Evaluasi Kinerja Enkripsi Algoritma LEA Mode CTR pada NodeMCU8266

Vian Satria Maulana Navalino[1], Ahmad Farid Wadjdi[2], Yudistira Asnar[3]

[1,2] Rekayasa Pertahanan Siber, Fakultas Sains dan Teknologi Pertahanan, Universitas Pertahanan,
[3] Teknik Informatika, Institut Teknologi Bandung, Bandung, Indonesia
[1]viansatria57@gmail.com, [2]achm047@brin.go.id, [3]yudis_asnar@yahoo.com

Abstrak

Dalam era digital saat ini, Internet of Things (IoT) telah menjadi integral dalam berbagai sektor seperti *Smart Cities*, *Smart Home*, kesehatan, dan industri, memainkan peran penting dalam memudahkan kegiatan manusia. Namun, pentingnya menjaga privasi dan keamanan data menjadi tantangan utama, terutama karena perangkat IoT sering kali memiliki sumber daya terbatas. Hal ini menuntut penggunaan algoritma kriptografi yang ringan untuk memastikan keamanan tanpa membebani sumber daya terbatas tersebut. *Lightweight Encryption Algorithm* (LEA), yang diaplikasikan dalam penelitian ini, muncul sebagai solusi yang efisien untuk mengatasi masalah ini. LEA menawarkan keunggulan dalam ukuran kode yang kecil, efisiensi pemrosesan yang tinggi pada perangkat dengan memori dan daya komputasi terbatas. Dalam penelitian ini, LEA diimplementasikan pada NodeMCU8266 dan dibandingkan dengan algoritma lain dalam mikrokontroler yang berbeda. Hasil menunjukkan bahwa LEA menawarkan kinerja yang lebih cepat dalam proses enkripsi dibandingkan dengan AES-128, meskipun dengan throughput yang lebih rendah jika dibandingkan dengan implementasi AES pada mikrokontroler ARM Cortex M3. Sebagai kesimpulan, LEA terbukti efektif untuk aplikasi IoT, menyeimbangkan keamanan dan efisiensi. Untuk penelitian mendatang, disarankan untuk mengimplementasi LEA menggunakan bahasa pemrograman Arduino, yang mungkin menawarkan hasil yang lebih optimal dibandingkan MicroPython.

Kata kunci: Enkripsi, IoT, LEA

1. Pendahuluan

Dewasa ini, Internet of Things (IoT) telah muncul sebagai teknologi yang membantu manusia menjadi lebih mudah. IoT telah membantu kita dalam beberapa sektor, seperti Smart Cities (Bellini et al., 2022), Smart Home (Choi et al., 2021), Kesehatan (Selvaraj & Sundaravaradhan, 2020), Industri (Malik et al., 2021) dan lainnya.

Dalam lanskap IoT yang luas, ranah di mana perangkat yang tak terhitung jumlahnya saling terhubung dan berkomunikasi, sejumlah tantangan keamanan muncul ke permukaan. Salah satunya, kerahasiaan pada transmisi data muncul sebagai masalah yang sangat penting (Shahzad et al., 2022). Sama seperti kita menghargai privasi percakapan dan informasi pribadi kita, perangkat IoT menangani data sensitif yang jika terekspos dapat membahayakan kehidupan pribadi, keamanan finansial, dan bahkan infrastruktur penting.

Kerahasiaan data berarti melindungi data dari akses yang tidak sah atau pencurian. Hal ini dapat dicapai dengan bantuan kriptografi melalui enkripsi dan dekripsi (Al-Amri et al., 2023). Namun, domain IoT menghadapi kendala yang berakar pada sumber daya komputasi yang terbatas. Sifat algoritma enkripsi yang rumit, mungkin tidak selaras dengan keterbatasan sumber daya pada perangkat IoT. Dengan demikian, muncul tantangan untuk mengembangkan mekanisme enkripsi yang tidak hanya memastikan keamanan data tetapi juga memberikan efisiensi dalam lingkungan IoT yang memiliki sumber daya terbatas, yaitu Lightweight Cryptography.

NodeMCU8266 merupakan contoh sederhana dari perangkat dengan sumber daya terbatas, yang sering digunakan dalam aplikasi IoT. Kemampuan pemrosesan dan memori yang terbatas menimbulkan tantangan tersendiri untuk penerapan algoritma kriptografi yang perlu mengelola keamanan dan kinerja secara efisien, seperti penerapan RSA dan AES yang mahal secara komputasi yang membutuhkan memori dalam jumlah besar yang dapat mengganggu performa perangkat (Alluhaidan & Prabu, 2023).

