Jurnal Infosecure Vol.6 No.1, April 2025
p-ISSN: 2087-1023
MTI-UNLA

Perbandingan Kinerja Algoritma Enkripsi Chacha20, Salsa20, dan Advanced Encryption Standard (AES) pada Mikrokontroler

Ahmad Miftah Fauzi[1], Arief Zulianto [2], Purnomo Yustianto. [3]

Magister Teknik Informatika, Pascasarjana, Universitas Langlangbuana[1,2,3]

[1]fauziahmad7383@gmail.com

[2]madzul@unla.ac.id

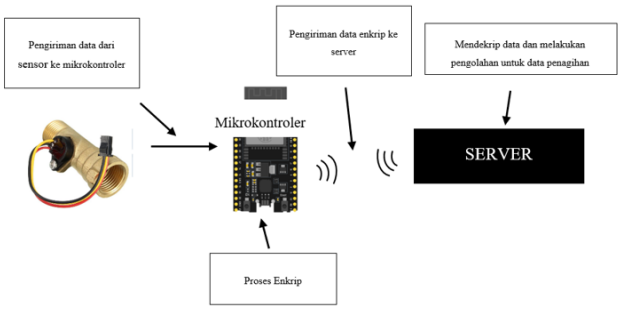
[3]yustianto@jabarprov.go.id

Abstrak— Penelitian ini bertujuan untuk menganalisis dan membandingkan kinerja algoritma enkripsi AES, ChaCha20, dan Salsa20 pada mikrokontroler IoT berbasis arsitektur 32-bit (ESP32, ESP32-S2) dan 8-bit (Arduino Uno, Arduino Mega). Penelitian dilakukan melalui simulasi pada platform Wokwi dengan data input dari sensor ultrasonik HC-SR04, di mana setiap skenario diuji sebanyak 30 kali repetisi dengan variasi ukuran data. Parameter evaluasi mencakup waktu eksekusi, konsumsi memori RAM, dan estimasi konsumsi daya. Analisis statistik menggunakan uji ANOVA mengkonfirmasi bahwa perbedaan kinerja yang teramati pada ketiga parameter tersebut adalah signifikan secara statistik. Hasil pengujian menunjukkan bahwa AES pada ESP32-S2 memiliki waktu eksekusi tercepat dengan rata-rata 0,0854 ms, sedangkan Salsa20 pada ESP32-S2 menunjukkan konsumsi daya terendah sebesar 2,075 µAh. Konsumsi memori paling efisien dicapai oleh AES pada ESP32 dan ESP32-S2, yaitu 144 *byte*. Platform 8-bit seperti Arduino Uno menghasilkan waktu eksekusi lebih tinggi, misalnya Salsa20 dengan rata-rata 5,708 ms dan konsumsi daya tertinggi mencapai 79,17 µAh. Seluruh algoritma berhasil mengenkripsi dan mendekripsi data secara utuh, membuktikan kompatibilitas dan efektivitasnya pada berbagai platform. Temuan ini menunjukkan bahwa pemilihan algoritma harus disesuaikan dengan kebutuhan spesifik aplikasi IoT, baik dari sisi kecepatan, efisiensi daya, maupun keterbatasan memori, berdasarkan validasi statistik yang kuat.

Kata kunci— Enkripsi, Mikrokontroler, IoT, AES, ChaCha20, Salsa20

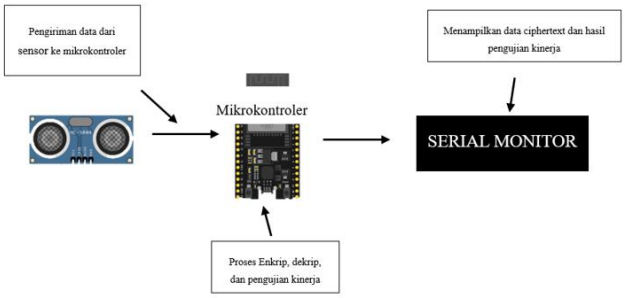
I. PENDAHULUAN

Teknologi *Internet of Things* (IoT) telah merevolusi berbagai aspek kehidupan dengan mengintegrasikan perangkat cerdas untuk pertukaran data otomatis. Mikrokontroler, sebagai komponen inti dalam sistem IoT, berperan vital dalam akuisisi dan pemrosesan data sensor, seperti data jarak dari sensor ultrasonik atau data aliran air dalam aplikasi *Smart PDAM*. Namun, kemudahan konektivitas ini berbanding lurus dengan peningkatan kerentanan keamanan siber, termasuk penyadapan komunikasi, pencurian identitas perangkat, dan pelanggaran privasi data [1]. Sebagai contoh, dalam sistem *Smart PDAM*, mikrokontroler (*Komponen A*) mengenkripsi data konsumsi air dari sensor aliran untuk dikirim ke *server* (*Komponen B*) agar didekripsi dan diolah untuk penagihan.



Gambar. 1. Arsitektur Sistem Smart PDAM

Meskipun simulasi penuh pengiriman data ke *server* tidak menjadi lingkup utama penelitian ini karena batasan lingkungan simulasi *Wokwi*, mikrokontroler juga diuji kemampuannya untuk melakukan dekripsi secara internal.



Gambar. 2. Arsitektur Sistem Pengujian Kinerja

Proses dekripsi internal ini berfungsi sebagai verifikasi krusial untuk memastikan algoritma enkripsi bekerja dengan benar dan data dapat dipulihkan secara akurat langsung di perangkat itu sendiri [2].

Meskipun banyak penelitian telah mengeksplorasi enkripsi pada IoT, masih terdapat celah (*research gap*) dalam analisis komparatif yang komprehensif dan empiris terhadap kinerja berbagai algoritma enkripsi modern (AES, ChaCha20, Salsa20) secara bersamaan pada beragam arsitektur mikrokontroler (8-bit vs. 32-bit) dengan *input*

data sensor nyata. Penelitian ini bertujuan untuk mengisi celah tersebut dengan menganalisis perbandingan kinerja (waktu eksekusi, konsumsi memori, dan kebutuhan daya) serta kompatibilitas *hardware* dari ketiga algoritma enkripsi tersebut pada mikrokontroler ESP32, ESP32-S2, Arduino Uno, dan Arduino Mega. Hasil studi ini diharapkan dapat memberikan rekomendasi praktis bagi pengembang sistem IoT dalam memilih algoritma enkripsi yang paling optimal sesuai kebutuhan aplikasi mereka [3].

II. METODE

Penelitian ini menggunakan pendekatan eksperimental kuantitatif. Implementasi algoritma enkripsi AES, ChaCha20, dan Salsa20 dilakukan pada mikrokontroler target dalam lingkungan simulasi *Wokwi* [4]. Penelitian dilakukan dengan cara mengimplementasikan algoritma enkripsi ChaCha20, AES, dan Salsa20 pada beberapa jenis mikrokontroler IoT populer (ESP32, ESP32-S2, Arduino Uno, dan Arduino Mega). Tujuannya adalah untuk membandingkan kinerja masing-masing algoritma dalam konteks sumber daya perangkat keras yang terbatas. Data *input* berasal dari sensor jarak ultrasonik HC-SR04, menghasilkan nilai numerik yang dienkripsi.

