JURIKOM (Jurnal Riset Komputer), Vol. 13 No. 1, Februari 2026
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v13i1.9524
Hal 430-440
https://ejurnal.stmik-budidarma.ac.id/index.php/jurikom

Evaluasi Kinerja Algoritma AES-128 dan SPECK-128/128 pada Sistem Smart Door Lock Berbasis IoT

Yunita Sari, Galura Muhammad Suranegara*

Sistem Telekomunikasi, Universitas Pendidikan Indonesia, Purwakarta, Indonesia
Email: [1]yunitas.21@upi.edu, [2*]galurams@upi.edu
Email Penulis Korespondensi: galurams@upi.edu
Submitted **21-01-2026;** Accepted **24-02-2026;** Published **28-02-2026**

Abstrak

Internet of Things (IoT) semakin banyak digunakan dalam sistem rumah pintar, salah satunya Smart Door Lock. Namun, komunikasi data pada perangkat IoT rentang terhadap serangan jaringan seperti sniffing dan man-in-the-middle, terutama karena banyak perangkat masih dirancang dengan mekanisme keamanan lemah. Masalah utama yang diteliti dalam penelitian ini adalah bagaimana melindungi data RFID agar tidak mudah disadap, sekaligus tetap mempertahankan efisiensi pada perangkat dengan keterbatasan sumber daya. Solusi yang ditawarkan adalah penerapan algoritma kriptografi AES-128 dan SPECK-128/128 pada protokol komunikasi MQTT menggunakan perangkat ESP8266. Penelitian dilakukan secara eksperimental dengan 50 kali pengukuran untuk masing-masing algoritma, mencakup parameter waktu enkripsi dan dekripsi, penggunaan memori, serta efektivitas enkripsi terhadap sniffing. Hasil menunjukkan bahwa AES-128 memiliki performa stabil namun membutuhkan waktu eksekusi dan memori lebih besar karena proses algoritmanya kompleks dengan banyak tahapan. Sebaliknya, SPECK-128/128 lebih cepat dan hemat memori berkat struktur sederhana berbasis operasi ARX, meski konsistensinya sedikit lebih rendah. Analisis sniffing memperlihatkan bahwa kedua algoritma mampu mengenkripsi seluruh paket dengan tingkat keamanan 100%, sehingga tidak ada data plaintext yang terbaca. Dengan demikian, penelitian ini menegaskan adanya trade-off antara keamanan efisiensi AES lebih sesuai untuk sistem dengan kapasitas komputasi tinggi, sedangkan SPECK lebih relevan untuk perangkat IoT low-end..

Kata kunci : Internet of Things (IoT); Smart Door Lock, RFID; Protokol MQTT; AES-128; SPECK-128/128; ESP8266;

Abstract

The Internet of Things (IoT) is increasingly being used in smart home systems, one of which is the Smart Door Lock. However, data communication on IoT devices is vulnerable to network attacks such as sniffing and man-in-the-middle, especially since many devices are still designed with weak security mechanisms. The main issue studied in this research is how to protect RFID data from being easily intercepted, while maintaining efficiency on devices with limited resources. The proposed solution is the implementation of AES-128 and SPECK-128/128 cryptographic algorithms on the MQTT communication protocol using ESP8266 devices. The research was conducted experimentally with 50 measurements for each algorithm, covering encryption and decryption time parameters, memory usage, and encryption effectiveness against sniffing. The results show that AES-128 has stable performance but requires more execution time and memory because its algorithm is complex with many stages. In contrast, SPECK-128/128 is faster and more memory-efficient thanks to its simple ARX-based structure, although its consistency is slightly lower. Sniffing analysis shows that both algorithms are capable of encrypting all packets with 100% security, so that no plaintext data can be read. Thus, this study confirms that there is a trade-off between security and efficiency: AES is more suitable for systems with high computing capacity, while SPECK is more relevant for low-end IoT devices.

Keywords : Internet of Things (IoT); Smart Door Lock; RFID; MQTT Protocol; AES-128; SPECK-128/128; ESP8266;

1. PENDAHULUAN

Perangkat berbasis *Internet of Things (IoT)* umumnya terhubung dengan perangkat jaringan yang apabila tidak dilindungi dengan mekanisme keamanan yang memadai akan terjadinya isu keamanan dan privasi sebagai masalah utama dalam penerapan *IoT*, khususnya pada sistem rumah pintar yaitu *Smart Door Lock* [1]. Keberhasilan penerapan *IoT* sangat bergantung pada kualitas keamanan siber yang diterapkan, karena *IoT* berisiko mudah diretas, diakses dan bahkan dimodifikasi oleh pihak yang tidak berwenang [2]. Beberapa penelitian menunjukkan bahwa banyak perangkat *IoT* masih dirancang dengan sistem keamanan yang lemah, contohnya hanya mengandalkan kata sandi atau *autentikasi* sederhana, menjadikannya target serangan siber yang dapat mengakibatkan pencurian data, maupun pelanggaran privasi [3]. Selain itu, *IoT* menghasilkan data dalam jumlah besar dari berbagai sumber sehingga isu privasi terkait pengumpulan, penggunaan, dan penyimpanan data menjadi semakin krusial. Karena perangkat ini terhubung langsung ke internet, maka kerentanan terhadap serangan siber meningkat dan tanpa mekanisme keamanan yang memadai informasi pribadi dapat diakses oleh pihak tidak berwenang [4].

Selain itu, dalam lingkungan *IoT*, sistem harus mampu mengintegrasikan berbagai jenis data yang berasal dari beragam komponen sensor yang berbeda sehingga diperlukan mekanisme komunikasi yang baik untuk menjamin pertukaran data [5]. Sistem *IoT* juga berhadapan dengan risiko yang lebih krusial, yakni potensi kebocoran data serta isu privasi. Berikut adalah jenis serangan yang dapat terjadi pada sistem *IoT*, mulai dari serangan *Distributed Denia of Service (DDoS)* yang memungkinkan peretas mengganggu jaringan dengan memberikan permintaan secara terus menerus pada lalu lintas komunikasi yang akan membuat server mengalami *lag* dan bingung saat akan melanjutkan interaksi dengan server lain atau *user*[6]. Penelitian lain menunjukkan bahwa serangan *DDoS* dapat menyebabkan kelebihan beban pada lalu lintas dan mengakibatkan server *down*, sehingga sistem tidak tersedia bagi pengguna yang sah[7]. Bentuknya dapat berupan SYN Floods, UDPflood, maupun metode *Slowloris* yang mengeksploitasi protokol komunikasi. Dalam skala lebih besar, *DDoS* memanfaatkan bonet untuk meluncurkan serangan dari berbagai sumber sekaligus, sehingga lebih sulit

JURIKOM (Jurnal Riset Komputer), Vol. 13 No. 1, Februari 2026
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v13i1.9524
Hal 430-440
https://ejurnal.stmik-budidarma.ac.id/index.php/jurikom

terdeteksi dan berpotensi menimbulkan *down*tim serta kerugian [8]. Selain itu, terdapat pula serangan *Man in the Middle (MitM)* memungkinkan peretas menyusup di antar dua pengguna dan pengguna tidak akan menyadari bahwa ada pihak ke tiga saat mereka sedang berkomunikasi[6].

Salah satu cara untuk mengamankan sistem *IoT* yang efektif adalah dengan implementasi kriptografi. Kriptografi merupakan teknik yang digunakan untuk menjaga kerahasiaan serta keamanan pesan ketika dikirimkan ke penerima, sehingga dapat mencegah upaya penyadapan. Proses ini melibatkan tiga komponen utama, yaitu *enkripsi*, *dekripsi*, dan kunci. Inti dari kriptografi adalah mengubah teks asli (*plaintext*) menjadi teks sandi (*ciphertext*) dengan memanfaatkan kunci, sehingga informasi tidak dapat dipahami oleh pihak yang tidak berwenang, tanpa harus menyembunyikan algoritma yang digunakan[9].