Dalam menghadapi tuntutan keamanan dalam kerangka kerja IoT dan keharusan untuk menjaga privasi data, peran metode kriptografi menjadi sangat penting. Dasar dari perlindungan data, proses enkripsi dan dekripsi harus disesuaikan dengan karakteristik kapasitas komputasi dan penyimpanan yang terbatas pada perangkat IoT. Kondisi ini membutuhkan perubahan ke arah penggunaan strategi enkripsi yang ringan. Inti dari kriptografi ringan terletak pada kemampuannya untuk menjaga keamanan data saat beroperasi dalam batasan penggunaan sumber daya secara minimal yang merupakan aspek penting untuk pertimbangan aplikasi IoT.

Penelitian ini berfokus pada implementasi LEA cipher pada *development board* NodeMCU8266 yang ditenagai oleh mikrokontroler ESP8266 dan menggunakan MicroPython. Pemilihan LEA sebagai fokus studi ini didasarkan pada ukuran LEA yang dapat diimplementasikan dengan kode yang kecil (Hong et al., 2014), menjadikannya solusi yang ideal untuk sistem dengan keterbatasan memori dan ruang penyimpanan, seperti NodeMCU8266. Selain itu, algoritma LEA menunjukkan keunggulan dalam kecepatan pemrosesan, terutama bila dibandingkan dengan algoritma SIMON pada prosesor 32-bit dan 64-bit. LEA juga memiliki kecepatan enkripsi yang lebih tinggi daripada XTEA dan TEA, yang keduanya berbasis Feistel (Mishra et al., 2019).

2. Kajian Literatur

2.1 *Lightweight cryptography*

Lightweight cryptography mengacu pada metode atau algoritma enkripsi yang secara khusus dirancang untuk memenuhi persyaratan keamanan perangkat dengan sumber daya terbatas dalam hal daya komputasi yang rendah, daya tahan baterai yang terbatas, ukuran kecil, memori kecil, dan catu daya yang terbatas, seperti yang penelitian yang dilakukan oleh (Aikawa et al., 1998)(Singh et al., 2017). Selain itu, *Lightweight cryptography* harus menunjukkan ketahanan terhadap metode kriptanalitik yang sudah ada, termasuk kriptanalisis linier dan diferensial ketika mengacu pada *block cipher*. Ini penting untuk memahami bahwa kriptografi ringan tidak sama dengan kriptografi yang lemah atau rentan. Dengan kata lain, elemen-elemen dasar dari kriptografi ringan tidak seharusnya menjadi titik lemah yang berisiko mengganggu keamanan sistem secara keseluruhan (Dinu et al., 2019).

2.2 Block Cipher

Block Cipher adalah salah satu teknik enkripsi simetris di mana data plaintext dibagi menjadi blok-blok dengan ukuran tetap dan setiap blok dienkripsi satu per satu dengan kunci yang sama untuk menghasilkan blok-blok *ciphertext* (Aumasson, 2018). Proses dekripsi pada *block cipher* akan mengonversi kembali blok-blok *ciphertext* menjadi blok-blok *plaintext*. Terdapat beberapa jenis arsitektur pada *block cipher*, diantaranya adalah *Substitution-Permutation Network* (SPN), *Feistel Network*, dan *Addition, Rotation, and XOR* (ARX).

SPN adalah *block cipher* yang mengkombinasikan dua operasi utama, yaitu substitusi dan permutasi (Aumasson, 2018). Dalam operasi substitusi, bit atau blok data digantikan dengan yang lain. Sementara dalam operasi permutasi, urutan bit atau blok data diatur ulang. Proses ini biasanya dilakukan dalam beberapa ronde.

Pada Feistel Network, blok data dibagi menjadi dua bagian, dan hanya satu bagian yang dienkripsi pada satu waktu, sementara bagian lainnya tetap tidak berubah. Proses ini diulangi dalam beberapa ronde untuk menciptakan enkripsi yang kuat (Zhang et al., 2018).

ARX merupakan jenis block cipher yang memanfaatkan tiga operasi dasar: penjumlahan (Add), rotasi (Rotate), dan operasi XOR (Thakor et al., 2021). Keunggulan ARX terletak pada operasi-operasinya yang sederhana dan mudah diimplementasikan. Selain itu, ARX efisien pada banyak platform karena hanya mengandalkan operasi bitwise.

