriptografi ringan.

Thabit, et. al [9] (2021) melakukan analisis keamanan dan kinerja NLCA dalam meningkatkan keamanan data di *cloud computing*. Mereka mengimplementasikan NLCA (128 dan 256 bit) menggunakan MATLAB dan menguji dengan data gambar dan teks. Penelitian ini membandingkan NLCA dengan beberapa algoritma kriptografi simetris lainnya seperti AES, DES, 3DES, dan lainnya. Hasilnya menunjukkan bahwa NLCA lebih andal dalam keamanan, fleksibilitas, penggunaan, dan kinerja memori dibandingkan algoritma lain. Penelitian ini [10] memberikan gambaran lebih terkait performansi NLCA.

El-hajj, et. al [10] (2023) melakukan penelitian untuk menganalisis berbagai algoritma *cipher* simetris pada perangkat dengan sumber daya terbatas. Mereka mengevaluasi 122 *cipher* dan membandingkannya menggunakan platform Arduino dan Raspberry Pi. Hasil penelitian menunjukkan bahwa LEA-128-128, COMET-64_CHAM-64-128, Hight-64-128, Speck-48-72, Speck-64-128, dan XTEA-64-128 adalah yang paling menjanjikan dalam hal daya, kecepatan, dan pengukuran ROM. Selain itu, Abd Yousif, et. al [11] (2023) mengusulkan pendekatan kriptografi ringan dengan menggabungkan algoritma Hummingbird 2 dan SPECK menjadi H2SPECK untuk melindungi data sensor IoT selama transisi di jaringan. Protokol MQTT diperkuat untuk keamanan yang lebih baik. Sensor ditempatkan di berbagai lokasi dalam ruangan dengan penggunaan mikrokontroler seperti Arduino dan Raspberry Pi untuk transmisi data. Pengujian standar NIST menunjukkan bahwa H2SPECK lebih cepat dalam enkripsi dan dekripsi dibandingkan dengan GOST dan SPECK, dengan durasi enkripsi yang lebih singkat tetapi tingkat keamanan yang lebih tinggi. Kedua penelitian tersebut [10, 11] memberikan gambaran terkait pengukuran performansi beberapa algoritma kriptografi ringan.

A. *Lightweight Cryptography*

Lightweight cryptography (LWC) atau kriptografi ringan merupakan teknik kriptografi yang dirancang untuk perangkat dengan sumber daya terbatas, seperti RFID, sensor, dan embedded systems [12]. Pada sistem dengan sumber daya terbatas, kriptografi ringan lebih efektif dibandingkan kriptografi konvensional karena memerlukan lebih sedikit ruang, sumber daya komputasi, daya, dan hanya menggunakan satu blok atau aliran untuk enkripsi end-to-end [12]. Terdapat empat tipe kriptografi ringan primitif yang dapat digunakan, yaitu *Lightweight Block Cipher* (LWBC), *Lightweight Stream Cipher* (LWSC), *Lightweight Hash Functions* (LWHF), dan *Elliptic Cruve Cryptography* (ECC) [5]. Faktor-faktor yang dapat dianalisis dalam kriptografi ringan meliputi ukuran kunci, ukuran blok, jumlah putaran, dan struktur [5].

B. *New Lightweight Cryptographic Algorithm* (NLCA)

New Lightweight Cryptographic Algorithm (NLCA) adalah algoritma kriptografi ringan yang diusulkan oleh Thabit, et. Al [6] untuk meningkatkan keamanan cloud. Algoritmanya sederhana dan dengan enkripsi/dekripsi yang sangat aman serta memberikan struktur yang mudah dan efektif untuk *cloud* [6]. Algoritma yang diperkenalkan adalah cipher blok kunci simetris dan idenya terinspirasi oleh kombinasi metode arsitektur Feistel dan SP untuk meningkatkan kompleksitas enkripsi [6]. Algoritma ini terdiri dari tiga tahap, yaitu *key generation*, enkripsi, dan dekripsi [6].

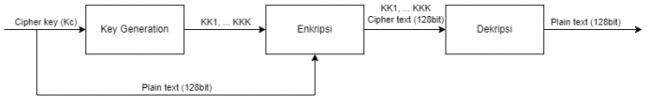
C. ESP32

Development board ini disediakan oleh produsennya sebagai perangkat berbiaya rendah dan termasuk dalam keluarga ESP32-XX, yang memiliki tiga model dengan perbedaan utama pada inti prosesor Xtensa® 32-bit yang digunakan [13]. Diluncurkan pada tahun 2016 sebagai penerus ESP8266, ESP32 menawarkan berbagai opsi konektivitas dan komunikasi, menjadikannya pilihan yang baik untuk prototipe IoT [13]. Dalam hal keamanan, prosesor ini mendukung protokol Wi-Fi IEEE 802.11 b/g/n dan dilengkapi dengan akselerator kriptografi perangkat keras, yang memungkinkan pemrosesan fungsi kriptografi secara independen dari prosesor utama [13].