Untuk mengantisipasi beragam ancaman pada sistem *IoT*, termasuk *sniffing*, *man-in-the-middle (MitM)*, dan *brute-force*, dibutuhkan mekanisme keamanan yang kokoh. Salah satu solusi yaitu penerapan kriptografi, khususnya algoritma *enkripsi*. AES-128 dipilih karena aman dan efisien untuk perangkat *IoT* dengan keterbatasan memori. Penelitian menunjukkan implementasinya pada modul *Particle Photon* tetap berjalan baik tanpa *hardware accelerator*, dengan waktu *enkripsi* sekitar 371–398 µs, *throughput* ±315 ribu bit/detik, serta penggunaan memori kecil ±16 KB *flash* dan 3 KB RAM[10]. AES-128 dipakai untuk mengamankan data RFID lewat MQTT.

Hasilnya, data lebih aman, RAM terpakai lumayan banyak, *delay* sedikit naik, tapi *throughput* jadi lebih tinggi. Jadi gabungan AES-128 sama MQTT dan RFID bisa membuat *IoT* tetap jalan lancar tapi lebih aman[11]. Selain AES-128, ada juga algoritma ringan SPECK-128/128 buatan NSA. Desainnya pakai operasi sederhana *Add-Rotate-XOR (ARX)*, jadi *enkripsi* dan *dekripsi* lebih cepat serta hemat memori. Uji coba menunjukkan waktu *enkripsi* sekitar 0,85 ms, *dekripsi* 0,87 ms, dengan penggunaan memori hanya 12,5 KB. Karena ringan, SPECK cocok untuk *IoT* dengan keterbatasan daya dan memori, meski tingkat keamanannya sedikit di bawah AES-128[12].

Namun, penelitian sebelumnya masih memiliki keterbatasan. Sebagian besar *Smart Door Lock* berbasis *IoT* hanya mengandalkan kontrol akses melalui RFID atau aplikasi Blynk tanpa adanya mekanisme *enkripsi* yang kuat, sehingga data komunikasi tetap rentan terhadap serangan. Di sisi lain, penelitian kriptografi ringan pada *IoT* lebih banyak berfokus pada analisis algoritma secara terpisah, bukan pada integrasi langsung ke sistem keamanan fisik seperti *Smart Door Lock*.

Keterbaruan penelitian ini adalah mengintegrasikan algoritma kriptografi AES-128 dan SPECK-128/128 pada sistem *Smart Door Lock* berbasis *IoT* dengan protokol MQTT. Pendekatan ini tidak hanya meningkatkan keamanan komunikasi data RFID, tetapi juga mempertimbangkan efisiensi memori dan waktu eksekusi pada perangkat dengan keterbatasan sumber daya. Dengan demikian, penelitian ini memberikan kontribusi nyata dalam pengembangan solusi keamanan rumah pintar yang lebih aman, efisien, dan ekonomis dibandingkan penelitian sebelumnya.

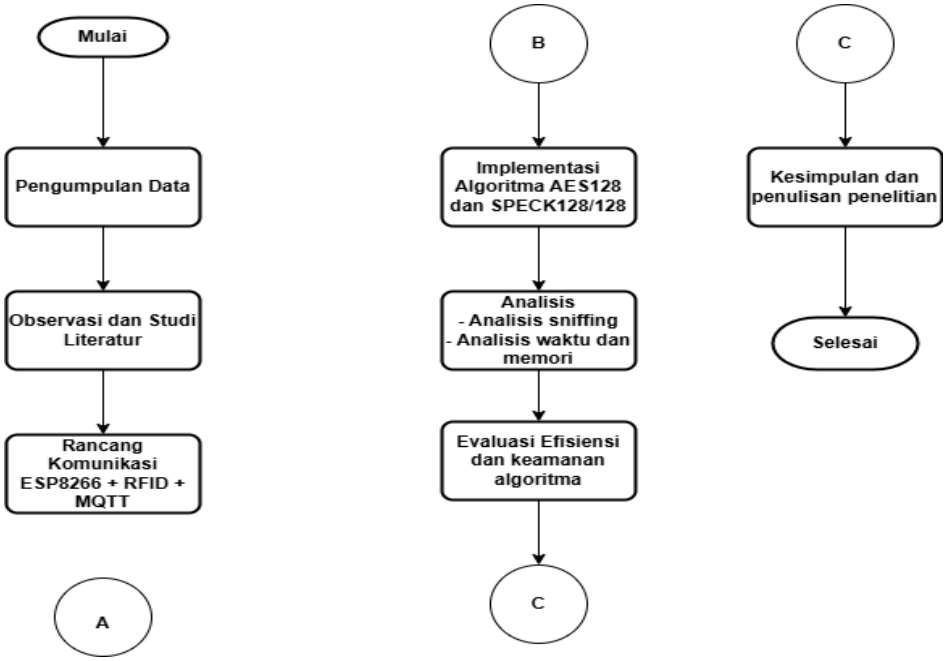
Tujuan dari penelitian ini adalah untuk mengembangkan sistem *Smart Door Lock* berbasis *IoT* yang lebih aman dengan mengintegrasikan algoritma kriptografi AES-128 dan SPECK-128/128 pada protokol MQTT. Penelitian ini diharapkan dpata memberikan solusi terhadap masalah utama berupa kerentanan data komunikasi RFID yang mudah disadap, serta meningkatkan efisiensi penggunaan memori dan waktu eksekusi pada perangkat *IoT* dengan sum ber-daya terbatas. Dengan demikian, penelitian ini bertujuan tidak hanya memperkuat aspek keamanan, tetapi juga memastikan sistem tetap efisien pada perangkat *IoT low-end.*.

2. METODOLOGI PENELITIAN

2.1 Tahapan Penelitian

Penelitian dilakukan di Laboratorium Transmisi Universitas Pendidikan Indonesia dengan metode eksperimental dan pendekatan kuantitatif yang berfokus pada analisis performa serta keamanan algoritma AES-128 dan SPECK-128/128 pada sistem *Smart Door Lock* berbasis *IoT* menggunakan protokol komunikasi MQTT. Tahapan penelitian divisualisasikan dalam bentuk *flowchart* sebagaimana ditunjukkan pada Gambar 1.

JURIKOM (Jurnal Riset Komputer), Vol. 13 No. 1, Februari 2026
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v13i1.9524
Hal 430-440
https://ejurnal.stmik-budidarma.ac.id/index.php/jurikom



Gambar 1. *Flowchart* Penelitian

Seperti terlihat pada Gambar 1, penelitian dimulai dari tahap pengumpulan data, dilanjutkan dengan observasi dan studi literatur, perancangan komunikasi ESP8266 + RFID + MQTT, implementasi algoritma, hingga uji performa *enkripsi* dan *dekripsi*. Selanjutnya dilakukan uji dan analisis memori, waktu, serta *sniffing*, yang kemudian dievaluasi untuk menentukan efisiensi dan keamanan algoritma sebelum ditarik kesimpulan.