2.3 Gambaran Umum Algoritma LEA

Lightweight Encryption Algorithm (LEA) (Hong et al., 2014) pertama kali diperkenalkan sebagai bagian dari proyek untuk menciptakan algoritma enkripsi yang ringan. Tujuan utama dari penciptaan LEA adalah untuk mengakomodasi perangkat yang memiliki sumber daya terbatas, seperti perangkat IoT. Seiring dengan pertumbuhan eksponensial perangkat IoT dan kebutuhan untuk menjaga komunikasi mereka aman, LEA menjadi salah satu solusi yang menawarkan efisiensi dan keamanan. LEA didesain dengan struktur blok cipher berdasarkan operasi ARX. Meskipun menggunakan pendekatan sederhana, LEA tetap memberikan tingkat keamanan yang memadai. Operasi dasar ARX memungkinkan implementasi yang efisien pada berbagai platform, termasuk perangkat dengan sumber daya terbatas. Dengan ronde-ronde enkripsi tertentu, LEA mampu menghadirkan tingkat proteksi yang tinggi terhadap berbagai jenis serangan. Berkat dasar operasi ARX, LEA dapat dijalankan dengan cepat dan memerlukan sedikit sumber daya. Meskipun ringan, LEA tetap menawarkan tingkat keamanan yang kompetitif, menjadikannya aman untuk berbagai aplikasi. LEA mudah diintegrasikan ke dalam berbagai platform, terutama perangkat dengan kapasitas memori dan daya pemrosesan terbatas.

Tabel 1. Parameter LEA (Song & Seo, 2021)

Cipher	Block Size	Key Size	Round Number	Word Size
LEA-128	128	128	24	32
LEA-196	128	196	28	32
LEA-256	128	256	32	32

2.4 *Block Cipher Operation*

Beberapa mode operasional tersedia untuk mengelola enkripsi dan dekripsi data, dan pemilihan mode ini secara signifikan berdampak pada keamanan dan keefektifan prosedur kriptografi. Mode *Counter* (CTR) menjadi mode yang sangat patut diperhatikan. Dalam mode ini, cipher blok beroperasi mirip dengan cipher stream. Sebuah

penghitung, biasanya dimulai dengan nilai tertentu, mengalami enkripsi, dan output terenkripsi, yang dikenal sebagai keystream, digabungkan dengan plaintext melalui operasi XOR untuk menghasilkan ciphertext. Selanjutnya, untuk setiap blok data baru, penghitungnya bertambah, biasanya satu. Manfaat utama dari mode CTR adalah kemampuannya untuk memfasilitasi paralelisasi (Pirzada et al., 2019), yang memungkinkan enkripsi independen dari berbagai blok, yang berkontribusi pada proses enkripsi yang cepat dan efisien.

Mode CTR juga menawarkan keuntungan dalam menangani blok data yang tidak sesuai dengan ukuran blok cipher konvensional. Dalam kasus di mana data yang tersisa lebih kecil dari ukuran blok standar, hanya sebagian dari aliran kunci yang diperlukan untuk enkripsi. Atribut ini membuat mode CTR menjadi sangat serbaguna, mengakomodasi berbagai situasi dan ukuran data.

3. Metodologi

3.1 Desain Penelitian

Penelitian ini dirancang sebagai penelitian eksperimental untuk menguji implementasi dan efektivitas dari enkripsi LEA-CTR pada NodeMCU8266. Fokus utama adalah pada analisis kinerja dan keamanan dari implementasi algoritma LEA dalam mode operasi CTR pada perangkat IoT dengan sumber daya terbatas.

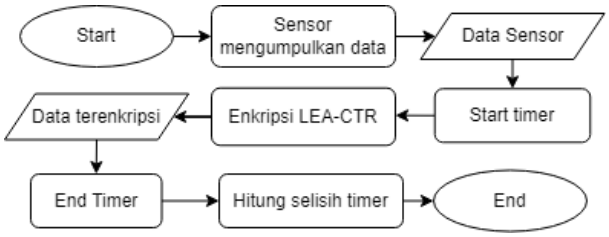
3.2 Alat dan Bahan

Beberapa komponen yang diperlukan dalam penelitian antara lain:

- NodeMCU8266: Digunakan sebagai platform utama untuk implementasi dan pengujian.
- MicroPython: Bahasa pemrograman yang digunakan untuk mengembangkan dan menjalankan kode enkripsi pada NodeMCU8266.
- Perangkat Komputer: Digunakan untuk pengembangan kode dan analisis data.
- Sensor DHT11: Digunakan sebagai sensor untuk mengukur suhu dan kelembapan.

