Analisis Kinerja Implementasi ASCON-128 vs AES-128-GCM pada Sistem Kunci Sepeda IoT

Ahmad Naufal Ramadan - 13522005
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Jl. Ganesha 10 Bandung 40132, Indonesia
13522005@std.stei.itb.ac.id, naufalahmad022@gmail.com

***Abstrak*—Perkembangan pesat Internet of Things (IoT) mendorong kebutuhan akan algoritma kriptografi yang efisien untuk perangkat dengan sumber daya terbatas. National Institute of Standards and Technology (NIST) baru-baru ini menetapkan ASCON sebagai standar Lightweight Cryptography (LWC). Penelitian ini menyajikan analisis komparatif kinerja antara ASCON-128 dan standar industri saat ini, AES-128-GCM, yang diterapkan pada simulasi sistem kunci sepeda pintar (Smart Bicycle Lock). Implementasi dilakukan menggunakan bahasa Python untuk mengevaluasi metrik latensi CPU, penggunaan memori, dan throughput. Hasil pengujian menunjukkan bahwa meskipun AES-128-GCM memiliki keunggulan kecepatan eksekusi pada lingkungan dengan akselerasi perangkat keras, ASCON-128 terbukti jauh lebih efisien dalam penggunaan memori, dengan konsumsi RAM sekitar 70% lebih rendah (1.40 KB) dibandingkan AES-128-GCM (4.88 KB). Temuan ini menegaskan bahwa ASCON-128 adalah kandidat ideal untuk perangkat IoT dengan memori sangat terbatas (*highly constrained*), sementara AES-128-GCM tetap menjadi pilihan utama untuk sistem yang memiliki dukungan instruksi kriptografi khusus.**

***Kata Kunci*—Kriptografi Ringan, ASCON, AES-GCM, Keamanan IoT, Benchmarking**

I. Pendahuluan

Internet of Things (IoT) telah bertransformasi menjadi elemen fundamental dalam infrastruktur teknologi modern, menghubungkan miliaran perangkat mulai dari peralatan rumah tangga hingga sensor industri. Namun, proliferasi perangkat ini menghadirkan tantangan keamanan yang signifikan. Banyak perangkat IoT, seperti kunci pintar (*smart locks*), sensor medis, dan aktuator industri, beroperasi dengan sumber daya yang sangat terbatas baik dari segi daya komputasi, memori, maupun energi baterai [1]. Standar enkripsi saat ini, Advanced Encryption Standard (AES), meskipun sangat aman dan efisien pada perangkat keras modern (desktop/server), seringkali membebani perangkat mikrokontroler murah yang tidak memiliki akselerasi perangkat keras khusus (AES-NI). Implementasi perangkat lunak AES dapat menghabiskan siklus CPU dan memori yang berlebihan, yang berdampak langsung pada latensi sistem dan umur baterai.

Menanggapi tantangan ini, National Institute of Standards and Technology (NIST) menyelenggarakan kompetisi untuk mencari standar kriptografi ringan (Lightweight Cryptography/LWC) baru. Puncaknya pada Februari 2023, NIST mengumumkan keluarga algoritma **ASCON** sebagai pemenang dan standar baru untuk perlindungan data pada perangkat terbatas. Makalah ini mengusulkan evaluasi kinerja ASCON-128 dibandingkan dengan standar AES-128-GCM dalam konteks aplikasi nyata, yaitu sistem *Smart Bicycle Lock*. Kasus penggunaan ini dipilih karena mewakili skenario IoT tipikal: perangkat bertenaga baterai yang memerlukan autentikasi aman, latensi rendah untuk pengalaman pengguna, dan ketahanan terhadap serangan siber.

Penelitian ini mencakup pengembangan simulasi berbasis Python untuk memodelkan interaksi kunci sepeda pintar dan mengukur metrik kinerja kunci. Fokus utama evaluasi adalah perbandingan latensi waktu enkripsi dan dekripsi, efisiensi penggunaan memori (RAM), serta kelayakan implementasi pada sistem yang beroperasi secara *real-time*. Kami membatasi pengujian pada varian ASCON-128 dan AES-128-GCM menggunakan protokol Authenticated Encryption with Associated Data (AEAD), untuk memberikan wawasan teknis yang relevan bagi perancang sistem IoT berdaya rendah.