2.2. Pengumpulan Data

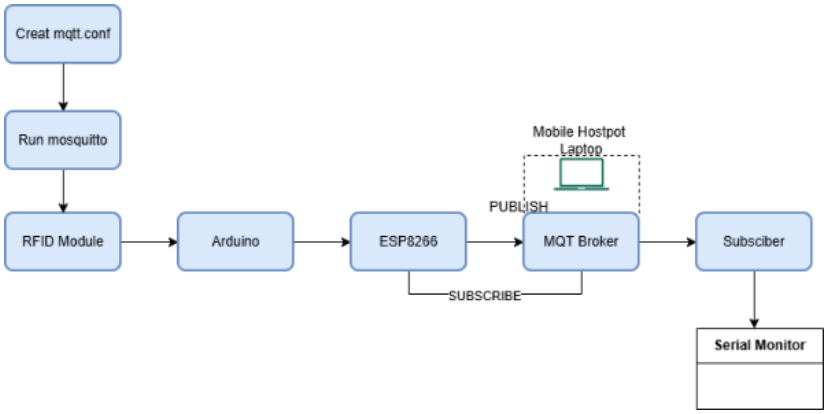
Untuk memperoleh informasi awal mengenai karakteristik perangkat ESP8266, protokol komunikasi MQTT, sensor RFID, serta algoritma kriptografi ringan seperti AES dan SPECK,

2.3. Observasi dan Studi Literatur

Melakukan observasi dan studi literatur terhadap sistem komunikasi *IoT* yang umum digunakan serta kajian berbagai sumber terkait algoritma *enkripsi*, efisiensi performa, dan keamanan data pada perangkat *IoT low-end.*, yang menjadi dasar dalam perancangan sistem dan penentuan parameter pengujian.

2.4. Perancangan Komunikasi ESP 8266 – RFID + MQTT

Tahapan komunikasi perangkat keras hingga proses *publish-subscribe* pada broker MQTT divisualisasikan dalam proses sistem sebagaimana ditunjukkan pada Gambar 2.



Gambar 2. Perancangan Komunikasi Sistem

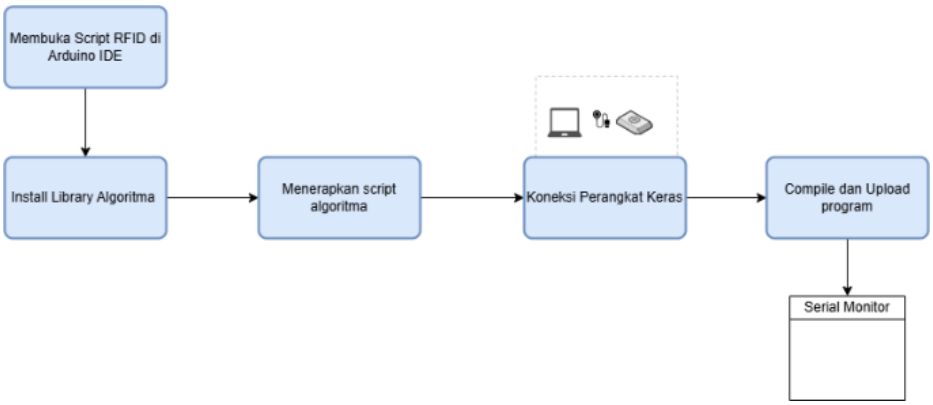
Gambar 2 menunjukkan proses komunikasi sistem, dimulai dari proses instalasi dan konfigurasi broker MQTT sampai data ditampilkan pada serial monitor. Langkah pertama mengunduh dan memasang Mosquitto sebagai broker MQTT, selanjutnya dilakukan pembuatan *file* mqtt.conf, dan menjalankan broker. Broker berhasil dijalankan dan berhasil mendengarkan koneksi pada *port* 1883, broker pun siap menerima komunikasi dari perangkat *IoT*.

JURIKOM (Jurnal Riset Komputer), Vol. 13 No. 1, Februari 2026
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v13i1.9524
Hal 430-440
https://ejurnal.stmik-budidarma.ac.id/index.php/jurikom

Setelah broker aktif, sistem perangkat keras mulai bekerja. Modul RFID membaca UID dari kartu akses dan mengirimkan data ke Arduino sebagai pengatur utama. Lalu Arduino meneruskan data ke modul ESP8266, yang berfungsi sebagai penghubung ke jaringan WiFi melalui *hotspot* laptop. Sebagai tanda akses berhasil, LCD yang terhubung ke Arduino menampilkan pesan "*Access Guarantee*". Lalu melalui koneksi WiFi, ESP8266 mengirimkan data ke broker MQTT dengan mekanisme *publish–subscribe*. ESP8266 bertindak sebagai *publisher* yang mengirimkan data UID, sementara *subscriber* menerima data tersebut untuk dapat diproses.

2.5. Implementasi Algoritma

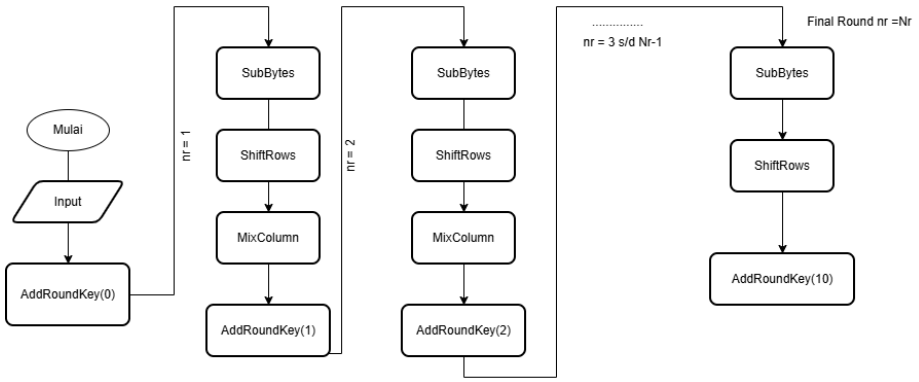
Langkah-langkah implementasi algoritma AES-128 dan SPECK-128/128 divisualisasikan dalam *flowchart* pada Gambar 3.



Gambar 3. Alur Implementasi Algoritma

Gambar 3 menunjukkan proses implementasi algoritma AES dan SPECK pada ESP8266 dengan modul RFID dilakukan secara bertahap. Langkah pertama, membuka kode RFID yang telah tersedia di Arduino IDE sebagai dasar sistem. Selanjutnya, *library* kriptografi AES dan SPECK dipasang melalui *library manager* agar fungsi *enkripsi* dapat digunakan. Setelah itu, algoritma dituliskan dan diterapkan ke dalam kode utama. ESP8266 kemudian dihubungkan ke laptop menggunakan kabel *micro* USB untuk proses *compile* dan *upload* program. Jika saat *compile* berhasil, perangkat akan menjalankan algoritma dan hasilnya dianalisis melalui Serial Monitor, parameter-Nya meliputi waktu eksekusi, penggunaan memori, serta efektivitas *enkripsi* terhadap data UID. Paparan proses algoritma AES-128 dan SPECK - 128/128 dapat dilihat pada Gambar 4 dan Gambar 5.

a. Proses AES 128



Gambar 4. Proses Algoritma AES-128

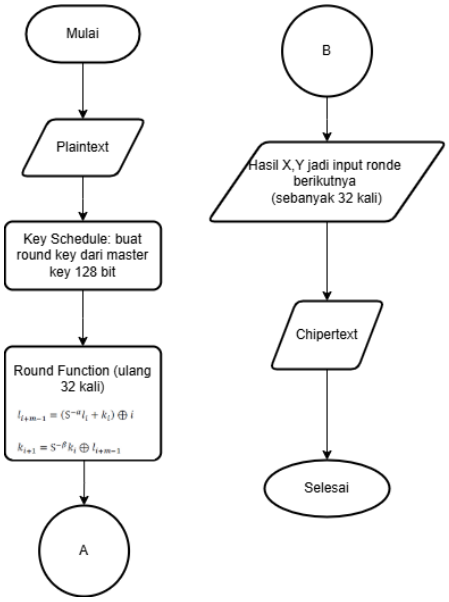
Sebagaimana divisualisasikan pada Gambar 4, *Enkripsi* AES-128 bekerja pada blok data berukuran 128 bit yang direpresentasikan sebagai matriks State berukuran 4 × 4 *byte*. Proses dimulai dengan *AddRoundKey*(0), yaitu penggabungan awal antara *plaintext* dan kunci utama menggunakan operasi XOR. Selanjutnya dilakukan 10 putaran (Nr = 10). Putaran 1-9 terdiri dari empat transformasi berurutan:

1. *SubBytes*(): substitusi non-linear setiap *byte* dengan nilai dari tabel S-box.
2. *ShiftRows*(): pergeseran siklik baris untuk menyebarkan data.
3. *MixColumns*(): pencampuran linier tiap kolom menggunakan operasi matriks di GF(2^8).
4. *AddRoundKey*(n): penggabungan State dengan kunci putaran ke-n.