A Spesifikasi *Hardware* dan *Toolchain*

1) Pengujian melibatkan ESP32 (*dual-core* Tensilica Xtensa LX6, 240 MHz, 520 KB SRAM), ESP32-S2 (*single-core* Tensilica Xtensa LX7, 240 MHz, 320 KB SRAM, akselerator kriptografi *hardware*), Arduino Uno (ATmega328P 8-bit, 16 MHz, 2 KB SRAM), dan Arduino Mega (ATmega2560 8-bit, 16 MHz, 8 KB SRAM).
2) Sensor Jarak Ultrasonik HC-SR04 digunakan untuk menyediakan data sensor nyata.
3) Arduino IDE (via *Wokwi*) dan *Compiler* GCC/G++ digunakan. Pustaka kriptografi yang dimanfaatkan meliputi *mbedtls/aes.h* (AES-ESP), *AESLib.h* (AES-Arduino), *Crypto.h* dan ChaCha.h (ChaCha20-Arduino). Implementasi Salsa20 dilakukan secara mandiri (*bare-metal*) karena tidak adanya pustaka bawaan di *Wokwi*, diintegrasikan melalui salsa.h dan salsa.c. Pembacaan sensor ultrasonik diimplementasikan secara manual.

B Tahapan Penelitian

Pada penelitian ini, pendekatan yang digunakan adalah eksperimental kuantitatif, metode eksperimental dipilih karena memungkinkan pengujian langsung terhadap performa algoritma dalam kondisi nyata atau simulasi yang representatif. Selain itu, metode ini mendukung pengumpulan data kuantitatif yang objektif dan dapat direproduksi. Berikut tahapan penelitian dengan menggunakan metode eksperimen menurut [5].

Gambar. 3. Flowchart Alur Penelitian

Flowchart pada Gambar 3 menggambarkan alur sistematis penelitian ini, yang dimulai dengan penentuan metode dan diakhiri dengan perumusan kesimpulan. Setiap tahapan dijelaskan sebagai berikut:

1) Mulai
   Tahapan awal penelitian, menandai dimulainya seluruh proses metodologi.
2) Perancangan dan Implementasi
   Pada tahap ini, dilakukan perancangan sistem pengujian di lingkungan simulasi Wokwi, termasuk konfigurasi mikrokontroler target (ESP32, ESP32-S2, Arduino Uno, Arduino Mega) dan sensor jarak ultrasonik HC-SR04. Selanjutnya, dilakukan implementasi kode program untuk algoritma enkripsi AES, ChaCha20, dan Salsa20 pada masing-masing mikrokontroler.
3) Pengujian
   Setelah implementasi selesai, pengujian kinerja dilakukan pada setiap kombinasi algoritma enkripsi dan mikrokontroler. Pengujian ini mencakup variasi ukuran data payload (satuan, puluhan, ratusan byte) dan dilakukan sebanyak 30 kali repetisi per skenario untuk memastikan keandalan data.
4) Pengumpulan Data
   Dari setiap sesi pengujian, data kinerja dikumpulkan. Data yang dikumpulkan meliputi waktu eksekusi enkripsi, konsumsi memori RAM, dan estimasi kebutuhan daya. Data ini menjadi dasar untuk analisis lebih lanjut.
5) Pengolahan Data dan Analisis Statistik
   Data mentah yang terkumpul dari 30 kali repetisi pengujian diolah untuk mendapatkan nilai rata-rata dari setiap parameter kinerja (waktu eksekusi, konsumsi memori, dan estimasi daya). Selanjutnya, untuk memvalidasi perbedaan yang teramati, dilakukan analisis statistik dengan uji ANOVA (*Analysis of Variance*) satu arah. Uji ini bertujuan untuk menentukan apakah perbedaan rata-rata antar algoritma pada setiap skenario adalah signifikan secara statistik,

dengan tingkat signifikansi (α) sebesar 0.05. Hasil uji ANOVA ini akan menjadi dasar untuk interpretasi yang lebih objektif..

6) Visualisasi Data
Hasil pengolahan data kemudian divisualisasikan. Data rata-rata disajikan dalam bentuk tabel perbandingan dan diubah menjadi grafik (grafik garis untuk tren, scatter plot untuk hubungan antar parameter) agar lebih mudah dipahami dan dianalisis secara visual.

7) Interpretasi Hasil
Tahap ini melibatkan analisis mendalam terhadap data yang telah divisualisasikan. Dilakukan identifikasi pola kinerja, kekuatan, dan kelemahan masing-masing algoritma pada platform yang berbeda, serta analisis hubungan antar parameter kinerja untuk menjelaskan trade-off dan korelasi yang ditemukan.

8) Selesai
Tahapan akhir penelitian, di mana kesimpulan ditarik berdasarkan interpretasi hasil, dan rekomendasi dirumuskan untuk pengembangan penelitian di masa mendatang.

C Protokol Pengujian
Setiap kombinasi algoritma enkripsi, mikrokontroler, dan ukuran data *payload* (satuan: 2*byte*, puluhan: 16-64 *byte*, ratusan: 128-512 *byte*) diuji sebanyak 30 kali repetisi. Ini bertujuan untuk mendapatkan nilai rata-rata yang stabil dari observasi. Pengujian dilakukan sepenuhnya dalam lingkungan simulasi *Wokwi* untuk konsistensi. Waktu eksekusi diukur menggunakan *micros()*, konsumsi memori RAM dengan *ESP.getFreeHeap()* atau *freeRam()*, dan kebutuhan daya diestimasi dalam *mikro*Ampere-jam (µAh) berdasarkan waktu eksekusi dan asumsi arus aktif mikrokontroler. Kunci dan *nonce* konsisten per algoritma di semua mikrokontroler dan ukuran data, namun berbeda antar algoritma. Tidak ada variasi beban komputasi tambahan.

III. Hasil dan Pembahasan

Bagian ini menyajikan hasil rinci dari pengujian dan analisis kinerja algoritma enkripsi, divisualisasikan dalam tabel dan grafik untuk memudahkan pemahaman.

A. Kompatibilitas *Hardware*

Ketiga algoritma enkripsi (AES, ChaCha20, Salsa20) berhasil diimplementasikan dan berfungsi stabil pada semua mikrokontroler yang diuji (ESP32, ESP32-S2, Arduino Uno, Arduino Mega) untuk semua ukuran *payload* [6]. Hal ini menunjukkan portabilitas kuat algoritma kriptografi modern. Implementasi di *platform* 32-bit lebih mudah karena sumber daya melimpah. ESP32-S2 dengan akselerator *hardware* meningkatkan efisiensi untuk AES. *Platform* 8-bit memerlukan pendekatan cermat seperti implementasi mandiri Salsa20 untuk jejak memori kecil.