III. METODE

A. Algoritma NLCA

Algoritma NLCA memiliki 3 blok proses yaitu Key Generation, Enkripsi dan Dekripsi (Gambar 1). Ide utama dari NLCA adalah menggunakan blok 16 byte (128-bit) cipher dan menginginkan 16 byte (128-bit) kunci untuk mengenkripsi data. Proses enkripsi membutuhkan putaran enkripsi dalam algoritma kunci simetris; setiap putaran selalu bergantung pada fungsi matematika untuk menghasilkan difusion dan confusion.



GAMBAR 1
Blok diagram proses keseluruhan

Dalam subbagian berikut, blok-blok ini akan diklarifikasi lebih lanjut secara detail dan terdapat beberapa catatan penting yang diikuti dalam interpretasi. Beberapa notasi yang digunakan disajikan pada Tabel 1.

TABEL 1
Tabel notasi

Notasi	Fungsi
$\oplus$	XOR
$\odot$	XNOR
$\|\|$	Concatenation

1. *Key generation*

Algoritma enkripsi berbasis Feistel bergantung pada berbagai putaran dan membutuhkan kunci yang berbeda untuk setiap putaran. Kunci diambil dari input pengguna sebanyak 16 karakter atau 128 bit. Algoritma NLCA memiliki lima putaran, oleh karena itu kita membutuhkan lima kunci unik. Gambar 3 menjelaskan proses key generation secara keseluruhan. Berikut adalah proses pembuatan kuncinya secara bertahap :

1. 128-bit *cipher key* (Kc) dibagi menjadi dua bagian, 64 bit kanan dan 64 bit kiri.
2. Kemudian, 64 bit kanan dan 64 bit kiri itu dibagi menjadi segmen 4 bit (total 16 segmen).
3. Lakukan shift-row untuk setiap 4 bit (diasumsikan shiftrow 1 bit ke kiri).
4. Karena F-Function membutuhkan 16 bit, 16 bit tersebut didapatkan dari substitusi setiap 4 bit menggunakan rumus berikut :

$$Kb_i f = \|_{j=1}^{4} Kc_{4(j-1)+i} \quad (1)$$

Dengan i = 1 hingga 4, Ka_i f merupakan :

$$Ka_i f = f(b_i f) \quad (2)$$

5. Setelah itu, 16 bit tersebut dimasukkan ke F-Function

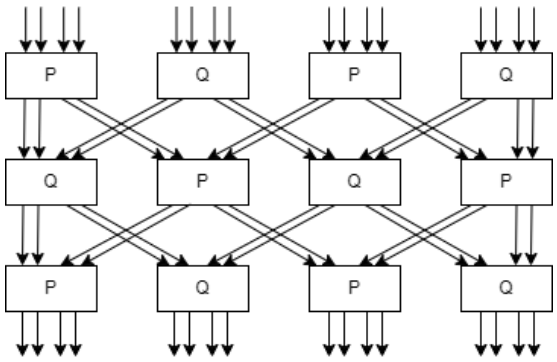
F-Function terdiri dari tabel P dan Q. Tabel-tabel ini melakukan transformasi linier dan non-linier. Gambar 2 menjelaskan proses tranformasi F-Function.

TABEL 2
Tabel P

Kci	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
F	3	F	E	0	5	4	B	C	D	A	9	6	7	8	2	1

TABEL 3
Tabel Q

Kci	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Q (Kci)	9	E	5	6	A	2	3	C	F	0	4	D	7	B	1	8



GAMBAR 2
F Function algoritma SIT

6. Hasil dari F-Function berupa matriks 4x4 dan lakukan transposisi dan *rail fence* dengan kunci 3 pada matriks tersebut.
7. Untuk mendapatkan round keys (K1, K2, …, K8), matriks diubah menjadi array 16 bit.
8. Untuk mendapatkan 4 public key, lakukan kombinasi antara 2 round key tersebut (KK1 = K1 + K2, KK2 = K3 + K4, KK3 = K5 + K6, KK4 = K7 + K8).
9. Untuk mendapatkan kunci ke-5, lakukan XOR dari keempat kunci tersebut.