Pada putaran terakhir (round ke-10), langkah *MixColumns*() dihilangkan sehingga hanya dilakukan *SubBytes*, *ShiftRows*, dan *AddRoundKey*[13].

b. Proses SPECK-128/128

JURIKOM (Jurnal Riset Komputer), Vol. 13 No. 1, Februari 2026
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v13i1.9524
Hal 430-440
https://ejurnal.stmik-budidarma.ac.id/index.php/jurikom



Gambar 5. Proses SPECK-128/128

Seperti ditunjukan pada Gambar 5, proses *enkripsi* algoritma SPECK diawali dengan masukan berupa *plaintext* yang dibagi menjadi dua bagian data, yaitu kata kiri dan kata kanan. Sebelum proses *enkripsi* utama dilakukan, algoritma SPECK membangkitkan serangkaian *round key* melalui tahap *key schedule* yang diturunkan dari kunci utama, di mana fungsi putaran SPECK juga dimanfaatkan untuk menghasilkan satu kunci pada setiap putaran. Selanjutnya, *plaintext* diproses melalui fungsi putaran (*round function*) yang diulang sebanyak 32 kali untuk varian SPECK 128/128. Pada setiap putaran, data diperbarui menggunakan operasi *Add-Rotate-XOR (ARX)*, yaitu rotasi bit, penjumlahan modulo, dan operasi XOR, sehingga menghasilkan pasangan data baru yang digunakan sebagai masukan pada putaran berikutnya. Setelah seluruh putaran selesai, pasangan data terakhir yang dihasilkan merupakan *ciphertext*. Struktur sederhana berbasis ARX ini menjadikan SPECK efisien dalam implementasi perangkat dengan sumber daya terbatas, khususnya pada sistem berbasis perangkat lunak dan *IoT*[14] .

2.6. Uji dan Analisis *Memory Library*

Library yang dianalisis terdiri dari dua algoritma kriptografi, yaitu AES dan SPECK. *Library* dari algoritma AES yaitu AESLib Versi 2.3.6, yang secara khusus fungsinya untuk mengimplementasikan algoritma AES. Pada perangkat berbasis Aduino atau ESP8266. *Library* ini mendukung penggunaan berbagai panjang kunci dan mode operasi AES, termasuk *Chiper Block Chaining (CBC)* dengan *Initialization Vector (IV)*. AESLib dapat berjalan di sebagian besar perangkat Arduino, meskipun kecepatan operasinya bisa terbatas karena keterlambatan sumber daya[15]. Selain AES, algoritma kriptografi lain yang di analisis adalah SPECK, yang diimplementasikan melalui SPECK *Library* atau secara manual dalam bentuk kode karena strukturnya sederhana berbasis operasi *Addition-Rotation-XOR (ARX)*.

Fungsi utamanya adalah menyediakan *enkripsi* ringan dengan efisiensi tinggi, sehingga cocok digunakan pada perangkat *IoT* dengan keterbatasan memori[16]. Setelah membahas implementasi algoritma kriptografi melalui *library* AESLib dan SPECK, perhatian berikut adalah bagaimana perangkat terbatas seperti Arduino atau ESP8266 mengatur sumber daya internalnya. Salah satu aspek yang paling menentukan performa adalah memori, karena efisiensi memori berpengaruh pada kelancaran proses *enkripsi* dan *dekripsi*. Semakin kecil penggunaan memori, semakin sesuai algoritma tersebut untuk perangkat dengan keterbatasan sumber daya[17]. Dengan demikian, meskipun AESLib menawarkan fitur lengkap, konsumsi memori yang lebih besar membuatnya kurang efisien pada perangkat yang memiliki sumber daya terbatas. Sebaliknya, SPECK dengan struktur sederhana berbasis ARX lebih hemat memori dan lebih sesuai untuk perangkat dengan keterbatasan sumber daya.

2.7. Uji dan Analisis waktu *enkripsi* dan *dekripsi*

Data hasil pengujian waktu *enkripsi* dan *dekripsi* dari masing-masing algoritma (AES-128 dan SPECK-128/128) dianalisis secara statistik untuk melihat perbedaan performa. Untuk keperluan tersebut digunakan metode Kruskal–Wallis merupakan salah satu metode statistik non-parametrik yang digunakan untuk membandingkan lebih dari satu kelompok independen. Teknik ini sangat bermanfaat ketika data yang dianalisis berbentuk ordinal atau tidak memenuhi asumsi distribusi normal. Dengan pendekatan berbasis peringkat, uji ini memungkinkan peneliti mengidentifikasi adanya perbedaan signifikan antar kelompok tanpa harus bergantung pada asumsi parametrik yang ketat [18].
Persamaan :

$$H = \frac{12}{N(N+1)} \sum_{i=1}^{k} \frac{R_i^2}{n_i} - 3\,(N+1) \qquad (1)$$

JURIKOM (Jurnal Riset Komputer), Vol. 13 No. 1, Februari 2026
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v13i1.9524
Hal 430-440
https://ejurnal.stmik-budidarma.ac.id/index.php/jurikom

Keterangan :
N = total data semua kelompok
R_i= jumlah ranking di kelompok ke-i
n_i= jumlah data di kelompok ke-i
k = Jumlah Kelompok

Metode Kruskal-Wallis dipilih karena data hasil pengujian tidak memenuhi asumsi distribusi normal dan melibatkan lebih dari dua kelompok independen. Sebagai uji statistik non-parametik, Kruskal-Wallis tidak mensyaratkan data terdistribusi normal dan digunakan untuk mengetahui perbedaan signifikan antar kelompok dengan berbentuk. Pengujian dilakukan sebanyak 50 kali menggunakan perangkat NodeMCU ESP8266. Tujuan dari analisis statistik ini Adalah untuk mengetahui apakah terdapat perbedaan signifikan dalam waktu eksekusi antara algoritma AES dan SPECK pada proses *enkripsi* maupun *dekripsi*. Perhitungan statistik dilakukan secara manual menggunakan rumus Kruskal-Wallis dengan bantuan *spreadsheet* untuk pengurutan rangking dan perhitungan nilai H.

2.8. Uji dan Analisis *Sniffing* sebagai Simulasi Ancaman Pasif

Metode uji *Sniffing* digunakan untuk mengevaluasi keamanan algoritma *enkripsi* dengan cara mengamati lalu lintas data yang dikirimkan melalui jaringan. Tujuan utama pengujian ini adalah memastikan bahwa data ter*enkripsi* tidak dapat dibaca atau dimanfaatkan oleh pihak yang tidak berwenang[19]. Pengujian dilakukan menggunakan perangkat lunak *packet sniffer* seperti Whireshark, yang berfungsi, menangkap paket data yang sedang ditransmisikan. Sistem diuji dengan mengirimkan data ter*enkripsi* menggunakan algoritma AES-128 dan SPECK-128/128, kemudian proses *sniffing* dilakukan untuk merekam paket yang lewat.