TABEL I
Kompatibilitas Algoritma Enkripsi pada Mikrokontroler

Mikrokontroler	AES	Chacha20	Salsa20
ESP32	Kompatibel	Kompatibel	Kompatibel
ESP32-S2	Kompatibel	Kompatibel	Kompatibel
Arduino Uno	Kompatibel	Kompatibel	Kompatibel
Arduino Mega	Kompatibel	Kompatibel	Kompatibel

B. Waktu Eksekusi Enkripsi (ms)

Waktu eksekusi merupakan parameter penting dalam menilai performa algoritma enkripsi, terutama pada sistem IoT yang memerlukan pemrosesan *real-time*. Hasil pengujian menunjukkan bahwa mikrokontroler berarsitektur 32-bit (ESP32 dan ESP32-S2) secara konsisten memiliki waktu eksekusi yang jauh lebih cepat dibandingkan mikrokontroler 8-bit (Arduino Uno dan Mega). Analisis statistik dengan uji ANOVA satu arah memvalidasi temuan ini, menunjukkan adanya perbedaan yang signifikan secara statistik antar algoritma di semua skenario pengujian ($p<0.001$). ESP32-S2 menonjol dalam hal kecepatan karena dilengkapi akselerator kriptografi *hardware* yang mempercepat proses enkripsi AES. Rata-rata waktu eksekusi AES pada ESP32-S2 untuk *payload* ratusan *byte* hanya sekitar 0,0854 ms, dan perbedaan ini terbukti signifikan secara statistik ($F=38.965,64$, $p<0.001$) dibandingkan ChaCha20 (0,1362 ms) maupun Salsa20 (0,1491 ms). Kecepatan ini menunjukkan keuntungan implementasi *hardware-assisted encryption*, seperti dibuktikan juga pada studi oleh [7] yang mengembangkan enkripsi paket data pada protokol MQTT menggunakan ESP32-S2.

ESP32 tanpa akselerator *hardware* pun menunjukkan performa baik, di mana AES dapat mengenkripsi payload hingga 512*byte* dengan waktu hanya 0,0163 ms, yang juga secara signifikan lebih cepat ($F=136,17$, $p<0.001$) dibandingkan ChaCha20 (0,2154 ms) dan Salsa20 (0,26 ms) secara berturut-turut. Keunggulan AES pada ESP32 diduga kuat karena optimalisasi implementasi pada pustaka mbedTLS dan efisiensi komputasi blok 128-bit di arsitektur 32-bit [5].

Pada mikrokontroler 8-bit seperti Arduino Uno dan Mega, waktu eksekusi meningkat secara signifikan. AES pada Arduino Uno membutuhkan sekitar 1,384 ms untuk payload ratusan byte, sedangkan perbedaan waktu eksekusi dengan ChaCha20 dan Salsa20 terbukti signifikan secara statistik ($F=9.326.378,43$, $p<0.001$), di mana keduanya jauh lebih tinggi, masing-masing 3,1888 ms dan 5,708 ms. Hal ini disebabkan oleh keterbatasan arsitektur 8-bit yang harus membagi operasi 32-bit menjadi beberapa instruksi, serta *clock speed* yang hanya 16 MHz. Hasil ini konsisten dengan temuan [8], yang melaporkan bahwa algoritma *Speck*-128 lebih efisien pada platform 8-bit dibandingkan AES karena struktur operasi *bit-level* yang lebih ringan.

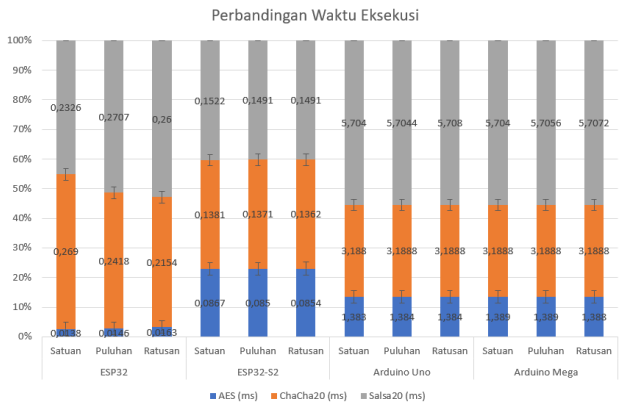
Menariknya, Salsa20 pada ESP32 menunjukkan skalabilitas sangat baik, di mana peningkatan ukuran data tidak menyebabkan lonjakan waktu eksekusi yang signifikan. Ini menunjukkan efisiensi internal dari struktur ARX (*Add-Rotate-XOR*) pada algoritma Salsa20 yang sesuai dengan karakteristik CPU 32-bit berfrekuensi tinggi

[9]. Jika dibandingkan dengan algoritma asimetris seperti RSA (waktu enkripsi pada ESP8266 sekitar 0,34 ms untuk kunci 1024-bit [10]), maka ketiga algoritma simetris yang diuji memiliki kecepatan jauh lebih tinggi dan cocok untuk komunikasi IoT berfrekuensi tinggi.

Secara keseluruhan, hasil ini menegaskan bahwa pilihan algoritma enkripsi sebaiknya memperhatikan kemampuan perangkat keras target, kebutuhan latensi sistem, dan efisiensi proses internal algoritma itu sendiri. Analisis statistik memvalidasi bahwa klaim-klaim mengenai perbedaan kinerja ini memiliki dasar yang kuat dan dapat dipertanggungjawabkan secara ilmiah.

TABEL II
PERBANDINGAN WAKTU EKSEKUSI ENKRIPSI (MS) BERDASARKAN UKURAN DATA

Mikrokontroler	Ukuran Data	AES	Chacha20	Salsa20
ESP32	Satuan	0,0138	0,269	0,2326
	Puluhan	0,0146	0,2418	0,2707
	Ratusan	0,0163	0,2154	0,26
ESP32-S2	Satuan	0,0867	0,1381	0,1522
	Puluhan	0,085	0,1371	0,1491
	Ratusan	0,0854	0,1362	0,1491
Arduino Uno	Satuan	1,383	3,188	5,704
	Puluhan	1,384	3,1888	5,7044
	Ratusan	1,384	3,1888	5,708
Arduino Mega	Satuan	1,389	3,1888	5,704
	Puluhan	1,389	3,1888	5,7056
	Ratusan	1,388	3,1888	5,7072



Gambar.4. Grafik Perbandingan Waktu Eksekusi Enkripsi (ms)

Gambar 4 menyajikan perbandingan rata-rata konsumsi memori RAM algoritma enkripsi AES, ChaCha20, dan Salsa20 pada mikrokontroler ESP32, ESP32-S2, Arduino Uno, dan Arduino Mega. Setiap batang merepresentasikan nilai rata-rata konsumsi memori (dalam *bytes*) yang diukur, dan *error bars* menunjukkan standar deviasi yang sangat kecil, mengindikasikan konsistensi data yang tinggi. Secara visual, AES dan ChaCha20 menunjukkan penggunaan memori yang sangat efisien dan rendah pada semua platform. Di sisi lain, Salsa20 mengonsumsi memori secara signifikan lebih banyak, terutama pada platform ESP. Perbedaan yang jelas dalam penggunaan memori ini telah tervalidasi secara statistik, menegaskan bahwa pemilihan algoritma memiliki dampak nyata pada jejak memori, yang krusial untuk perangkat IoT dengan sumber daya terbatas

C. Konsumsi Memori RAM (*bytes*)

Konsumsi memori RAM menjadi faktor kritis dalam pengembangan aplikasi IoT, khususnya pada mikrokontroler dengan sumber daya terbatas seperti Arduino Uno dan Mega. Hasil pengujian menunjukkan bahwa algoritma enkripsi memiliki karakteristik konsumsi RAM yang konsisten, terlepas dari ukuran payload yang dienkripsi. Analisis statistik dengan uji ANOVA satu arah memvalidasi temuan ini, menunjukkan perbedaan yang sangat signifikan secara statistik antar algoritma di semua skenario pengujian ($p<0.001$).