Sistematika penulisan makalah ini disusun sebagai berikut: Bagian II memaparkan landasan teori terkait konsep Kriptografi Ringan, spesifikasi ASCON, dan mode operasi GCM. Bagian III menjelaskan metodologi penelitian, termasuk desain sistem simulasi dan skenario benchmarking. Bagian IV menyajikan analisis hasil eksperimen secara kuantitatif, membandingkan latensi, memori, dan throughput. Terakhir, Bagian V memberikan kesimpulan dari temuan penelitian ini serta saran untuk pengembangan masa depan.

II. Landasan Teori

A. Kriptografi

Kriptografi adalah ilmu dan seni untuk menjaga keamanan pesan [2]. Dalam skenario komunikasi klasik yang sering digambarkan dengan pengirim (Alice) dan penerima (Bob), kriptografi bertujuan untuk melindungi informasi yang dikirimkan melalui saluran publik yang tidak aman dari pihak ketiga (Eve).

Secara umum, tujuan utama kriptografi meliputi:

- **Kerahasiaan (*Confidentiality*)**: Menjamin bahwa informasi hanya dapat diakses oleh pihak yang berwenang.
- **Integritas Data (*Data Integrity*)**: Menjamin bahwa data tidak diubah atau dimanipulasi selama transmisi.

- **Autentikasi (*Authentication*)**: Memastikan identitas pengirim atau sumber data.
- **Nir-sangkal (*Non-repudiation*)**: Mencegah pengirim menyangkal bahwa mereka telah mengirimkan pesan tersebut.

B. Kriptografi Ringan (Lightweight Cryptography)

Kriptografi ringan (*Lightweight Cryptography* atau LWC) adalah cabang riset kriptografi yang dirancang khusus untuk perangkat dengan sumber daya sangat terbatas (*constrained devices*) [1]. Berbeda dengan kriptografi konvensional yang dioptimalkan untuk kecepatan tinggi pada prosesor desktop atau server, LWC berfokus pada efisiensi implementasi.

Perangkat target untuk LWC meliputi:

- *Internet of Things* (IoT) *devices* seperti sensor dan node jaringan kecil.
- *Smart card* dan RFID *tags*.
- *Medical implants* (implan medis).
- Mikrokontroler *embedded* 8-bit atau 16-bit.

Keterbatasan utama yang dihadapi perangkat-perangkat ini adalah memori yang kecil (RAM dan ROM terbatas), daya komputasi rendah, dan sumber daya energi yang terbatas (beroperasi dengan baterai kecil atau *energy harvesting*). Oleh karena itu, algoritma standar seperti AES atau RSA seringkali dianggap terlalu ”berat” atau boros energi untuk lingkungan ini.

C. Authenticated Encryption with Associated Data (AEAD)

Authenticated Encryption with Associated Data (AEAD) adalah kelas algoritma kriptografi yang menyediakan kerahasiaan, integritas, dan autentikasi secara bersamaan [3]. Pendekatan ini mengatasi kelemahan dari penggabungan manual antara enkripsi dan *Message Authentication Code* (MAC) yang rentan terhadap kesalahan implementasi.

Berdasarkan RFC 9771, algoritma AEAD menerima empat masukan:

- **Secret Key** (K): Kunci rahasia yang dibagikan antar pihak.
- **Nonce** (N): Nilai unik (*Number used once*) untuk mencegah serangan ulang (*replay attack*).
- **Associated Data** (AD): Data publik yang perlu diautentikasi integritasnya namun tidak dienkripsi, seperti *header* paket jaringan.
- **Plaintext** (P): Data rahasia yang akan dienkripsi dan diautentikasi.

Keluaran dari operasi AEAD adalah *Ciphertext* (C) dan *Authentication Tag* (T). Proses dekripsi akan memverifikasi tag T terlebih dahulu; jika verifikasi gagal, maka ciphertext tidak akan didekripsi dan dianggap tidak sah.

D. Advanced Encryption Standard (AES)

AES (*Advanced Encryption Standard*) adalah standar enkripsi blok simetris yang ditetapkan oleh NIST dalam publikasi FIPS 197 [4]. AES menggantikan DES sebagai standar global.