3.3 Prosedur Penelitian

Gambar 1 menunjukkan alur proses pengujian kinerja LEA Chiper. Proses dimulai dengan perolehan data dari sensor terpasang, yang menghasilkan input berupa plaintext. Data ini mencerminkan informasi mentah dalam keadaan yang belum diproses dan belum terlindungi. Setelah data sensor berhasil dikumpulkan, sebuah variabel start_time dibuat, menandai saat tepat sebelum proses enkripsi LEA-CTR dimulai. Enkripsi LEA-CTR kemudian dilakukan pada data plaintext, menghasilkan ciphertext, atau data yang telah dienkripsi. Setelah enkripsi berhasil, variabel end_time dibuat.



Gambar 1. Diagram Alir Pengujian Kinerja LEA Cipher

Langkah terakhir adalah menghitung perbedaan waktu antara timer awal dan akhir, yang menunjukkan durasi yang dibutuhkan oleh algoritma LEA-CTR untuk mengenkripsi data. Proses dekripsi menyerupai langkah-langkah enkripsi data sebelumnya. Setelah ciphertext diterima, variabel start_time diinisialisasi, mencatat saat tepat sebelum proses dekripsi LEA-CTR dimulai. Algoritma LEA-CTR kemudian mendekripsi ciphertext kembali menjadi plaintext, untuk mendapatkan kembali data sensor yang asli. Setelah dekripsi berhasil, variabel end_time dibuat. Tahap terakhir adalah menghitung waktu yang telah berlalu antara start_time dan end_time, yang menunjukkan durasi yang dibutuhkan oleh algoritma LEA-CTR untuk mendekripsi data. Perbedaan waktu ini kemudian dapat digunakan untuk analisis kinerja algoritma pada NodeMCU8266.

3.4 Metode Analisis data

Untuk setiap varian kunci yang berbeda, yaitu 128, 192, dan 256 bit, waktu yang dibutuhkan untuk melakukan proses enkripsi dan dekripsi akan dihitung. Pengukuran waktu ini akan dilakukan dengan menggunakan timer yang telah diintegrasikan dalam perangkat lunak pengembangan, dan akan dicatat dalam satuan waktu mikrodetik

Analisis juga akan dilakukan terhadap kinerja enkripsi dan dekripsi dengan ukuran data yang bervariasi. Hal ini akan memberikan gambaran mengenai bagaimana kinerja algoritma LEA-CTR berubah seiring dengan perubahan ukuran data yang dienkripsi dan didekripsi.

4. Pengujian

Pengujian ini bertujuan untuk mengetahui apakah program enkripsi telah berhasil menghasilkan *ciphertext* dan dapat dikembalikan ke bentuk *plaintext* seperti semula.