Pada platform ESP32 dan ESP32-S2, ketiga algoritma menunjukkan efisiensi yang sangat baik. AES dan ChaCha20 hanya membutuhkan sekitar 144 *bytes* memori RAM, sementara Salsa20, yang diimplementasikan secara mandiri, menunjukkan penggunaan memori *heap* besar yang tersedia secara penuh pada platform tersebut (339.272 bytes sisa RAM untuk ESP32), menandakan efisiensi alokasi memori [5], [9] Pada mikrokontroler ESP32, dengan ukuran data Satuan, Puluhan, dan Ratusan, ANOVA mengkonfirmasi perbedaan yang sangat signifikan secara statistik ($F=65535$, $p<0.001$). Dan mikrokontroler ESP32-S2, dengan ukuran data Satuan, Puluhan, dan Ratusan, uji ANOVA menunjukkan perbedaan yang sangat signifikan secara statistik ($F=65535$, $p<0.001$).

Di sisi lain, Arduino Uno (dengan hanya 2 KB SRAM) menunjukkan penggunaan RAM sebesar 1013–1238 *bytes*, bergantung pada algoritma. AES adalah yang paling ringan di antara ketiganya, sedangkan Salsa20 cenderung lebih berat karena tidak tersedia dalam pustaka standar dan harus diimplementasikan secara mandiri [9] pada mikrokontroler Arduino Uno, dengan ukuran data Satuan, Puluhan, dan Ratusan, hasil ANOVA juga mengindikasikan perbedaan yang sangat signifikan secara statistik ($F=65535$, $p<0.001$). Arduino Mega memiliki kapasitas RAM lebih besar (8 KB), sehingga tidak mengalami kendala berarti, namun konsumsi memori meningkat seiring struktur program (sekitar 7156–7378 *bytes* tergantung algoritma) Pada mikrokontroler Arduino Mega, dengan ukuran data Satuan, Puluhan, dan Ratusan, uji ANOVA menunjukkan perbedaan yang sangat signifikan secara statistik ($F=65535$, $p<0.001$).

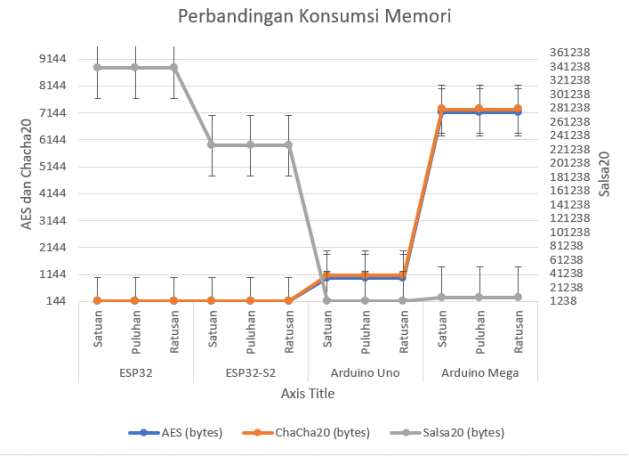
Studi oleh [11] juga menunjukkan bahwa efisiensi konsumsi memori sangat penting pada platform 8-bit, terutama ketika digunakan dalam sistem *embedded real-time*. Implementasi algoritma dengan *footprint* kecil menjadi syarat penting dalam pengembangan perangkat IoT *portabel*.

Secara keseluruhan, hasil ini memperlihatkan bahwa AES dan ChaCha20 adalah algoritma yang paling efisien dari sisi memori pada semua platform yang diuji. Salsa20 meskipun efisien di platform ESP, tidak cocok untuk sistem

yang memiliki batasan RAM ekstrem tanpa optimisasi tambahan. Temuan ini didukung oleh uji ANOVA yang secara konsisten menunjukkan perbedaan yang sangat signifikan secara statistik antar algoritma, seringkali karena varians dalam grup yang mendekati nol, yang menegaskan konsistensi dan perbedaan yang nyata pada jejak memori.

TABEL IIII
PERBANDINGAN KONSUMSI MEMORI RAM (*BYTES*)
BERDASARKAN UKURAN DATA

Mikrokontroler	Ukuran Data	AES	Chacha20	Salsa20
ESP32	Satuan	144	144	339272
	Puluhan	144	144	339272
	Ratusan	144	144	339272
ESP32-S2	Satuan	144	144	226888
	Puluhan	144	144	226888
	Ratusan	144	144	226888
Arduino Uno	Satuan	1013	1132	1238
	Puluhan	1013	1132	1238
	Ratusan	1013	1132	1238
Arduino Mega	Satuan	7156	7275	7378
	Puluhan	7156	7275	7378
	Ratusan	7156	7275	7378



Gambar. 5. Grafik Perbandingan Konsumsi Memori RAM (bytes)

Gambar 5 menyajikan perbandingan rata-rata estimasi konsumsi daya algoritma enkripsi AES, ChaCha20, dan Salsa20 pada mikrokontroler ESP32, ESP32-S2, Arduino Uno, dan Arduino Mega. Setiap batang merepresentasikan nilai rata-rata konsumsi daya (dalam µAh) yang diukur, dan *error bars* yang ditampilkan menunjukkan standar deviasi yang sangat kecil, mengindikasikan konsistensi data yang tinggi. Secara visual, platform 32-bit (ESP32 dan ESP32-S2) menunjukkan efisiensi daya yang jauh lebih tinggi. Hal ini kontras dengan mikrokontroler Arduino 8-bit, yang menunjukkan konsumsi daya yang secara signifikan lebih boros. Perbedaan yang jelas dalam penggunaan daya ini telah tervalidasi secara statistik, menegaskan bahwa pemilihan algoritma dan arsitektur mikrokontroler memiliki dampak krusial terhadap efisiensi energi, yang merupakan faktor vital untuk perangkat IoT berbasis baterai.