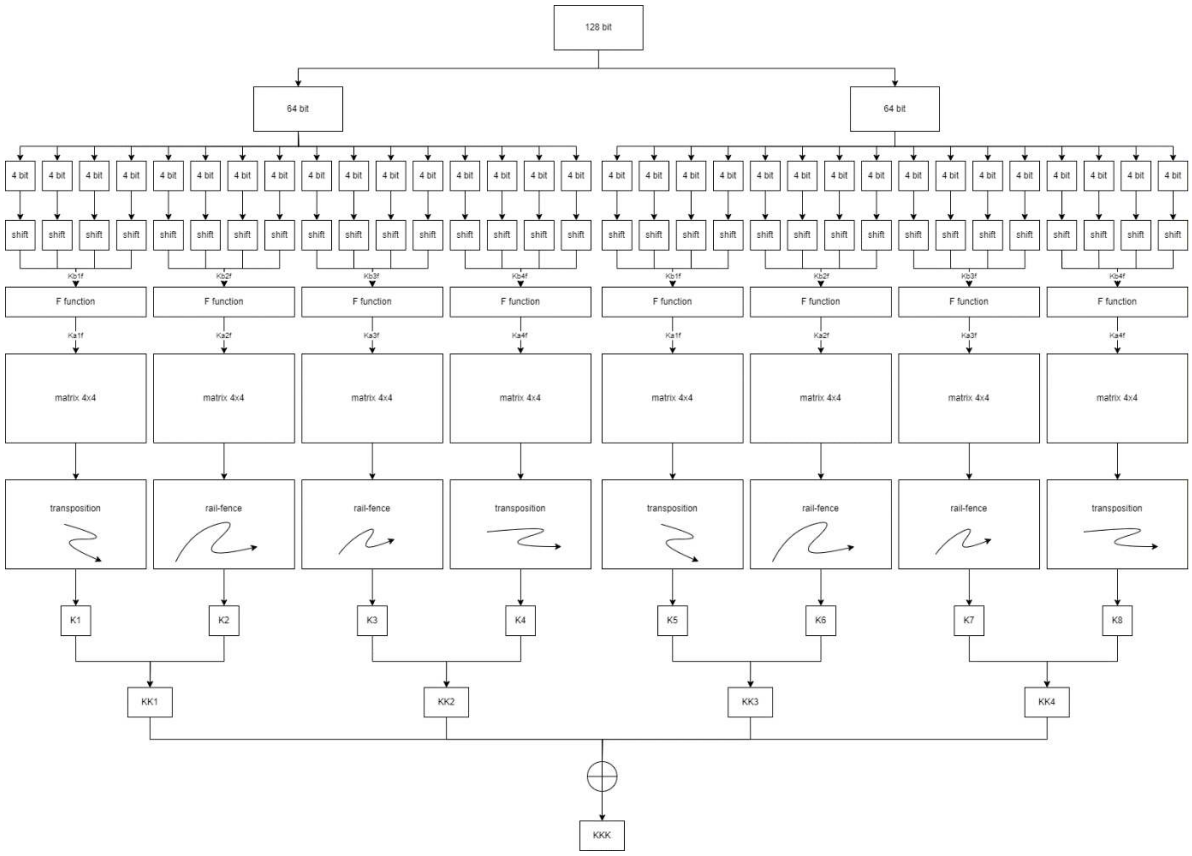
2. Enkripsi

Proses enkripsi dilakukan setelah mendapatkan lima kunci (KK1, KK2, … KKK) dari proses key generation. Pesan yang akan dienkripsi (*plain text*) dibagi menjadi blok 128-bit. Diagram proses enkripsi secara menyeluruh dapat dilihat pada Gambar 5, berikut merupakan detail tahapan enkripsi untuk satu putaran :

1. 128-bit dibagi menjadi 4 bagian yang terdiri dari 32 bit, dinamakan P1, P2, P3, dan P4.
2. Lakukan XNOR antara P1 dengan KK1 yang menghasilkan Ro11, hasilnya akan menjadi P1.
3. Ro11 dimasukkan pada F-Function (2) untuk menghasilkan EFL1.
4. Lakukan XNOR antara P4 dengan KK1 yang menghasilkan Ro14, hasilnya akan menjadi P4.
5. Ro14 dimasukkan pada F-Function (2) menghasilkan EFR1.
6. Lakukan XOR antara P3 dengan EFL1 untuk menghasilkan Ro12, hasilnya akan menjadi P2.
7. Lakukan XOR antara P2 dengan EFR1 untuk menghasilkan Ro13, hasilnya akan menjadi P3.
8. Lakukan switch antara P1 dengan P2 dan P3 dengan P4 untuk putaran berikutnya.
9. Proses ini dilakukan sebanyak 5 putaran yang menghasilkan 128 *cipher text*.