Data hasil tangkapan dianalisis untuk melihat apakah informasi UID masih dapat dikenali dan melihat apakah pola atau struktur *chipertext* memberikan indikasi terhadap isi asli. Selain itu, pengujian juga menilai ketahanan algoritma terhadap serangan pasif, yaitu serangan yang hanya mengandalkan pengamatan lalu lintas tanpa melakukan modifikasi data. Hasil uji *sniffing* menjadi indikator tambahan dalam menilai keamanan algoritma, khususnya pada sistem tertanam dan *IoT* yang rentan intersepsi data.

3. HASIL DAN PEMBAHASAN

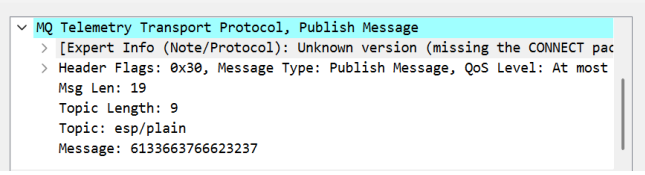
Pada tahap pengujian, performa *enkripsi* dan *dekripsi* diuji sebanyak 50 kali untuk setiap algoritma dengan parameter utama berupa waktu proses dalam satuan mili-detik. Selain itu, analisis penggunaan memori dilakukan dengan melihat ukuran *library* masing-masing algoritma, tanpa pengukuran *runtime*, karena perbedaan kebutuhan memori sudah dapat dilihat dari *footprint library* yang digunakan. Pengujian *sniffing* juga dilakukan sebanyak 50 kali untuk memastikan apakah data yang dikirim melalui protokol MQTT dapat disadap dalam bentuk *plaintext* jika ditemukan data yang tidak ter*enkripsi*, kembali ke tahap implementasi untuk diperbaiki. Seluruh hasil pengujian kemudian dibandingkan secara menyeluruh untuk menilai efisiensi dan tingkat keamanan masing-masing algoritma, khususnya dalam konteks penggunaan pada perangkat *IoT* jenis *low-end*.

Untuk memberikan ilustrasi data sampel, Gambar 6 dan Gambar 7 menampilkan UID kartu, data tanpa *enkripsi*, serta hasil *enkripsi* menggunakan algoritma AES-128 dan SPECK-128/128.

```
15:34:53.663 -> Akses diterima, kartu valid!
15:34:55.747 -> UID Kartu:  A3 F7 FB 27
```

Gambar 6. Data Sampel UID Kartu

Gambar 6 menunjukkan UID kartu RFID yang dibaca sebagai data *plaintext*. Data ini merupakan input awal sebelum dilakukan proses *enkripsi*.



Gambar 7. Hasil *Sniffing* sebelum di enkripsi

Seperti terlihat pada Gambar 7, playload yang ditangkap berupa string heksadesimal 6133663766623237, yang jika dikonversi ke ASCII menghasilkan a3f7fb27. Nilai ini sesuai dengan UID kartu asli A3 F7 FB 27, sehingga data dapat dibaca langsung tanpa *enkripsi*. Untuk memperjelas hasil *sniffing*, Tabel 1 menunjukkan konversi playload heksadesimal yang ditangkap Wireshark menjadi ASCII, sehingga sesuai dengan kartu RFID.

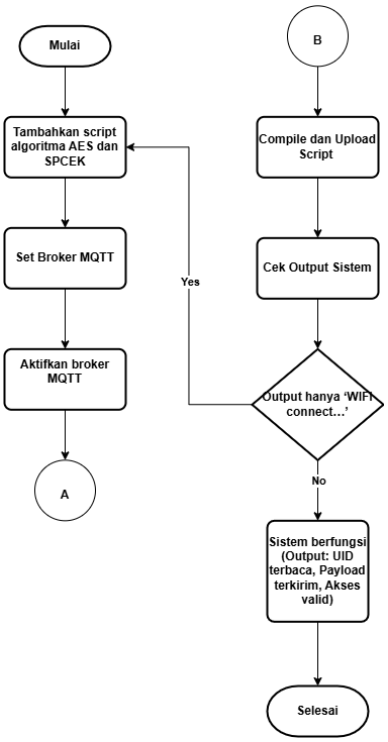
Tabel 1. Konversi *payload*(hex) ke ASCII

Playload Wireshark (Hex)	ASCII	UID Kartu asli
61	a	A3 F7 FB 27

JURIKOM (Jurnal Riset Komputer), Vol. 13 No. 1, Februari 2026
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v13i1.9524
Hal 430-440
https://ejurnal.stmik-budidarma.ac.id/index.php/jurikom

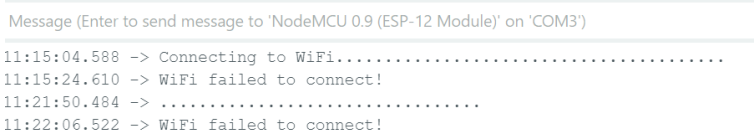
33	3
66	f
37	7
66	f
62	b
32	2
37	7

Dari Tabel 1 terlihat bahwa string heksadesimal 6133663766623237 yang muncul pada hasil *sniffing* sebenarnya merepresentasikan UID kartu A3 F7 FB 27 setelah dikonversi ke ASCII. Hal ini menegaskan bahwa data dapat dibaca langsung tanpa *enkripsi*, sehingga rawan disadap. Oleh karena itu, penerapan algoritma kriptografi seperti AES dan SPECK diperlukan untuk melindungi data. Untuk mengatasi permasalahan tersebut, Gambar 9 menyajika *flowchart* tahapan penerapan algoritma kriptografi AES-128 dan SPECK-128/128 pada sistem *Smart Door Lock* berbasis *IoT*. *Flowchart* ini menggambarkan alur proses mulai dari implementasi algoritma, konfigurasi broker MQTT, hingga verifikasi *output* sistem, sehingga dapat memastikan bahwa data komunikasi RFID tidak lagi dikirim dalam bentuk *plaintext* yang mudah disadap.

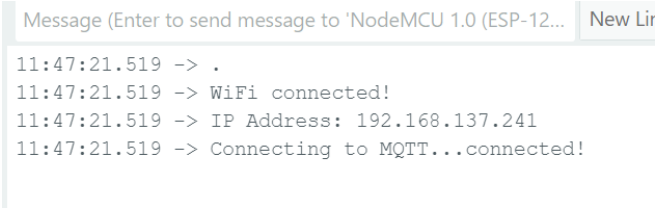


Gambar 8. *Flowchart* Penerapan Algoritma

Gambar 8 Memperlihatkan alur mulai dari penambahan *script* algoritma, konfigurasi broker MQTT, hingga pengujian *output* sistem. Ketika sistem berfungsi normal, *output* menunjukkan UID terbaca, *payload* terkirim, dan akses valid, sehingga penerapan kriptografi dapat dipastika berjalan dengan baik. Gambar 9 dan Gambar 10 menampilkan *output* sistem ketika perangkat mencoba terhubung ke jaringan WiFi dan broker MQTT.



Gambar 9.*Output* serial monitor saat perangkat gagal terhubung ke broker MQTT dan WiFi



Gambar 10. *Output* serial monitor saat perangkat berhasil terhubung ke broker MQTT dan WiFi

JURIKOM (Jurnal Riset Komputer), Vol. 13 No. 1, Februari 2026
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v13i1.9524
Hal 430-440
https://ejurnal.stmik-budidarma.ac.id/index.php/jurikom

Gambar 9 menunjukkan perangkat gagal terhubung ke WiFi sehingga proses komunikasi tidak berjalan. Gambar 10 menunjukkan *output* saat perangkat berhasil terhubung dan sistem dapat mengirimkan *payload* melalui MQTT.