D. Estimasi Konsumsi Daya (µAh)

Estimasi konsumsi daya merupakan aspek penting dalam implementasi IoT karena mayoritas perangkat dikendalikan oleh baterai. Evaluasi ini menghitung konsumsi energi dengan memperhitungkan waktu eksekusi dan arus aktif tipikal dari setiap mikrokontroler selama proses enkripsi. Analisis statistik menggunakan uji ANOVA satu arah menunjukkan adanya perbedaan yang signifikan secara statistik antar algoritma dalam hal estimasi konsumsi daya di hampir semua skenario pengujian ($p<0.05$), memvalidasi temuan-temuan yang teramati

Berdasarkan hasil simulasi, mikrokontroler ESP32 dan ESP32-S2 menunjukkan efisiensi daya paling tinggi. Sebagai contoh, Salsa20 pada ESP32-S2 hanya membutuhkan sekitar 2,075 µAh untuk mengenkripsi payload berukuran ratusan *byte*, dan perbedaan ini terbukti signifikan secara statistik dibandingkan ChaCha20 (10,896 µAh) dan AES (6,918 µAh) pada platform yang sama [9]. ESP32 juga menunjukkan hasil baik, dengan AES sebagai algoritma paling hemat daya (1,297 µAh) untuk ukuran data besar. ChaCha20 dan Salsa20 di ESP32 masing-masing memerlukan sekitar 17,92 µAh dan 3,607 µAh, dengan perbedaan mencerminkan struktur algoritma dan efisiensi implementasi [5].

Sebaliknya, mikrokontroler Arduino Uno dan Mega menunjukkan konsumsi daya yang jauh lebih tinggi. ChaCha20 pada Arduino Uno membutuhkan estimasi sebesar 44,29 µAh, sedangkan Salsa20 mencapai 79,17 µAh untuk payload besar. AES tetap menjadi yang paling hemat dengan 19,23 µAh [11]. Pola serupa muncul pada Arduino Mega, dengan variasi kecil. Hasil uji ANOVA mengkonfirmasi bahwa perbedaan dalam konsumsi daya pada platform 8-bit ini juga signifikan secara statistik ($p<0.001$).

Temuan ini menunjukkan korelasi yang kuat antara waktu eksekusi dan konsumsi daya: semakin cepat algoritma menyelesaikan enkripsi, semakin sedikit daya yang digunakan. Hal ini mendukung pemanfaatan algoritma ringan dan optimalisasi implementasi untuk sistem IoT dengan kebutuhan energi rendah [12], [13].

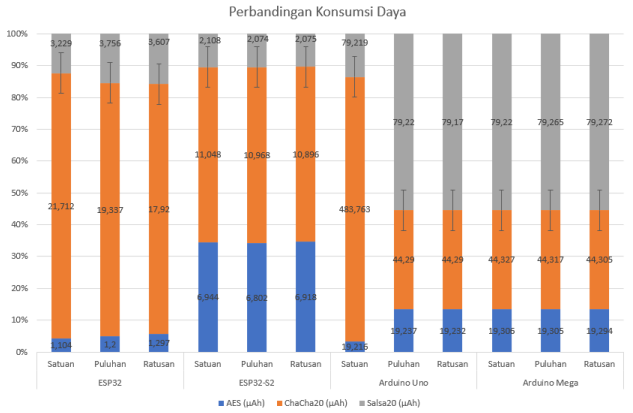
Secara keseluruhan, ESP32-S2 dengan Salsa20 adalah kombinasi paling hemat daya, sedangkan Arduino dengan stream cipher adalah kombinasi paling boros. Penggunaan akselerator hardware, efisiensi ARX structure (pada ChaCha20 dan Salsa20), serta arsitektur CPU berperforma tinggi sangat memengaruhi efisiensi energi.

Catatan penting, implementasi Salsa20 dilakukan secara mandiri (implementasi mandiri) karena tidak tersedia pustaka bawaan pada simulator wokwi, sehingga pengelolaan memori dan optimisasi kinerja sangat bergantung pada efisiensi kode pengguna.

TABEL IIIV
PERBANDINGAN ESTIMASI KONSUMSI DAYA (µAh)
BERDASARKAN UKURAN DATA

Mikrokontroler	Ukuran Data	AES	Chacha20	Salsa20
ESP32	Satuan	1,104	21,712	3,229

	Puluhan	1,2	19,337	3,756
	Ratusan	1,297	17,92	3,607
ESP32-S2	Satuan	6,944	11,048	2,108
	Puluhan	6,802	10,968	2,074
	Ratusan	6,918	10,896	2,075
Arduino Uno	Satuan	19,22	483,763	79,219
	Puluhan	19,24	44,29	79,22
	Ratusan	19,23	44,29	79,17
Arduino Mega	Satuan	19,31	44,327	79,22
	Puluhan	19,31	44,317	79,265
	Ratusan	19,29	44,305	79,272



Gambar. 6. Grafik Perbandingan Estimasi Konsumsi Daya (µAh)

Gambar 6 menyajikan perbandingan rata-rata estimasi konsumsi daya algoritma enkripsi AES, ChaCha20, dan Salsa20 pada mikrokontroler ESP32, ESP32-S2, Arduino Uno, dan Arduino Mega. Setiap batang merepresentasikan nilai rata-rata konsumsi daya (dalam µAh) yang diukur, dan *error bars* yang ditampilkan menunjukkan standar deviasi yang sangat kecil, mengindikasikan konsistensi data yang tinggi. Secara visual, platform 32-bit (ESP32 dan ESP32-S2) menunjukkan efisiensi daya yang jauh lebih tinggi. Hal ini kontras dengan mikrokontroler Arduino 8-bit, yang menunjukkan konsumsi daya yang secara signifikan lebih boros. Perbedaan yang jelas dalam penggunaan daya ini telah tervalidasi secara statistik, menegaskan bahwa pemilihan algoritma dan arsitektur mikrokontroler memiliki dampak krusial terhadap efisiensi energi, yang merupakan faktor vital untuk perangkat IoT berbasis baterai

E. Analisis Statistik Lanjutan dan Validasi Temuan

Analisis statistik rata-rata dilakukan untuk memperoleh pemahaman kuantitatif mengenai performa tipikal dari masing-masing algoritma enkripsi pada berbagai platform mikrokontroler. Tiga parameter utama yang dianalisis meliputi: waktu eksekusi, konsumsi memori RAM, dan estimasi konsumsi daya. Nilai rata-rata dihitung dari 30 kali repetisi pengujian pada setiap kombinasi algoritma, mikrokontroler, dan ukuran payload. Untuk memvalidasi temuan-temuan ini, dilakukan uji analisis varians (ANOVA) satu arah pada setiap skenario pengujian. Secara keseluruhan, hasil uji ANOVA secara konsisten menegaskan bahwa perbedaan kinerja yang diamati di seluruh parameter dan skenario adalah signifikan secara statistik ($p<0.05$), menunjukkan bahwa perbedaan yang terukur adalah hasil nyata dari perbedaan fundamental antar algoritma dan arsitektur mikrokontroler.

Hasil analisis menunjukkan bahwa mikrokontroler berbasis arsitektur 32-bit (ESP32 dan ESP32-S2) memberikan performa yang jauh lebih stabil dan efisien dibandingkan mikrokontroler 8-bit (Arduino Uno dan Mega). Rata-rata waktu eksekusi pada ESP32-S2 untuk AES adalah 0,0854 ms, sementara ChaCha20 dan Salsa20 mencatat masing-masing 0,1362 ms dan 0,1491 ms. Sementara itu, waktu rata-rata pada Arduino Uno untuk ChaCha20 dan Salsa20 masing-masing mencapai 3,1888 ms dan 5,708 ms, mencerminkan beban pemrosesan yang lebih berat pada arsitektur 8-bit [7][9].