Kemudian, *cipher text* didapatkan dengan menggabungkan hasil P1 (Ro51) hingga P4 (Ro54) setelah lima putaran atau menggunakan rumus **Error! Reference source not found.**

$$Ct = Ro51 \,\|\, Ro52 \,\|\, Ro53 \,\|\, Ro54 \quad (3)$$



GAMBAR 3
Proses *key generation*

3. Dekripsi

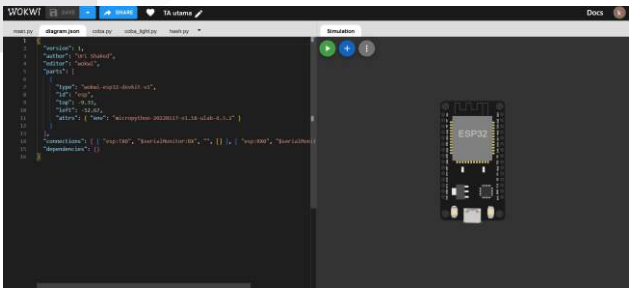
Sedangkan pada proses dekripsi, kunci terenkripsi akan diekstrak dari kunci yang dikirim oleh pengguna. Prosedurnya sama seperti proses enkripsi. Blok yang panjangnya 128-bit pertama-tama dipecah menjadi 4 sub-blok, dan kemudian ditangani dengan kunci yang sama menggunakan operasi campuran XOR dan Sub. Perbedaannya adalah bahwa kunci yang digunakan dalam enkripsi digunakan dalam urutan terbalik.

B. Simulasi Mikrokontroler

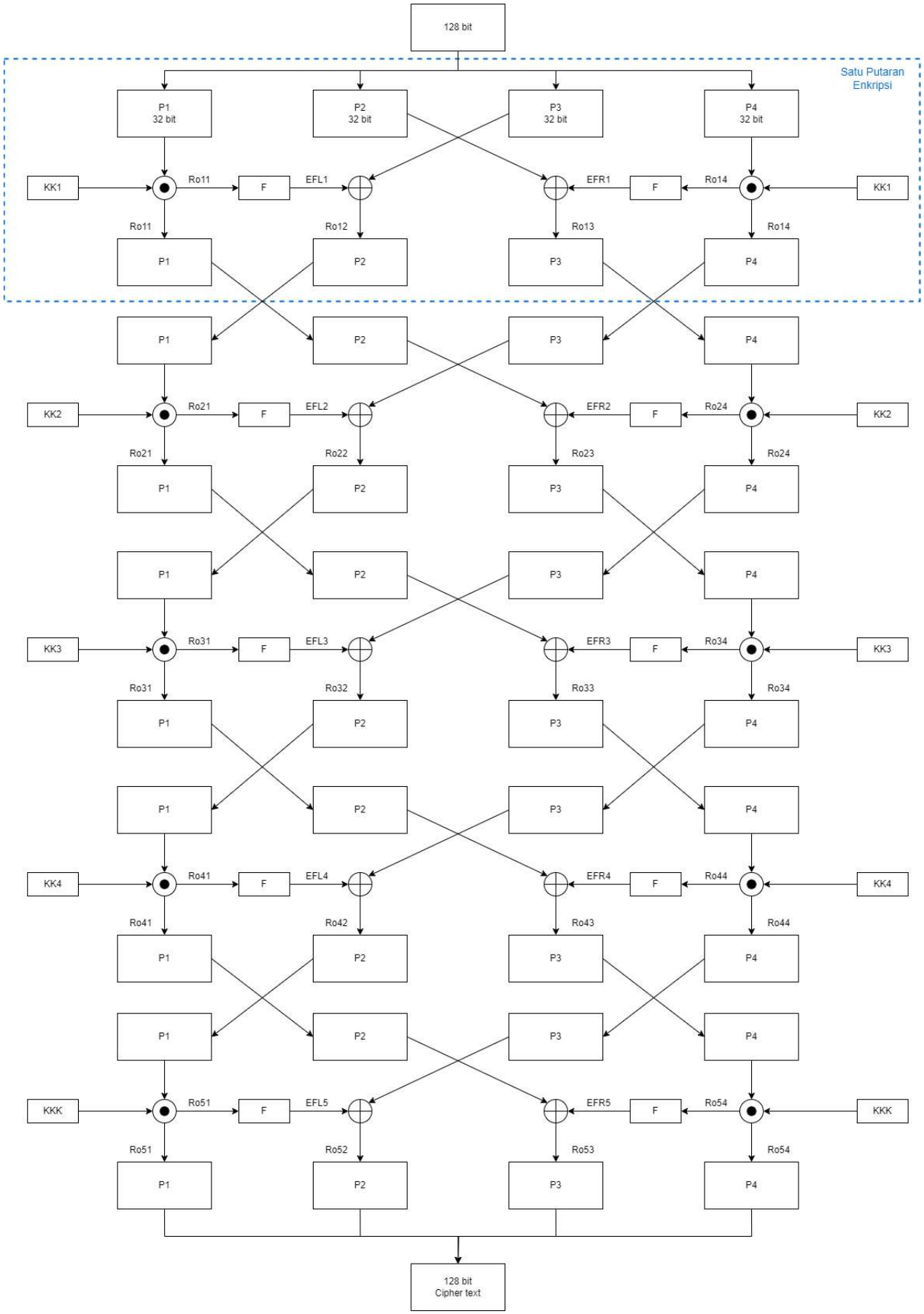
Penelitian ini memanfaatkan simulasi mikrokontroler Wokwi (wokwi.com) [14]. Wokwi memungkinkan pemrograman dan simulasi untuk Arduino, ESP32, Raspberry Pi, serta berbagai papan dan sensor populer lainnya. Papan yang digunakan merupakan papan ESP32. Simulasi ini mendukung berbagai fitur seperti inti prosesor, GPIO, PSRAM, WiFi, timer, ADC, RNG, debugging GDB, serta akselerator AES, SHA, dan RSA [15]. Konfigurasi yang digunakan dalam penelitian ini adalah wokwi-esp32-devkit-v1 dengan MicroPython versi 1.18. Gambar 4 menunjukan tampilan simulasi pada wokwi menggunakan papan ESP32.