Tabel 2. Hasil pengujian enkripsi dan dekripsi

Key	CTR	Plaintext	Ciphertext	Decrypted Text
b'Q\xcb\x12\xbe\x82\xdc\x86\xefb+\x8b\xa34;\xab6\x15\x98\xb8:t\x8f\x83\xe5\xf6SXv#\xe3\xe8\xad'	b"\xe2\xa1'\xcf\xd6\xf5\xd7\x82\xe2\xc7H\xf05\x85J\t"	{"Humidity": 64, "Temperature": 29}	46749fa5a654479b6158146f7e14e7c6c2e78e5074412c2d4e74a689d653ee85a69562	{"Humidity": 64, "Temperature": 29}
b'Q\xcb\x12\xbe\x82\xdc\x86\xefb+\x8b\xa34;\xab6\x15\x98\xb8:t\x8f\x83\xe5\xf6SXv#\xe3\xe8\xad'	b'\xc2\x85\xcd\xe6~\x1dP\x0em\xbb\x0e\r\xa9\x01\x153'	{"Humidity": 64, "Temperature": 28}	b87bdd1bc61b92e56c01da58a12cb6ed61bc037049c973e2a03dee5aa94ba376cbf733	{"Humidity": 64, "Temperature": 28}
b'Q\xcb\x12\xbe\x82\xdc\x86\xefb+\x8b\xa34;\xab6\x15\x98\xb8:t\x8f\x83\xe5\xf6SXv#\xe3\xe8\xad'	b'+\x0c\xe2\x90\xfc\xa6\x87\xa2\x17\xc3\x87\x8e\xaeEOb'	{"Humidity": 63, "Temperature": 28}	486ce7267458523d4cab030c1362c188f1f885b50fd2e9d0aaf8a046eed8e3c8d50f54	{"Humidity": 63, "Temperature": 28}
b'Q\xcb\x12\xbe\x82\xdc\x86\xefb+\x8b\xa34;\xab6\x15\x98\xb8:t\x8f\x83\xe5\xf6SXv#\xe3\xe8\xad'	b'\xb3\x1b$\xcb\x82\xdf\x1d\xb6.\xd8\xb2\xc9\x99\x19\x19\x05'	{"Humidity": 63, "Temperature": 28}	bd84207f2f6922550914b983799dffdd49bfe1e364a671942af3ac9e658833085d4e15	{"Humidity": 63, "Temperature": 28}
b'Q\xcb\x12\xbe\x82\xdc\x86\xefb+\x8b\xa34;\xab6\x15\x98\xb8:t\x8f\x83\xe5\xf6SXv#\xe3\xe8\xad'	b'B\x97\x90W5\x0b\x17\xa1\x0f-h\xcc\x19\xb3\x06\xd7'	{"Humidity": 63, "Temperature": 28}	bd6c3a887035726c1edcaa1c8a000e3a24f7832e555e00f8deb571a272a5fb11a0c2e7	{"Humidity": 63, "Temperature": 28}
b'Q\xcb\x12\xbe\x82\xdc\x86\xefb+\x8b\xa34;\xab6\x15\x98\xb8:t\x8f\x83\xe5\xf6SXv#\xe3\xe8\xad'	b"\xb7\xc9\xb1\x17\xaf\x8f\x99_\x00)|\x16'z\xeb\xaa"	{"Humidity": 63, "Temperature": 28}	a33f66105f1d000859c0b75b75b920338e6389847d18d723ace3063cb34c8d85c5de2c	{"Humidity": 63, "Temperature": 28}
b'Q\xcb\x12\xbe\x82\xdc\x86\xefb+\x8b\xa34;\xab6\x15\x98\xb8:t\x8f\x83\xe5\xf6SXv#\xe3\xe8\xad'	b'\x99)\xc1\xd6\x9d\x13I\t\x81\x10\xe3\x1b\xd7r\xbb'	{"Humidity": 63, "Temperature": 29}	d51683806b0e42db0621df5105bedd5df17d6510a09cbe228f97edd1962ecdcf9fc819	{"Humidity": 63, "Temperature": 29}
b'Q\xcb\x12\xbe\x82\xdc\x86\xefb+\x8b\xa34;\xab6\x15\x98\xb8:t\x8f\x83\xe5\xf6SXv#\xe3\xe8\xad'	b'\x19s\x19^\x84N\xfe\xa8oM}\xf2\x08\xe0!\xd7'	{"Humidity": 64, "Temperature": 28}	96a5c3061a2a94176b20e3e99dc407c6da23b01ddb7afdc537179e9bffc65f37be31f1	{"Humidity": 64, "Temperature": 28}
b'Q\xcb\x12\xbe\x82\xdc\x86\xefb+\x8b\xa34;\xab6\x15\x98\xb8:t\x8f\x83\xe5\xf6SXv#\xe3\xe8\xad'	b'a\xeaB\x8b\xca\x86\x93\xdff\xe6}\xb1\xc5/\xa1\xe5'	{"Humidity": 64, "Temperature": 29}	8ca5881a078910d7df550fe71cd76d684c35f162d2462aaebc3d640a16b609ac44fb9a	{"Humidity": 64, "Temperature": 29}
b'Q\xcb\x12\xbe\x82\xdc\x86\xefb+\x8b\xa34;\xab6\x15\x98\xb8:t\x8f\x83\xe5\xf6SXv#\xe3\xe8\xad'	b'<@}\xcb\xfcxb\xa5\xc6s\x84\xbc\xe2\xaa\xcfi'	{"Humidity": 63, "Temperature": 29}	73e1a4697269197bc63f7f053e9b991847c6cf7b407098faf69c267ecc2586e4b7d0eb	{"Humidity": 63, "Temperature": 29}

Tabel 2 menggunakan variasi kunci 256-bit, dan temuan pada Tabel 2 mengindikasikan bahwa program enkripsi mampu menghasilkan ciphertext yang kemudian dapat dikembalikan ke plaintext aslinya.