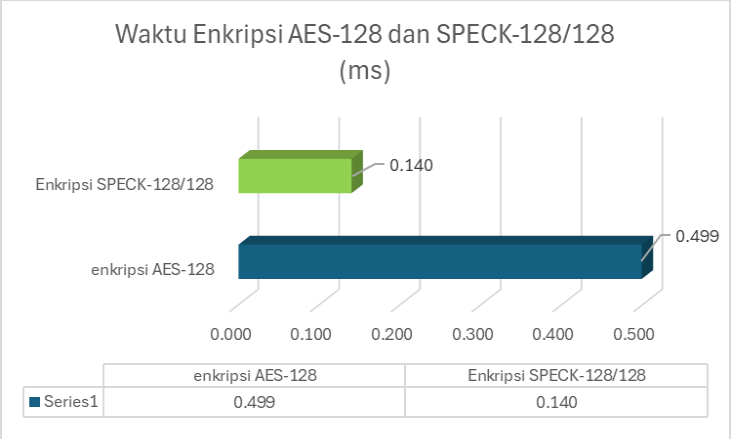
3.1 Hasil Uji Performa Algoritma

Untuk mengevaluasi perbedaan performa antara algoritma AES-128 dan SPECK-128/128, dilakukan pengujian terhadap dua parameter utama, yaitu waktu *enkripsi* dan *dekripsi*. Pengujian dilakukan sebanyak 50 kali untuk masing-masing algoritma, kemudian hasilnya dianalisis menggunakan metode statistik Kruskal-Wallis. Hasil pengujian disajikan pada Tabel 1.

Tabel 2. Uji Kruskal-Wallis Algoritma AES-128 dan SPECK-128

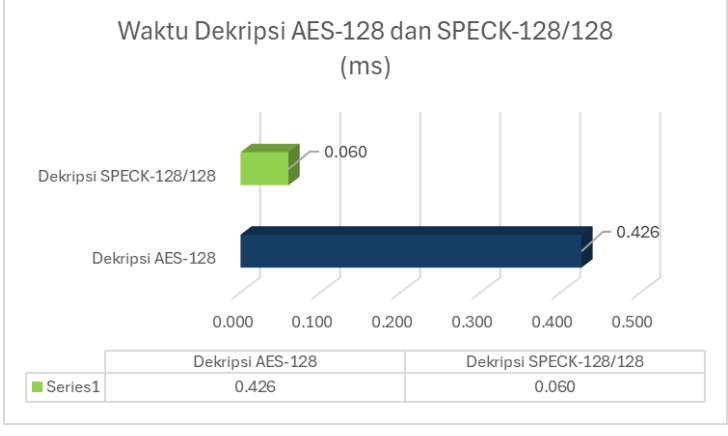
AES 128 dan SPECK-128/128	Statistik H	p-value	Kesimpulan
Enkripsi	42,48	$7{,}13\times10^{-11}$	Signifikan
Dekripsi	67,35	$1{,}37\times10^{-15}$	Signifikan

Hasil pengujian pada tabel 1 menunjukkan nilai H untuk *enkripsi* sebesar 42,48 dengan p-value $7{,}13\times10^{-11}$, sedangkan untuk *dekripsi* nilai H sebesar 67,35 dengan p-value $1{,}37\times10^{-15}$. Kedua p-value jauh di bawah batas signifikansi 0,05, sehingga terdapat perbedaan yang signifikan antara kedua algoritma. Menurut teori kriptografi ringan, perangkat *IoT* dengan keterbatasan sumber daya seperti ESP8266 membutuhkan algoritma yang hemat memori dan cepat dieksekusi agar sistem tetap efisien[20]. Hal ini sejalan dengan teori *trade-off* dalam kriptografi ringan, di mana algoritma yang lebih ringan seperti SPECK menawarkan efisiensi lebih tinggi namun dengan tingkat keamanan yang relatif lebih rendah dibanding AES[17]. SPECK128/128 terbukti lebih cepat dibandingkan AES-128, walaupun Tingkat keamanannya sedikit lebih rendah. Hasil analisis ini konsisten dengan penelitian sebelumnya yang menunjukkan bahwa AES membutuhkan waktu proses lebih lama dibandingkan SPECK pada perangkat dengan sumber daya terbatas. Dengan demikian, AES lebih sesuai untuk sistem dengan kapasitas komputasi tinggi, sedangkan SPECK lebih unggul dalam konteks *IoT* karena mampu memberikan performa waktu yang lebih cepat dengan beban komputasi rendah[17]. Untuk memperjelas distribusi data, rata-rata waktu eksekusi *real-time* ditampilkan pada Gambar 11 dan Gambar 12.



Gambar 11. Waktu *Enkripsi* AES-128 dan SPECK-128/128

Gambar 11 menunjukkan perbandingan waktu enkripsi antara algoritma AES-128 dan SPECK-128/128. Terlihat bahwa SPECK lebih cepat (0,140 ms) dibandingkan AES (0,499 ms), sehingga lebih efisien untuk perangkat *IoT* dengan keterbatasan sumber daya.



Gambar 12. Waktu *Dekripsi* AES-128 dan SPECK-128/128

JURIKOM (Jurnal Riset Komputer), Vol. 13 No. 1, Februari 2026
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v13i1.9524
Hal 430-440
https://ejurnal.stmik-budidarma.ac.id/index.php/jurikom

Gambar 12 memperlihatkan perbandingan waktu *dekripsi* antara algoritma AES-128 dan SPECK-128/128. Hasilnya menunjukkan bahwa SPECK jauh lebih cepat (0,060 ms) dibandingkan AES (0,426 ms).

3.2 Analisis Penggunaan Memori *Library*

Penggunaan memori dibandingkan berdasarkan ukuran *library* masing-masing algoritma. Analisis ini bertujuan untuk menilai efisiensi algoritma dalam konteks perangkat *IoT* dengan sumber daya terbatas.

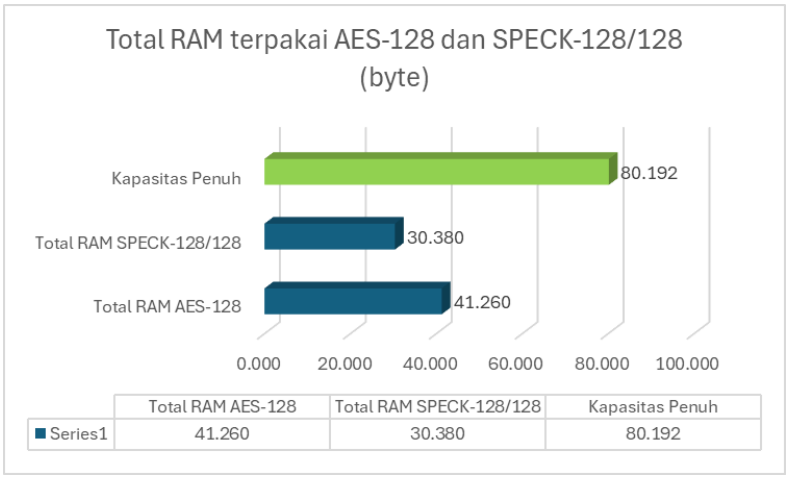
Tabel 3. Detail Pemakaian Memori Algoritma AES-128

Jenis Memori	Segmen	Ukuran (*Byte*)	Keterangan
RAM	Data	1.776	*Variable* yang di inisialisasi
RAM	Rodata	6.860	Konstanta
RAM	BSS	32.624	*Variable* yang di nol-kan
Total RAM	**-**	**41.260**	**51% dari kapasitas (80.192 *byte*)**
FLASH	IROM	434.004	Kode disimpan di FLASH
Total	**-**	**434.004**	**41% dari kapasitas (1.048.576 *byte*)**