C. Dataset

Dataset yang digunakan dalam penelitian ini meliputi kunci dan *plain text* yang dihasilkan secara acak. . Pemilihan data dalam penelitian ini memungkinkan pengujian dengan berbagai panjang data. Penelitian ini menggunakan data sepanjang 16 karakter untuk menguji data tepat satu blok (128 bit). Selain itu, data dengan panjang 200, 400, 600, dan 800 karakter dipilih untuk mengevaluasi kinerja algoritma terhadap berbagai ukuran data. Data dengan panjang tersebut juga dipilih karena menghasilkan blok parsial, yakni data yang tidak terdiri dari jumlah blok yang bulat, sehingga memungkinkan untuk menguji algoritma dalam menangani data dengan blok yang tidak lengkap. Selain itu, penelitian ini menggunakan 30 data untuk menghitung waktu eksekusi rata-rata karena telah mempertimbangkan statistik, ketepatan, kepraktisan, dan manajemen sumber daya. Menggunakan 30 data untuk menghitung waktu eksekusi rata-rata adalah pendekatan yang menyeimbangkan keandalan statistik, ketepatan, kepraktisan, dan manajemen sumber daya. Ini cukup untuk memastikan bahwa rata-rata tidak terlalu besar sehingga praktis dan tidak memakan waktu yang tidak perlu.



GAMBAR 4
Tampilan simulasi wokwi



GAMBAR 5
Proses enkripsi

IV. HASIL DAN PEMBAHASAN

A. Hasil Pengujian

Berikut merupakan contoh implementasi algoritma ini pada block data sepanjang 16 karakter (128-bit) :

TABEL 4
Contoh input 16 karakter

Blok data	Hide data safely
Key	LLDp4uxrCCxb26gI

Data dan kunci tersebut menghasilkan 5 kunci dengan panjang 64 bit. Gambar 6 merupakan hasil tangkapan layar dari program yang dibuat. Berikut merupakan kunci ke-1 hingga 5 yang direpresentasikan dalam binary, dan data yang terenkripsi direpresentasikan dalam hex :