Uji selanjutnya dilakukan untuk mengevaluasi kinerja algoritma LEA dalam melakukan enkripsi dan dekripsi mada microcontroller NodeMCU ESP8266. Serangkaian uji coba dilakukan dengan menggunakan tiga varian panjang kunci yang berbeda, yakni 128, 196, dan 256 bit. Setiap varian kunci tersebut akan diujikan secara sepuluh kali, dan dihitung kecepatan enkripsi serta deskripsinya. Selain itu dapat dihitung juga nilai *Throughput*-nya dengan persamaan berikut (Endrayanto et al., 2019):

$$Throughput = \frac{Plaintext(bit)}{Waktu\ (detik)} \quad (1)$$

Pada tabel-tabel berikut ini, waktu yang digunakan untuk menghitung *throughput* adalah waktu yang dibutuhkan dari enkripsi.

Tabel 3. Hasil Pengujian Enkripsi dan Dekripsi LEA 128-Bit

	Enkripsi (µs)	Dekripsi (µs)	Ukuran Plaintext (bytes)	Throughput (bit/detik)
1	178768	183069	35	1566.27584
2	182830	180869	35	1531.47733
3	182661	181613	35	1532.89427
4	180948	183870	35	1547.40588
5	180153	183588	35	1554.23446
6	182860	181189	35	1531.22607
7	182879	181544	35	1531.06699
8	182643	183796	35	1533.04534
9	180225	183575	35	1553.61354
10	180441	182931	35	1551.75376
AVG	**181440.8**	**182604.4**		**1543.299348**

Tabel 3 menyajikan hasil kinerja untuk kecepatan enkripsi dan dekripsi menggunakan varian kunci 128-bit dengan 24 putaran. Waktu rata-rata untuk enkripsi adalah 0.181440 detik, sedangkan waktu rata-rata untuk dekripsi adalah 0.182604 detik. *Throughput* rata rata yang dihasilkan oleh varian ini adalah 1543.29 bit/detik.

Tabel 4. Hasil Pengujian Enkripsi dan Dekripsi LEA 196-Bit

	Enkripsi (µs)	Dekripsi (µs)	Ukuran Plaintext (bytes)	Throughput (bit/detik)
1	244456	249167	35	1145.4004
2	247589	249944	35	1130.90646
3	247720	249666	35	1130.30841
4	247533	249724	35	1131.16231
5	247643	249697	35	1130.65986
6	247763	249927	35	1130.11224
7	247459	249189	35	1131.50057
8	247817	249113	35	1129.86599
9	247063	250718	35	1133.31417
10	248323	249935	35	1127.5637
AVG	**247336.6**	**249708**		**1132.079412**

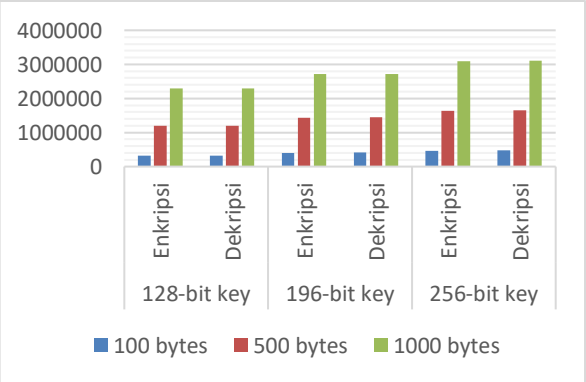
Tabel 4 menampilkan hasil kinerja terkait kecepatan enkripsi dan dekripsi untuk varian kunci 196-bit dengan 28 putaran. Rata-rata waktu enkripsi tercatat sebesar 0.247336 detik, sedangkan rata-rata waktu dekripsi adalah 0.249708 detik. *Throughput* rata rata yang dihasilkan oleh varian ini adalah 1132.0 bit/detik.

Tabel 5. Hasil Pengujian Enkripsi dan Dekripsi LEA 256-Bit

	Enkripsi (µs)	Dekripsi (µs)	Ukuran Plaintext (bytes)	Throughput (bit/detik)
1	286442	290175	35	977.510281
2	288707	289118	35	969.841396
3	290422	289524	35	964.114289
4	290831	289054	35	962.75844
5	288443	289372	35	970.729052
6	290535	289728	35	963.739309
7	290954	288938	35	962.351437
8	287749	289388	35	973.07028
9	290509	289623	35	963.825561
10	290883	288881	35	962.586332
AVG	**289547.5**	**289380.1**		**967.0526378**

Tabel 5 menunjukkan hasil kinerja untuk kecepatan enkripsi dan dekripsi menggunakan varian kunci 256-bit dengan 32 putaran. Rata-rata waktu enkripsi adalah 0.289547 detik, dan rata-rata waktu dekripsi adalah 0.289380 detik. *Throughput* rata rata yang dihasilkan oleh varian ini adalah 967.0526 bit/detik.