Tabel 4. Detail Pemakaian Memori Algoritma SPECK-128/128

Jenis Memori	Segmen	Ukuran (*Byte*)	Keterangan
RAM	Data	1.540	*Variable* yang di inisialisasi
RAM	Rodata	1.624	Konstanta
RAM	BSS	27.216	*Variable* yang di nol-kan
Total RAM	**-**	**30.380**	**37% dari kapasitas (80.192 *byte*)**
FLASH	IROM	260.452	Kode disimpan di FLASH
Total	**-**	**260.452**	**24% dari kapasitas (1.048.576 *byte*)**

Analisis detail segmen memori menunjukkan bahwa AES 128 membutuhkan RAM lebih besar (41.260 *byte*) dibandingkan SPECK 128 (30.380 *byte*). Hal ini terutama disebabkan oleh ukuran segmen BSS yang lebih besar pada AES. Dari sisi FLASH, AES juga lebih berat dengan 434.004 *byte* dibandingkan SPECK yang hanya 260.452 *byte*. Dengan demikian, SPECK lebih efisien dalam penggunaan memori, sedangkan AES lebih kompleks dan membutuhkan ruang lebih besar untuk *variable* dan kode program. Gambar 13 memperlihatkan perbandingan penggunaan RAM antara algoritma AES-128 dan SPECK-128/128 pada perangkat *IoT* dengan kapasitas penuh 80.192 *byte*.



Gambar 13. Diagram Total RAM terpakai antara algoritma AES-128 dan SPECK-128/128

Dari Gambar 13 terlihat bahwa AES-128 menggunakan 41.260 *byte* (51% dari kapasitas), sedangkan SPECK-128/128 hanya 30.380 *byte* (37% dari kapasitas). Hal ini menunukan bahwa SPECK lebih efisien dalam penggunaan memori dibandingkan AES.

3.3 Analisis Hasil *Sniffing*

Hasil *sniffing* dari 50 kali percobaan dievaluasi untuk melihat apakah data yang dikirim melalui MQTT dapat disadap atau tetap ter*enkripsi*. Ini menjadi indikator efektivitas *enkripsi* dalam menjaga kerahasiaan data.

Tabel 5. Persentase data berhasil di *enkripsi*

Algoritma	Total Percobaan	Paket Ter*enkripsi*	Paket *Plaintext*	Persentase aman
AES-128	50	50	0	100%
SPECK128/128	50	50	0	100%

JURIKOM (Jurnal Riset Komputer), Vol. 13 No. 1, Februari 2026
e-ISSN 2715-7393 (Media Online), p-ISSN 2407-389X (Media Cetak)
DOI 10.30865/jurikom.v13i1.9524
Hal 430-440
https://ejurnal.stmik-budidarma.ac.id/index.php/jurikom

Pengujian efektivitas *enkripsi* dilakukan sebanyak 50 kali untuk masing-masing algoritma. Hasilnya menunjukkan bahwa baik AES-128 maupun SPECK-128/128 berhasil meng*enkripsi* seluruh paket yang diuji tanpa ada satu pun paket yang tertinggal dalam bentuk *plaintext*. Dengan demikian, kedua algoritma mencapai tingkat keamanan 100% dalam proses *enkripsi*, yang ditunjukkan oleh persentase aman sebesar 100% pada masing-masing metode. Temuan ini memperkuat bahwa baik AES maupun SPECK mampu menjalankan fungsi *enkripsi* secara utuh dan efektif dalam konteks pengujian yang dilakukan.

Cuplikan hasil *sniffing* yang ditampilkan merupakan data yang dikirim melalui protokol MQTT dengan tipe pesan "*Publish Message*". Topik yang digunakan adalah SPECK/*encrypted*, yang secara eksplisit menunjukkan bahwa *payload* merupakan hasil *enkripsi* menggunakan algoritma SPECK-128/128. Pada bagian Message, terlihat deretan karakter acak dalam format hexadecimal, seperti 3365323286330434..., yang tidak mengandung informasi *plaintext*.

Karakteristik ini menunjukkan bahwa proses *enkripsi* berjalan dengan baik dan *payload* telah terlindungi sepenuhnya. Karena implementasi menggunakan mode *Cipher Block Chaining (CBC)*, setiap blok *ciphertext* bergantung pada blok sebelumnya, sehingga hasil *enkripsi* tampak acak dan tidak mengungkap struktur data asli. Dengan demikian, algoritma SPECK-128/128 yang digunakan dalam pengujian berhasil menjaga kerahasiaan data selama transmisi dan tidak rentan terhadap pengintaian melalui *sniffing*.

Selain perbedaan pola visual, hasil *sniffing* juga menunjukkan bahwa panjang *ciphertext* yang dihasilkan oleh algoritma AES-128 dan SPECK-128/128 tidak sama. Pada pengujian, *payload* hasil *enkripsi* AES-128 memiliki panjang sekitar 90 *byte*, sedangkan *payload* hasil *enkripsi* SPECK-128/128 lebih pendek, yaitu sekitar 82 *byte*. Perbedaan ini mencerminkan kompleksitas internal masing-masing algoritma. AES, dengan struktur blok dan operasi yang lebih kompleks, menghasilkan *ciphertext* yang relatif lebih panjang dan padat. Sebaliknya, SPECK yang dirancang untuk efisiensi pada perangkat dengan keterbatasan sumber daya menghasilkan *ciphertext* lebih ringkas, meskipun tetap acak dan tidak dapat diinterpretasikan sebagai *plaintext*.

Dengan demikian, perbedaan panjang *ciphertext* ini semakin menegaskan karakteristik kedua algoritma: AES lebih berat namun konsisten, sementara SPECK lebih ringan dan efisien. Keduanya tetap aman karena menggunakan mode CBC, yang menyamarkan pola data asli sehingga hasil *sniffing* hanya menampilkan deretan karakter acak.

Dengan demikian, perbedaan panjang *ciphertext* ini semakin menegaskan karakteristik kedua algoritma: AES lebih berat namun konsisten, sementara SPECK lebih ringan dan efisien. Keduanya tetap aman karena menggunakan mode CBC, yang menyamarkan pola data asli sehingga hasil *sniffing* hanya menampilkan deretan karakter acak

3.4 Pembahasan

Hasil Penelitian menunjukkan bahwa algoritma SPECK-128/128 lebih efisien dibandingkan AES-128, baik dari sisi waktu eksekusi maupun penggunaan memori. SPECK mampu melakukan enkripsi dan *dekripsi* lebih cepat serta membutuhkan RAM lebih kecil, sehingga lebih sesuai untuk perangkat *IoT* dengan keterbatasan sumber daya. Sebaliknya, AES-128 membutuhkan waktu dan memori lebih besar karena proses algoritmanya lebih panjang dan kompleks. Sepert yang terlihat pada Gambar 4 , AES melibatkan tahapan berulang seperti *SubBytes*, *ShiftRows*, *MixColumns* dan *AddRoundKey* . Sementara itu, pada Gambar 5, SPECK menggunakan struktur sederhana berbasis operasi ARX (Add – Rotate – XOR) yang membuatnya lebih ringan dan cepat.

Dari sisi keamanan, kedua algoritma terbukti sama-sama efektif karena seluruh paket terenkripsi dengan aman tanpa ada data plaintext yang terbaca. Panjang *chipertext* yang lebih besar mencerminkan kompleksitas internalnya, sementara SPECK tetap mampu menjaga kerajasiaan dengan struktur yang lebih sederhana.

Secara keseluruhan, temuan ini menegaskan adanya *trade-off* antara efisiensi dan kompleksitas, AES lebih cocok untuk sistem dengan kapasitas komputasi tingi, sedangkan SPECK lebih relevan untuk perangkat *IoT low-end.*. Hasil ini konsisten dengan kajian teoretis yang menekankan pentingnya kriptografi ringan untuk mengatasi keterbatasan memori dan komputasi pada mikrokontroler *IoT* [17].