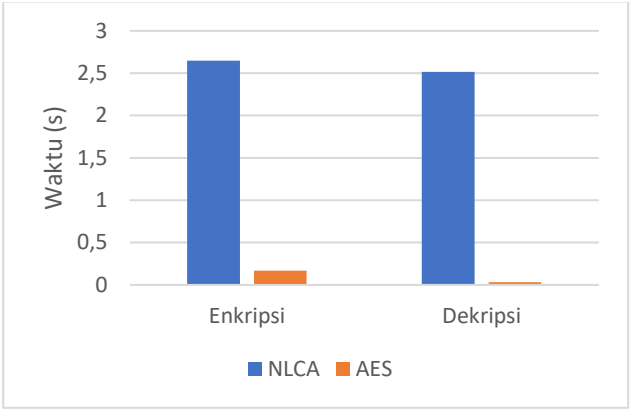
TABEL 5
Hasil pengujian data 16 karakter

Key 1	11011110100010110101100001000010
Key 2	10110111110110001111101001010000
Key 3	10011000110100110001000011001111
Key 4	10110011110110000001001011011110
Key 5	01000010010110001010000000000011
Data terenkripsi (hex)	b396b2a96520a1e9246428529d9acbf0

```
NLCA for Data Encryption

DATA  1
Secret key :  LLDp4uxrCCxb26gI
Key Generation
KK 1 11011110100010110101100001000010
KK 2 10110111110110001111101001010000
KK 3 10011000110100110001000011001111
KK 4 10110011110110000001001011011110
KK 5 01000010010110001010000000000011
Data Encryption
Plain text :
Hide data safely

Cipher text :
hex : b396b2a96520a1e9246428529d9acbf0

Data Decryption

Plain text :
Hide data safely


==================================

No, Key gen time (us), Enc time (us), Dec time (us), Total time (us)
1, 1971234.798431, 2648114.919662, 2516939.878464, 7205172.538757
```

GAMBAR 6
Hasil implementasi data 16 karakter

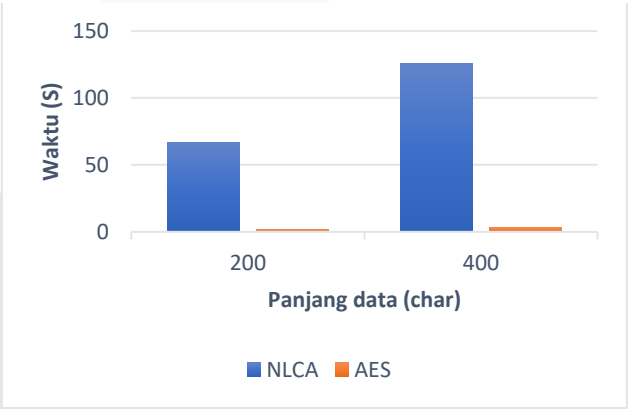
Pengujian dilakukan menggunakan data pada Tabel 4 yang dilakukan sebanyak satu kali. NLCA menghasilkan waktu enkripsi selama 2,648 detik dan dekripsi selama 2,517 detik. Sementara itu, AES menghasilkan waktu enkripsi selama 0,335 detik dan dekripsi selama 0,038 detik. Perbandingan waktu enkripsi dan dekripsi dari kedua algoritma sebagai berikut :



GAMBAR 7
Perbandingan waktu eksekusi NLCA dengan AES

Eksperimen dilakukan pada data dengan panjang 200, 400, dan 800 karakter, masing-masing sebanyak 30 kali. Kunci yang digunakan berupa string acak dengan 16 karakter (128-bit). Untuk data sepanjang 200 karakter, NLCA menghasilkan rata-rata waktu dari pembuatan kunci hingga dekripsi sebesar 66,645 detik. Untuk data 400 karakter, rata-rata waktunya adalah 125,512 detik. Namun, untuk data 800 karakter, NLCA hanya berhasil dijalankan sekali sebelum munculnya error MemoryError.

Algoritma simetris AES juga diimplementasikan dalam simulator yang sama dengan NLCA untuk data sepanjang 200, 400, dan 800 karakter. Pada data 200 karakter, AES menghasilkan rata-rata waktu 1,682 detik. Untuk data 400, rata-rata waktunya adalah 3,482 detik. Sedangkan untuk data 800 karakter menghasilkan rata-rata waktu sebesar 7,553 detik. Hasil perbandingan rata-rata waktu enkripsi antara NLCA dan AES setelah 30 kali percobaan ditunjukkan pada Gambar 8.



GAMBAR 8
Perbandingan rata-rata waktu eksekusi NCLA dengan AES

B. Analisis Hasil Pengujian

NLCA berhasil dijalankan pada simulator papan ESP32 menggunakan data sepanjang 200 dan 400 karakter sebanyak 30 kali. Pada kedua algoritma, waktu enkripsi lebih besar daripada dekripsi. Waktu yang dihasilkan oleh NLCA jauh lebih besar dibanding AES. Pada data satu block, waktu enkripsi NLCA 16 kali lebih besar dibanding AES. Sama halnya dengan proses dekripsi, waktu yang dibutuhkan NLCA mencapai 76 kali lebih besar dibanding AES yang hanya membutuhkan waktu 0,033 detik.