Dibandingkan dengan penelitian yang dilakukan oleh (Rachmayanti & Wirawan, 2022), kinerja enkripsi pada penelitian ini menunjukkan hasil yang lebih cepat. Pada (Rachmayanti & Wirawan, 2022), enkripsi membutuhkan waktu 0,560 detik untuk mengenkripsi 16 bit. Namun, jika dibandingkan dengan penelitian yang dilakukan oleh (Endrayanto et al., 2019), algoritma AES-128 menunjukkan hasil *throughput* yang lebih tinggi dibandingkan LEA-CTR pada penelitian ini. Perbedaan ini dapat dikaitkan dengan penggunaan mikrokontroler yang berbeda, (Endrayanto et al., 2019) menggunakan ARM Cortex M3, sedangkan penelitian ini menggunakan NodeMCU ESP8266.



Gambar 2. Grafik Kinerja LEA Cipher

Grafik pada Gambar 2 mengilustrasikan performa dari berbagai versi LEA Cipher dalam hal kecepatan enkripsi dan dekripsi pada berbagai ukuran. Terlihat jelas bahwa waktu yang dibutuhkan untuk enkripsi dan dekripsi meningkat ketika ukuran kunci bertambah dari 128-bit ke 256-bit. Pola ini sesuai dengan perkiraan umum, karena kunci yang lebih

panjang sering kali membutuhkan upaya komputasi yang lebih rumit. Selain itu, jumlah round yang berbeda pada setiap varian berkontribusi pada perbedaan kecepatan ini.

5. Kesimpulan dan Saran

Pengujian ini memberikan wawasan yang mendalam tentang *Lightweight Encryption Algorithm* (LEA) dengan menggunakan varian kunci 256-bit pada NodeMCU ESP8266, khususnya untuk perangkat IoT. Hasilnya, yang dirinci pada Tabel 5, mengungkapkan waktu enkripsi dan dekripsi rata-rata sekitar 0,2895 detik, dengan *throughput* 967,0526 bit per detik. Hal ini menggarisbawahi efisiensi LEA dalam menjalankan fungsi enkripsi di lingkungan dengan sumber daya yang terbatas dan kelayakannya sebagai solusi keamanan yang tangguh untuk aplikasi IoT. Penelitian ini menunjukkan peningkatan kecepatan yang jelas dibandingkan dengan algoritma AES-128 yang ditemukan dalam penelitian (Rachmayanti & Wirawan, 2022), karena implementasi LEA membutuhkan waktu yang jauh lebih sedikit untuk proses enkripsi. Hal ini secara jelas terlihat dari penelitian mereka yang membutuhkan waktu 0,560 detik untuk mengenkripsi 16-bit, yang mengindikasikan kecepatan yang lebih lambat. Namun, perlu dicatat bahwa dibandingkan dengan penelitian (Endrayanto et al., 2019) yang menggunakan algoritma AES-128 pada mikrokontroler ARM Cortex M3, varian LEA-CTR menunjukkan hasil *throughput* yang lebih rendah. Hal ini menunjukkan bagaimana perangkat keras mempengaruhi kinerja algoritma kriptografi. Meskipun NodeMCU ESP8266 memiliki keterbatasan dalam hal kemampuan komputasi dibandingkan dengan ARM Cortex M3, algoritma LEA tetap menunjukkan kinerja yang efisien dan efektif dalam konteks IoT secara umum. Untuk penelitian mendatang, disarankan untuk mengimplementasikan LEA pada bahasa pemrograman Arduino. Penggunaan bahasa pemrograman Arduino mungkin akan memberikan hasil kinerja yang lebih baik jika dibandingkan dengan micropython.

Daftar Pustaka:

Aikawa, M., Takaragi, K., Furuya, S., & Sasamoto, M. (1998). A lightweight encryption method suitable for copyright protection. *IEEE Transactions on Consumer Electronics*, *44*(3), 902–910.

Al-Amri, R. M., Hamood, D. N., & Farhan, A. K. (2023). Theoretical Background of Cryptography. *Mesopotamian Journal of Cyber Security*, *2023*, 7–15. https://doi.org/10.58496/mjcs/2023/002

Alluhaidan, A. S. D., & Prabu, P. (2023). End-to-End Encryption in Resource-Constrained IoT Device. *IEEE Access*, *11*(July), 70040–70051. https://doi.org/10.1109/ACCESS.2023.3292829

Aumasson, J.-P. (2018). *Serious Cryptography*.