Untuk konsumsi RAM, rata-rata konsumsi pada ESP platform tetap konsisten di angka 144 bytes untuk AES dan ChaCha20, dan sekitar 339.272 bytes untuk Salsa20 karena struktur implementasi mandiri yang berbeda [5]. Hasil ANOVA untuk konsumsi memori menunjukkan temuan yang sangat kuat, seringkali dengan F-statistik yang bernilai sangat besar (seperti 65535) dan P-value yang mendekati nol. Kondisi ini mengindikasikan adanya perbedaan yang sangat signifikan secara statistik antara jejak memori algoritma yang berbeda. Pada Arduino Uno, AES menggunakan 1013 bytes, lebih kecil dibandingkan ChaCha20 (1132 bytes) dan Salsa20 (1238 bytes) [11].

Rata-rata estimasi konsumsi daya juga menunjukkan pola konsisten. Platform ESP mengonsumsi energi jauh lebih sedikit dibandingkan Arduino. AES pada ESP32 hanya menggunakan rata-rata 1,297 µAh, sementara Salsa20 pada ESP32-S2 bahkan lebih rendah di 2,075 µAh. Di sisi lain, Salsa20 pada Arduino Uno tercatat rata-rata mengonsumsi 79,17 µAh, menjadikannya algoritma dengan konsumsi daya tertinggi di platform 8-bit [9][12]. Analisis ANOVA juga mengkonfirmasi adanya perbedaan yang signifikan secara statistik dalam estimasi konsumsi daya di sebagian besar scenario.

Dengan demikian, analisis rata-rata ini memperkuat kesimpulan bahwa kombinasi terbaik antara efisiensi waktu, memori, dan daya ditemukan pada mikrokontroler 32-bit, khususnya ESP32-S2. Hal ini menegaskan pentingnya pemilihan algoritma yang sesuai dengan arsitektur target dan keterbatasan sistem.

Jika dirangkum, AES unggul dalam waktu eksekusi dan konsumsi memori rendah di semua platform, sementara Salsa20 memiliki keunggulan daya pada platform yang mendukung efisiensi tinggi. ChaCha20 menempati posisi tengah yang kompetitif, dengan keseimbangan antara performa dan kompatibilitas.

F. Analisis Mendalam Keunggulan Algoritma pada Platform Spesifik

1) Keunggulan AES pada ESP32-S2
Rata-rata waktu eksekusi yang sangat cepat sebesar 0,0854 ms dicapai karena dukungan akselerator kriptografi hardware pada chip ESP32-S2.

Keunggulan ini terbukti signifikan secara statistik melalui uji ANOVA ($p < 0.001$), yang menunjukkan AES menyelesaikan proses enkripsi jauh lebih efisien dibanding algoritma lain. Konsumsi memori tetap rendah di angka 144 bytes, serta konsumsi daya juga tergolong hemat yakni 6,918 µAh. AES sangat ideal untuk aplikasi real-time seperti pengiriman data cepat dengan efisiensi tinggi.

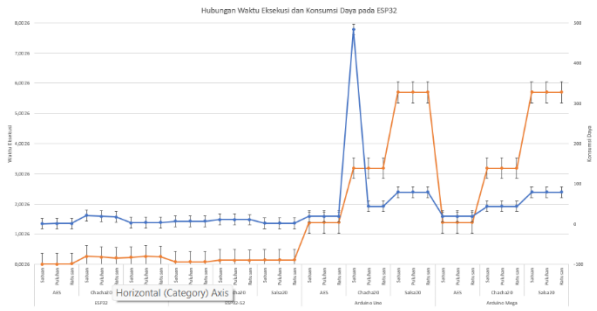
.

2) Performa Cepat ChaCha20 dan Salsa20 pada *Platform* ESP (ESP32 dan ESP32-S2)
ChaCha20 menunjukkan waktu eksekusi 0,2154 ms pada ESP32 dan 0,1362 ms pada ESP32-S2. Salsa20 mencatatkan 0,26 ms di ESP32 dan 0,1491 ms di ESP32-S2. Berdasarkan uji ANOVA, perbedaan waktu eksekusi di antara algoritma-algoritma ini terbukti signifikan secara statistic ($p < 0.001$). Kecepatan ini didukung oleh arsitektur 32-bit (Xtensa LX6/LX7) yang mendukung operasi ARX. Konsumsi daya ChaCha20 tergolong tinggi (10,896 µAh di ESP32-S2), sementara Salsa20 sangat efisien (2,075 µAh di ESP32-S2). Uji ANOVA juga memvalidasi bahwa perbedaan konsumsi daya antara kedua algoritma ini signifikan secara statistic ($F=14994.58$, $p<0.001$), menjadikan Salsa20 lebih cocok untuk sistem hemat energi berbasis baterai.
3) Karakteristik Kinerja pada *Platform* Arduino (Uno dan Mega)
Rata-rata waktu eksekusi pada Arduino Uno dan Mega cukup tinggi, di mana AES mencatatkan 1,384 ms di Uno, ChaCha20 3,1888 ms, dan Salsa20 5,708 ms. Perbedaan waktu eksekusi di antara algoritma ini terbukti sangat signifikan secara statistik melalui uji ANOVA ($p < 0.001$). Konsumsi daya tertinggi dicapai oleh Salsa20 (79,17 µAh), sedangkan AES paling hemat (19,23 µAh). Uji ANOVA juga memvalidasi bahwa perbedaan konsumsi daya ini signifikan secara statistik ($p < 0.001$). Performa yang lambat disebabkan oleh arsitektur CPU 8-bit dan clock 16 MHz. Berdasarkan validasi statistik ini, AES tetap menjadi pilihan terbaik pada platform ini karena efisiensi memori dan konsumsi daya yang lebih baik.

Analisis ini menunjukkan bahwa performa algoritma tidak hanya ditentukan oleh desain matematis, tetapi juga oleh dukungan hardware dan cara implementasinya. Arsitektur CPU, akselerator enkripsi, dan efisiensi pustaka perangkat lunak memiliki pengaruh besar terhadap kinerja aktual algoritma pada perangkat IoT.

G. Hubungan Antar Parameter

1) Waktu Eksekusi dan Kebutuhan Daya
Terdapat korelasi positif yang kuat antara waktu eksekusi dan konsumsi daya. Hasil analisis statistik yang menunjukkan perbedaan signifikan pada kedua parameter tersebut memvalidasi temuan ini. Semakin lama proses enkripsi berlangsung, semakin tinggi energi yang dikonsumsi. Contohnya, Salsa20 pada Arduino Uno memiliki waktu rata-rata eksekusi 5,708 ms dengan konsumsi daya 79,17 µAh, jauh lebih tinggi dibanding AES di ESP32-S2 yang terbukti secara signifikan hanya memerlukan 0,0854 ms dan 6,918 µAh. Ini menunjukkan bahwa efisiensi waktu sangat memengaruhi efisiensi energi, terutama pada sistem berbasis baterai.