Rata-rata waktu yang dibutuhkan NLCA untuk mengolah data sepanjang 200 karakter 40 kali lebih besar dibanding AES. Pada data sepanjang 400 karakter, kedua algoritma membutuhkan waktu dua kali lipat dari data sepanjang 200 karakter. NLCA menghasilkan waktu 37 kali lebih besar dibanding AES. Namun, untuk data sepanjang 600 dan 800 karakter, NLCA hanya berhasil dijalankan satu kali enkripsi dan dekripsi sebelum munculnya MemoryError, kemungkinan disebabkan oleh kehabisan RAM. Hal ini mungkin terjadi karena program mengonsumsi terlalu banyak memori. Setelah ditelusuri lebih lanjut, mulai dari data 580 karakter, NLCA hanya dapat dijalankan satu kali sebelum munculnya MemoryError.

TABEL 6
Hasil skenario pengujian rata-rata waktu eksekusi

Karakter	200	400	580	600	800
Rata-rata (s)	66,645	125,512	MemoryError	MemoryError	MemoryError

NLCA menggunakan 5 putaran untuk setiap blok enkripsi. Dalam setiap putaran, algoritma ini melakukan 2 operasi XNOR, 2 operasi F-Function, dan 2 operasi XOR. Mengingat satu blok terdiri dari 128-bit, data sepanjang 200 karakter memerlukan 12,5 blok, atau 13 blok setelah *padding*. Dengan demikian, untuk data 200 karakter, NLCA menjalankan total 5 x 13 putaran. Setelah dilakukan analisis lebih lanjut, waktu yang dibutuhkan oleh masing-masing fungsi dan perhitungannya dijelaskan sebagai berikut :

TABEL 7
Analisis waktu setiap fungsi

Fungsi	Waktu (s)
XNOR	0,056
F-Function	0,164
XOR	0,007
1 putaran	**0,454**
5 putaran	**2,271**
65 putaran	**29,527**

Berdasarkan Tabel 7, analisis menunjukkan bahwa untuk satu putaran, total waktu yang dibutuhkan adalah 0,454 detik, dan untuk 5 putaran adalah 2,271 detik. Dengan 65 putaran yang diperlukan untuk data 200 karakter, total waktu yang dihasilkan adalah 29,527 detik hanya untuk ketiga fungsi tersebut. Namun, total waktu enkripsi satu kali putaran adalah 34,378 detik, yang berarti 4,851 detik digunakan untuk proses tambahan seperti pemecahan data, penetapan, dan pemindahan variabel. Analisis juga menunjukkan bahwa F-Function memakan waktu paling lama, yaitu sekitar 62% dari total waktu satu kali enkripsi, dengan waktu eksekusi tiga kali lebih lama daripada XNOR dan 22 kali lebih lama daripada XOR.

V. KESIMPULAN

Kemajuan teknologi memungkinkan smart devices terhubung melalui Internet of Things (IoT), membuka berbagai peluang aplikasi dalam kehidupan sehari-hari. Namun, hal ini juga menimbulkan kekhawatiran mengenai keamanan data dan privasi, terutama karena perangkat IoT memiliki keterbatasan daya komputasi, memori, dan daya baterai.

Dalam penelitian ini, implementasi New Lightweight Cryptographic Algorithm (NLCA) diuji pada perangkat berdaya komputasi rendah menggunakan simulasi ESP32 pada Wokwi. NLCA adalah algoritma block cipher 128-bit dengan kunci 128-bit untuk enkripsi data. Eksperimen dilakukan menggunakan data sepanjang 200, 400, dan 800 karakter. Hasilnya menunjukkan bahwa waktu eksekusi NLCA lebih besar dibandingkan dengan AES, dengan rentang 37 hingga 40 kali lebih lama. Fungsi F menjadi fungsi yang memakan waktu paling lama. Jumlah putaran yang akan dijalankan bergantung dengan panjang input, semakin panjang inputnya maka semakin banyak putaran yang dijalankan. Selain itu, keterbatasan memori juga menjadi tantangan dalam implementasi NLCA.

Selanjutnya, NLCA dapat dioptimasi untuk meningkatkan efisiensi dan efektivitas serta membandingkannya dengan lebih banyak algoritma. Pengukuran keamanan dari berbagai parameter juga perlu dilakukan untuk memastikan keamanan algoritma ini pada perangkat berdaya komputasi rendah.