Bellini, P., Nesi, P., & Pantaleo, G. (2022). IoT-Enabled Smart Cities: A Review of Concepts, Frameworks and Key Technologies. *Applied Sciences (Switzerland)*, *12*(3). https://doi.org/10.3390/app12031607

Choi, W., Kim, J., Lee, S. E., & Park, E. (2021). Smart home and internet of things: A bibliometic study. *Journal of Cleaner Production*, *301*, 126908. https://doi.org/10.1016/j.jclepro.2021.126908

Dinu, D., Corre, Y. Le, Khovratovich, D., Perrin, L., Großschädl, J., & Biryukov, A. (2019). Triathlon of lightweight block ciphers for the Internet of things. *Journal of Cryptographic Engineering*, *9*(3), 283–302. https://doi.org/10.1007/s13389-018-0193-x

Endrayanto, R. K., Muttaqin, A., & Setyawan, R. A. (2019). Advanced Encryption Standard (AES) pada Modul Internet of Things (IoT). *TELKA - Telekomunikasi, Elektronika, Komputasi Dan Kontrol*, *5*(2), 103–113. https://doi.org/10.15575/telka.v5n2.103-113

Hong, D., Lee, J. K., Kim, D. C., Kwon, D., Ryu, K. H., & Lee, D. G. (2014). LEA: A 128-bit block cipher for fast encryption on common processors. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *8267 LNCS*, 3–27. https://doi.org/10.1007/978-3-319-05149-9_1

Malik, P. K., Sharma, R., Singh, R., Gehlot, A., Satapathy, S. C., Alnumay, W. S., Pelusi, D., Ghosh, U., & Nayak, J. (2021). Industrial Internet of Things and its Applications in Industry 4.0: State of The Art. *Computer Communications*, *166*, 125–139. https://doi.org/10.1016/j.comcom.2020.11.016

Mishra, Z., Ramu, G., & Acharya, B. (2019). Hight Speed Low Area VLSI Architecture for LEA Encryption Algorithm. In *Lecture Notes in Electrical Engineering* (Vol. 556). Springer Singapore. https://doi.org/10.1007/978-981-13-7091-5_14

Pirzada, S. J. H., Murtaza, A., Xu, T., & Jianwei, L. (2019). Initialization Vector Generation for AES-CTR Algorithm to Increase Cipher-text Randomness. *2019 2nd International Conference on Information Systems and Computer Aided Education, ICISCAE 2019*, 138–142. https://doi.org/10.1109/ICISCAE48440.2019.221605

Rachmayanti, A., & Wirawan. (2022). Implementasi Algoritma Advanced Encryption Standard (AES) pada Jaringan Internet of Things (IoT) untuk Mendukung Smart Healthcare. *Jurnal Teknik ITS*, *11*(3). https://doi.org/10.12962/j23373539.v11i3.97042

Selvaraj, S., & Sundaravaradhan, S. (2020). Challenges and opportunities in IoT healthcare systems: a systematic review. *SN Applied Sciences*, *2*(1), 1–8. https://doi.org/10.1007/s42452-019-1925-y

Shahzad, K., Zia, T., & Qazi, E. U. H. (2022). A Review of Functional Encryption in IoT Applications. *Sensors*, *22*(19), 1–50. https://doi.org/10.3390/s22197567

Singh, S., Sharma, P. K., Moon, S. Y., & Park, J. H. (2017). Advanced lightweight encryption algorithms for IoT devices: survey, challenges and solutions. 4

Journal of Ambient Intelligence and Humanized Computing, *0*(0), 1–18. https://doi.org/10.1007/s12652-017-0494-

Song, J., & Seo, S. C. (2021). Efficient parallel implementation of ctr mode of arx-based block ciphers on armv8 microcontrollers. *Applied Sciences (Switzerland)*, *11*(6). https://doi.org/10.3390/app11062548

Thakor, V. A., Razzaque, M. A., & Khandaker, M. R. A. (2021). Lightweight Cryptography Algorithms for Resource-Constrained IoT Devices: A Review, Comparison and Research Opportunities. *IEEE Access*, *9*, 28177–28193. https://doi.org/10.1109/ACCESS.2021.3052867

Zhang, X., Zhou, Z., & Niu, Y. (2018). An Image Encryption Method Based on the Feistel Network and Dynamic DNA Encoding. *IEEE Photonics Journal*, *10*(4), 1. https://doi.org/10.1109/JPHOT.2018.2859257