4. KESIMPULAN

Penelitian ini menunjukkan bahwa baik algoritma AES-128 maupun SPECK-128/128 mampu menjaga kerahasiaan data pada sistem *Smart Door Lock* berbasis *IoT* dengan protokol MQTT. Hasil pengujian performa memperlihatkan bahwa AES memiliki waktu eksekusi yang lebih lama namun stabil, sedangkan SPECK lebih cepat tetapi kurang konsisten. Dari sisi penggunaan memori, AES membutuhkan RAM dan FLASH lebih besar dibandingkan SPECK, sehingga SPECK lebih efisien untuk perangkat dengan keterbatasan sumber daya. Analisis *sniffing* memperkuat temuan ini, di mana seluruh paket ter*enkripsi* dan tidak ada data *plaintext* yang terbaca, sehingga tingkat keamanan mencapai 100% pada kedua algoritma. Secara keseluruhan, AES lebih sesuai untuk sistem dengan kapasitas komputasi tinggi karena stabilitas dan tingkat keamanan yang lebih mapan, sementara SPECK lebih cocok untuk perangkat *IoT low-end.* karena efisiensi memori dan kecepatan eksekusi yang lebih tinggi. Perbedaan panjang *ciphertext* antara AES (±90 *byte*) dan SPECK (±82 *byte*) juga menegaskan karakteristik internal masing-masing algoritma. Dengan demikian, integrasi algoritma kriptografi AES-128 dan SPECK-128/128 pada sistem *Smart Door Lock* berbasis *IoT* dengan protokol MQTT memberikan kontribusi nyata dalam pengembangan solusi keamanan rumah pintar yang lebih aman, efisien, dan ekonomis dibandingkan penelitian sebelumnya.

REFERENCES

[1] F. P. E. Putra, S. M. Dewi, A. Hamzah, and U. Madura, "Privasi dan Keamanan Penerapan IoT Dalam Kehidupan Sehari-Hari : Tantangan dan Implikasi," vol. 5, no. 2, 2023.

[2] D. A. S. Ilhami, "Data Privasi dan Keamanan Siber pada Smart-City: Tinjauan Literatur," *J. Sains Nalar Dan Apl. Teknol. Inf.*, vol. 2, no. 1, Sep. 2022, doi: 10.20885/snati.v2i1.19.

[3] M. Mut'Mainna, A. M. Rivai, A. Febisatria, T. Sarnawiyah, dan M. R. Setiawan, "Internet of Things (IoT) sebagai Pilar Keamanan Data pada Sistem Distribusi di Indonesia," *Jurnal Ilmiah Ekonomi dan Manajemen*, vol. 3, no. 12, pp. 332–341, Des. 2025, doi: 10.61722/jiem.v3i12.7507

[4] M. Octaria and M. I. P. Nasution, "Peluang dan Tantangan Penerapan Internet of Things (IoT) dalam Sistem Informasi Manajemen".

[5] H. P. Pratama, A. S. Prihatmanto, and A. Sukoco, "Implementation Messaging Broker Middleware for Architecture of Public Trans*port*ation Monitoring System," in *2020 6th International Conference on Interactive Digital Media (ICIDM)*, Bandung, Indonesia: IEEE, Dec. 2020, pp. 1–5. doi: 10.1109/ICIDM51048.2020.9339673.

[6] M. T. Mbejo dan I. Sufian, "Analisis Tantangan Keamanan Jaringan IoT dan Strategi Mitigasinya," *JR – Jurnal Responsive*, vol. 9, no. 1, Jun. 2025, doi: 10.36352/jr.v9i01.1178.

[7] H. Ulfa, A. I. Basuki, G. M. Suranegara, and A. Fauzi, "DDoS Protection System for SDN Network Based on Multi Controller and Load Balancer," *SISTEMASI*, vol. 13, no. 2, p. 555, Mar. 2024, doi: 10.32520/stmsi.v13i2.3802.

[8] A. P. Paramaputra, G. M. Suranegara, and E. Setyowati, "MITIGATION OF MULTI TARGET DENIAL OF SERVICE (DOS) ATTACKS USING WAZUH ACTIVE RESPONSE," *J. Comput. Netw. Archit. High Perform. Comput.*, vol. 7, no. 2, pp. 483–493, Apr. 2025, doi: 10.47709/cnahpc.v7i2.5755.

[9] M. Sari, H. D. Purnomo, and I. Sembiring, "Review : Algoritma Kriptografi Sistem Keamanan SMS di Android," *J. Inf. Technol.*, vol. 2, no. 1, pp. 11–15, Mar. 2022, doi: 10.46229/jifotech.v2i1.292.

[10] R. K. Endrayanto, A. Muttaqin, and R. A. Setyawan, "Advanced Encryption Standard (AES) pada Modul Internet of Things (IoT)".

[11] M. Apriannur, D. T. Nugrahadi, A. Farmadi, and M. I. Mazdadi, "ANALISIS IMPLEMENTASI ALGORITMA ENKRIPSI AES-128 BIT DATA RFID PADA JARINGAN 802.11n UNTUK MONITORING JARAK JAUH BERBASIS MQTT," vol. 10, 2023.

[12] Muhammad Naufal Rabbani and D Farah Afianti, "Analisis Performansi Algoritma Small AES Menggunakan Arduino UNO, Studi Kasus : Pemantauan Suhu," *Log. J. Penelit. Inform.*, vol. 2, no. 2, Jan. 2025, doi: 10.25124/logic.v2i2.8814.

[13] National Institute of Standards and Technology (US), "Advanced Encryption Standard (AES)," National Institute of Standards and Technology (U.S.), Washington, D.C., NIST FIPS 197-upd1, May 2023. doi: 10.6028/NIST.FIPS.197-upd1.

[14] J. Chakraborty, S. Bhattacharya, A. Mukhopadhyay, dan S. Das, Eds., *The Speck Cipher*. Cham, Switzerland: Springer, 2019, doi: 10.1007/978-3-030-10591-4.

[15] I. Rozlomii, A. Yarmilko, S. Naumenko, and P. Mykhailovskyi, "Hardware encryptors and cryptographic libraries for optimizing security in IoT ⋆".

[16] H. J. Hamiza and A. Fanfakh, "Parallel lightweight Block Cipher algorithm for Multicore CPUs," *Baghdad Sci. J.*, p. 3289, Oct. 2024, doi: 10.21123/bsj.2024.9052.

[17] R. W. Febrianto and A. Zulianto, "Kriptografi Ringan dengan Menggunakan Algoritma di Internet Of Things (IoT)," vol. 4, no. 3, 2024.

[18] S. Lukmaini, M. Faisal, dan S. A. W. Dinata, "Analisis Tingkat Hunian Kamar (TPK) di Provinsi Kalimantan Timur dengan Metode Kruskal Wallis, Uji Friedman dan Uji Spearman," *Journal of Innovation Research and Knowledge*, vol. 3, no. 6, Nov. 2023, pp. 1239–1248.

[19] Y. T. Kantari, I. M. Widiarta, D. Wiryatari, S. Esabella, and Y. W, "Analisis Keamanan Jaringan Modem Zte Terhadap Serangan Packet Sniffing Menggunakan Metode Penetration Testing," *Digit. Transform. Technol.*, vol. 5, no. 1, pp. 41–51, Apr. 2025, doi: 10.47709/digitech.v5i1.5552.

[20] I. Radhakrishnan, S. Jadon, and P. B. Honnavalli, "Efficiency and Security Evaluation of Lightweight Cryptographic Algorithms for Resource-Constrained IoT Devices," *Sensors*, vol. 24, no. 12, p. 4008, Jun. 2024, doi: 10.3390/s24124008.