REFERENSI

[1] G. Lampropoulos, K. Siakas, and T. Anastasiadis, “Internet of Things in the Context of Industry 4.0: An Overview,” *International Journal of Entrepreneurial Knowledge*, vol. 7, no. 1, pp. 4–19, Jul. 2019, doi: 10.2478/ijek-2019-0001.

[2] E. Oztemel and S. Gursev, “Literature review of Industry 4.0 and related technologies,” *Journal of Intelligent Manufacturing*, vol. 31, no. 1. Springer, pp. 127–182, Jan. 01, 2020. doi: 10.1007/s10845-018-1433-8.

[3] A. Kumar Reddy Nadikattu Sr, “IOT AND THE ISSUE OF DATA PRIVACY,” *NOVATEUR PUBLICATIONS INTERNATIONAL JOURNAL OF INNOVATIONS IN ENGINEERING RESEARCH AND TECHNOLOGY [IJIERT]*, vol. 5, no. 10, 2018.

[4] N. F. Ibrahim and J. I. Agbinya, “A Review of Lightweight Cryptographic Schemes and Fundamental Cryptographic Characteristics of Boolean Functions,” *Advances in Internet of Things*, vol. 12, no. 01, pp. 9–17, 2022, doi: 10.4236/ait.2022.121002.

[5] S. S. Dhanda, B. Singh, and P. Jindal, “Lightweight Cryptography: A Solution to Secure IoT,” *Wirel Pers Commun*, vol. 112, no. 3, pp. 1947–1980, Jun. 2020, doi: 10.1007/s11277-020-07134-3.

[6] F. Thabit, A. P. S. Alhomdy, A. H. A. Al-Ahdal, and P. D. S. Jagtap, “A new lightweight cryptographic algorithm for enhancing data security in cloud computing,” *Global Transitions Proceedings*, vol. 2, no. 1, pp. 91–99, Jun. 2021, doi: 10.1016/j.gltp.2021.01.013.

[7] M. Usman, I. Ahmed, M. I. Aslam, S. Khan, and U. A. Shah, “SIT: A Lightweight Encryption Algorithm for Secure Internet of Things,” Apr. 2017, doi: 10.14569/IJACSA.2017.080151.

[8] B. A. Buhari, A. A. Obiniyi, K. Sunday, and S. Shehu, “Performance Evaluation of Symmetric Data

Encryption Algorithms: AES and Blowfish," *Saudi Journal of Engineering and Technology*, vol. 04, no. 10, pp. 407–414, Oct. 2019, doi: 10.36348/sjeat.2019.v04i10.002.

[9] F. Thabit, S. Alhomdy, and S. Jagtap, "Security analysis and performance evaluation of a new lightweight cryptographic algorithm for cloud computing," *Global Transitions Proceedings*, vol. 2, no. 1, pp. 100–110, Jun. 2021, doi: 10.1016/j.gltp.2021.01.014.

[10] M. El-hajj, H. Mousawi, and A. Fadlallah, "Analysis of Lightweight Cryptographic Algorithms on IoT Hardware Platform †," *Future Internet*, vol. 15, no. 2, Feb. 2023, doi: 10.3390/fi15020054.

[11] I. Abd Yousif, S. Ayad Hussein, H. K. Hoomod, and Q. Humadi Mohammed, "New Hybrid Lightweight Data Encryption Algorithm for Operation System Protocol in Internet of Thing Environment," *International Journal Of Latest Technology In Engineering & Management*, vol. 8, no. 5, pp. 13–21, Aug. 2023, doi: 10.56581/ijltem.8.5.13-21.

[12] S. Aruna, G. Usha, P. Madhavan, and M. V Ranjith Kumar, "Lightweight Cryptography Algorithms for IoT Resource-Starving Devices," 2020.

[13] M. A. Caraveo-Cacep, R. Vázquez-Medina, and A. Hernández Zavala, "A survey on low-cost development boards for applying cryptography in IoT systems," *Internet of Things (Netherlands)*, vol. 22, Jul. 2023, doi: 10.1016/j.iot.2023.100743.

[14] CodeMagic LTD, "Wokwi." Accessed: May 24, 2024. [Online]. Available: wokwi.com

[15] CodeMagic LTD, "ESP32 Simulation | Wokwi Docs." Accessed: Jun. 25, 2024. [Online]. Available: https://docs.wokwi.com/guides/esp32