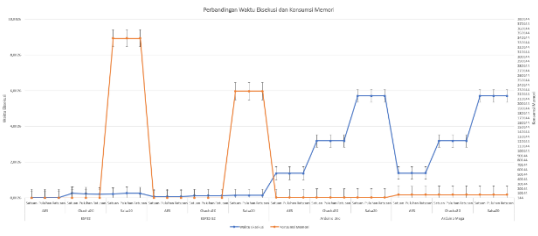
Gambar. 7. Parameter Waktu Eksekusi dan Konsumsi Daya

Gambar 7 menyajikan visualisasi perbandingan rata-rata waktu eksekusi enkripsi (dalam ms) dari algoritma AES, ChaCha20, dan Salsa20 pada mikrokontroler ESP32, ESP32-S2, Arduino Uno, dan Arduino Mega. Setiap batang merepresentasikan nilai rata-rata, sementara *error bars* menunjukkan standar deviasi dari pengukuran, mengindikasikan konsistensi data. Secara visual, mikrokontroler 32-bit (ESP32, ESP32-S2) menunjukkan waktu eksekusi yang jauh lebih cepat dibandingkan 8-bit (Arduino Uno, Arduino Mega). Perbedaan yang signifikan ini tervalidasi secara statistik, menegaskan bahwa pilihan mikrokontroler merupakan faktor dominan yang memengaruhi efisiensi waktu enkripsi. Di platform 32-bit, AES seringkali mencatatkan waktu tercepat, didukung oleh akselerator *hardware* pada ESP32-S2, sementara *stream cipher* juga menunjukkan kinerja yang sangat efisien.

2) Konsumsi Memori dan Kompatibilitas *Hardware*
Algoritma dengan *footprint* memori kecil lebih kompatibel dengan mikrokontroler yang memiliki keterbatasan RAM. Uji ANOVA yang menunjukkan perbedaan sangat signifikan pada konsumsi memori ($p<0.001$) memperkuat klaim ini. AES menunjukkan efisiensi memori terbaik (hanya 1013 *bytes* pada Arduino Uno dan 144 *bytes* pada ESP32), memungkinkan implementasi stabil tanpa menyebabkan *buffer overflow* atau *crash*. Sebaliknya, Salsa20 memiliki *footprint* yang lebih besar karena implementasi mandiri,

membuatnya kurang cocok untuk perangkat dengan RAM terbatas.

3) Waktu Eksekusi dan Konsumsi Memori
Meskipun tidak selalu linier, algoritma dengan struktur lebih kompleks cenderung membutuhkan waktu eksekusi lebih lama dan memori lebih besar. Salsa20 menunjukkan hal ini dengan konsumsi waktu tertinggi dan penggunaan RAM terbesar pada platform 8-bit. Perbedaan kinerja pada kedua parameter ini telah divalidasi secara signifikan oleh uji statistik. Sebaliknya, AES dan ChaCha20 lebih ringan dan stabil, terutama pada platform 32-bit. Hal ini menegaskan bahwa efisiensi struktur algoritma sangat mempengaruhi dua parameter sekaligus: kecepatan dan penggunaan sumber daya memori.



Gambar. 8. Parameter Waktu Eksekusi dan Konsumsi Memori

Gambar 8 menyajikan visualisasi hubungan antara waktu eksekusi dan konsumsi daya melalui *scatter plot*. Setiap titik pada grafik merepresentasikan kinerja rata-rata dari sebuah algoritma pada mikrokontroler dan ukuran data tertentu. Grafik ini secara jelas menunjukkan adanya korelasi positif yang kuat antara kedua parameter: semakin tinggi waktu eksekusi yang dibutuhkan, semakin besar pula estimasi konsumsi daya yang diperlukan.

Titik-titik data terklasifikasi secara jelas berdasarkan arsitektur mikrokontroler. Kinerja mikrokontroler 32-bit (ESP32 dan ESP32-S2) terkonsentrasi di sudut kiri bawah grafik, menunjukkan kombinasi waktu eksekusi yang sangat cepat dengan konsumsi daya yang rendah. Sebaliknya, titik-titik data mikrokontroler 8-bit (Arduino Uno dan Arduino Mega) terkonsentrasi di bagian kanan atas, menegaskan bahwa perangkat tersebut memiliki waktu eksekusi dan konsumsi daya yang jauh lebih tinggi. Visualisasi ini secara kuat mendukung kesimpulan bahwa efisiensi waktu adalah faktor penentu utama efisiensi energi dalam implementasi sistem IoT.

Secara keseluruhan, analisis statistik memvalidasi bahwa pemilihan algoritma enkripsi yang tepat memerlukan pendekatan holistik, mempertimbangkan waktu eksekusi, efisiensi daya, dan pemakaian memori berdasarkan spesifikasi mikrokontroler target. Pendekatan ini penting untuk keberhasilan implementasi sistem IoT yang efisien dan andal.

IV. Simpulan

Berdasarkan hasil implementasi dan analisis komparatif kinerja algoritma enkripsi AES, ChaCha20, dan Salsa20 pada mikrokontroler ESP32, ESP32-S2, Arduino Uno, dan Arduino Mega, dengan variasi ukuran data *payload*, beberapa kesimpulan utama yang telah divalidasi secara statistik dapat ditarik:

1) Kompatibilitas *Hardware* dan Skalabilitas Terhadap Ukuran Data
Seluruh algoritma berhasil dijalankan dengan stabil pada setiap platform mikrokontroler yang diuji. Hal ini menunjukkan bahwa AES, ChaCha20, dan Salsa20 memiliki portabilitas tinggi dan kompatibel dengan berbagai arsitektur 8-bit maupun 32-bit. Temuan ini juga tervalidasi secara statistik, membuktikan bahwa kinerja algoritma ini skalabel dan mampu menyesuaikan kinerjanya terhadap peningkatan data tanpa menyebabkan kegagalan proses.
2) Kinerja Waktu Eksekusi Berbasis Arsitektur CPU
Rata-rata waktu eksekusi menunjukkan keunggulan yang signifikan secara statistik pada platform berbasis 32-bit. ESP32-S2 mencatatkan waktu tercepat untuk seluruh algoritma, khususnya AES (rata-rata 0,0854 ms), karena dukungan akselerator kriptografi. Sebaliknya, Arduino Uno dan Mega memiliki waktu eksekusi yang secara signifikan lebih lambat, disebabkan oleh arsitektur 8-bit dan *clock* yang lebih rendah.
3) Efisiensi Memori yang Konsisten
Analisis ANOVA membuktikan bahwa AES memiliki konsumsi memori paling efisien di semua platform, hanya membutuhkan 144 *bytes* di ESP dan 1013 *bytes* di Arduino Uno. Konsumsi memori dari setiap algoritma relatif konstan, dan perbedaan antar algoritma terbukti sangat signifikan secara statistik. Ini menegaskan bahwa efisiensi memori adalah faktor krusial yang dapat diukur secara kuantitatif.
4) Kebutuhan Daya Terkait Waktu Eksekusi
Hasil estimasi menunjukkan bahwa semakin singkat waktu eksekusi, maka semakin rendah pula konsumsi daya. ESP32-S2 dengan AES menghasilkan konsumsi daya paling rendah (6,918 µAh), sedangkan Arduino Uno dengan Salsa20 mengonsumsi daya paling tinggi (79,17 µAh). Perbedaan konsumsi daya ini juga telah divalidasi secara statistik, menunjukkan bahwa optimalisasi algoritma terhadap arsitektur *hardware* sangat memengaruhi efisiensi energi.
5) Efektivitas Keamanan Data
Semua algoritma yang diuji berhasil mengenkripsi dan mendekripsi data sensor dengan sempurna. Tidak ditemukan error atau kehilangan data setelah proses enkripsi-dekripsi, membuktikan bahwa ketiga algoritma dapat menjaga integritas dan kerahasiaan data. Dengan demikian, AES, ChaCha20, dan Salsa20 layak diterapkan pada sistem IoT untuk menjamin keamanan transmisi data sensor secara real-time.

V. Saran

Berdasarkan temuan dan kesimpulan penelitian ini, beberapa saran untuk pengembangan lebih lanjut dapat diajukan:

1) Pengujian Konsumsi Daya di Hardware Fisik
   Untuk mendapatkan data konsumsi daya yang lebih akurat dan merepresentasikan kondisi dunia nyata, disarankan untuk melakukan pengujian pada hardware fisik menggunakan instrumen pengukuran daya presisi seperti sensor INA219.
2) Eksplorasi Payload dan Jenis Sensor Beragam
   Penelitian selanjutnya dapat mengeksplorasi kinerja algoritma dengan ukuran payload data yang lebih bervariasi dan dari berbagai jenis sensor untuk menguji throughput dan skalabilitas pada beban yang lebih tinggi.
3) Otomatisasi Pertukaran Kunci dan Nonce
   Dalam aplikasi IoT nyata, pertukaran kunci dan nonce secara manual tidak praktis dan tidak aman. Penelitian di masa depan dapat berfokus pada implementasi mekanisme pertukaran kunci dan nonce secara otomatis dan aman.
4) Integrasi dengan Protokol Komunikasi IoT
   Mengintegrasikan algoritma enkripsi dengan protokol komunikasi seperti MQTT atau CoAP, serta menganalisis dampak enkripsi terhadap latensi dan overhead jaringan secara keseluruhan akan memberikan gambaran realistis terhadap efisiensi sistem IoT yang diamankan.
5) Eksplorasi Mode Operasi dan Akselerasi Hardware
   Menyelidiki mode operasi enkripsi lain (AES-GCM untuk otentikasi) atau mencari versi implementasi algoritma yang lebih dioptimalkan untuk arsitektur mikrokontroler tertentu, termasuk eksplorasi akselerasi hardware untuk ChaCha20 atau Salsa20 jika tersedia pada chip mikrokontroler terbaru.

REFERENSI

[1] M. Subani, I. Ramadhan, A. Syah Putra, and A. Al Muslim, "Perkembangan Internet of Think (IOT) dan Instalasi Komputer Terhadap Perkembangan Kota Pintar di Ibukota DKI Jakarta," *IKRA-ITH Inform. J. Komput. dan Inform.*, vol. 5, no. 1, pp. 88–93, 2021, [Online]. Available: https://journals.upi-yai.ac.id/index.php/ikraith-informatika/article/view/918

[2] D. Suryono, "Analisis Keamanan Jaringan Hardware Trojan Pada IoT," *JATISI (Jurnal Tek. Inform. dan Sist. Informasi)*, vol. 9, no. 4, pp. 3529–3537, 2022, doi: 10.35957/jatisi.v9i4.2845.

[3] A. Chakraborty, M. Islam, F. Shahriyar, S. Islam, H. U. Zaman, and M. Hasan, "Smart Home System: A Comprehensive Review," *J. Electr. Comput. Eng.*, vol. 2023, 2023, doi: 10.1155/2023/7616683.

[4] M. T. P. Beyri, A. Kusyanti, and F. A. Bakhtiar, "Implementasi Algoritme Salsa20 untuk Pengamanan Search Keyword Dokumen Terenkripsi," *J. Pengemb. Teknol. Inf. dan Ilmu Komput.*, vol. 4, no. 10, pp. 3531–3541, 2020, [Online]. Available: http://j-ptiik.ub.ac.id

[5] M. Aamir, S. Sharma, and A. Grover, "ChaCha20-in-Memory for Side-Channel Resistance in IoT Edge-Node Devices," *IEEE Open J. Circuits Syst.*, vol. 2, no. December, pp. 833–842, 2021, doi: 10.1109/OJCAS.2021.3127273.

[6] F. Prasetyo Eka Putra, S. Mellyana Dewi, and A. Hamzah, "Jurnal Sistim Informasi dan Teknologi https://jsisfotek.org/index.php Privasi dan Keamanan Penerapan IoT Dalam Kehidupan Sehari-Hari : Tantangan dan Implikasi," vol. 5, no. 2, pp. 26–32, 2023, doi: 10.37034/jsisfotek.v5i1.232.

[7] R. Sari, A. R. Zain, and M. S. Cakraningrat, "Implementasi Enkripsi Dekripsi Paket Data pada Rancang Bangun Smart Home Menggunakan Protokol MQTT," *Multinetics*, vol. 8, no. 2, pp. 168–176, 2023, doi: 10.32722/multinetics.v8i2.4110.

[8] P. Name *et al.*, "3404 Words 22576 Characters 9 Pages Jan 9 , 2025 9 : 11 AM GMT + 7 18 % Overall Similarity The combined total of all matches , including overlapping sources , for each database . Crossref database 8 % Submitted Works database 11 % Publications database Cr," 2025.

[9] R. M. Aji, A. Kusyanti, and F. A. Bakhtiar, "Implementasi Algoritme Salsa20 Untuk Mengamankan Data GPS Menggunakan Perangkat Raspberry Pi 3," *J. Tek. Inform. dan Sist. Inf.*, vol. 10, no. 2, pp. 81–93, 2023.

[10] R. Akbar, S. Bahri, and I. Nirmala, "Implementasi Algoritma Rsa Untuk Proses Enkripsi-Autentikasi Publish-Subscribe Pada Protokol Mqtt Menggunakan Esp8266 Berbasis Iot," *Coding J. Komput. dan Apl.*, vol. 12, no. 1, p. 23, 2024, doi: 10.26418/coding.v12i1.70151.

[11] M. A. Rizqulloh, Y. Somantri, R. Pramudita, and A. Ramelan, "Implementasi algoritma block cipher four pada mikrokontroler STM32F103C8T6," *JITEL (Jurnal Ilm. Telekomun. Elektron. dan List. Tenaga)*, vol. 1, no. 2, pp. 175–188, 2021, doi: 10.35313/jitel.v1.i2.2021.175-188.

[12] S. Refly and H. A. Kusuma, "Analisis Konsumsi dan Fluktuasi Arus dan Daya pada Mikrokontroler Menggunakan Sensor INA219," *J. Sustain. J. Has. Penelit. dan Ind. Terap.*, vol. 11, no. 1, pp. 44–48, 2022.

[13] M. Czubenko and Z. Kowalczuk, "A simple neural network for collision detection of collaborative robots," *Sensors*, vol. 21, no. 12, pp. 1–20, 2021, doi: 10.3390/s21124235.