AES beroperasi pada blok data berukuran tetap 128-bit. Panjang kunci yang didukung adalah 128, 192, dan 256 bit, yang menentukan jumlah putaran (*rounds*) transformasi: 10 putaran untuk kunci 128-bit, 12 untuk 192-bit, dan 14 untuk 256-bit. Setiap putaran AES terdiri dari empat transformasi lapisan:

1) **SubBytes**: Substitusi non-linear menggunakan S-Box untuk memberikan sifat *confusion*.
2) **ShiftRows**: Permutasi baris untuk difusi.
3) **MixColumns**: Operasi pencampuran kolom linear untuk difusi lebih lanjut (tidak dilakukan pada putaran terakhir).
4) **AddRoundKey**: Operasi XOR antara state dengan *round key*.

E. AES-128-GCM

Galois/Counter Mode (GCM) adalah mode operasi untuk blok cipher yang direkomendasikan oleh NIST dalam SP 800-38D untuk menyediakan layanan *authenticated encryption* [5].

AES-GCM menggabungkan:

- **Counter Mode (CTR)**: Untuk enkripsi, yang mengubah blok cipher menjadi stream cipher, memungkinkan paralelisasi penuh.
- **GMAC**: Untuk autentikasi, yang menggunakan fungsi hash universal GHASH. GHASH beroperasi pada medan Galois $GF(2^{128})$.

Meskipun GCM sangat cepat pada prosesor yang memiliki instruksi *carry-less multiplication* (PCLMULQDQ) dan AES-NI, operasi perkalian pada medan Galois $GF(2^{128})$ sangat membebani mikrokontroler sederhana yang tidak memiliki dukungan perangkat keras tersebut, mengakibatkan latensi tinggi dan kode program yang besar untuk tabel prekomputasi.

F. Algoritma ASCON

ASCON adalah algoritma AEAD yang dirancang khusus untuk memenuhi kebutuhan kriptografi ringan dan telah ditetapkan sebagai pemenang kompetisi NIST Lightweight Cryptography. Spesifikasi ASCON v1.2 [6] mendefinisikan ASCON menggunakan konstruksi *Sponge*.

ASCON-128, varian utama yang direkomendasikan, memiliki karakteristik:

- **State Size**: 320 bit, dibagi menjadi rate $r = 64$ bit dan capacity $c = 256$ bit.
- **Key & Tag Size**: 128 bit.
- **Rounds**: $a = 12$ putaran untuk inisialisasi/finalisasi, dan $b = 6$ putaran untuk memproses data.

Operasi ASCON terdiri dari empat fase:

1) **Initialization**: Memuat kunci dan nonce ke dalam state, diikuti permutasi p^a.
2) **Associated Data Processing**: Menyerap AD blok demi blok.
3) **Plaintext Processing**: Menyerap plaintext dan memeras ciphertext (mode *duplex*).
4) **Finalization**: Menghasilkan tag autentikasi.

Desain ASCON hanya menggunakan operasi bitwise sederhana (AND, NOT, XOR, ROT), membuatnya sangat efisien dan kecil secara ukuran kode pada mikrokontroler apa pun, sekaligus tahan terhadap serangan *side-channel*.

G. Metrik Evaluasi Kinerja

Evaluasi kinerja pada perangkat terbatas difokuskan pada tiga metrik utama:

- **Latensi CPU**: Waktu siklus atau durasi (dalam detik/mikrodetik) yang dibutuhkan untuk melakukan operasi enkripsi dan dekripsi lengkap.
- **Penggunaan Memori (RAM)**: Jumlah memori tumpukan (*stack*) maksimum yang digunakan fungsi kriptografi. Hal ini krusial karena mikrokontroler seringkali hanya memiliki RAM dalam orde kilobyte.
- **Ukuran Kode (ROM/Flash)**: Besarnya ruang penyimpanan program yang dibutuhkan untuk menyimpan implementasi algoritma.

III. Metodologi

Sistem kunci sepeda pintar disimulasikan menggunakan bahasa pemrograman Python 3.10. Implementasi terdiri dari modul utama `BicycleLockSystem` dan wrapper kriptografi untuk ASCON dan AES.

A. Arsitektur Perangkat Lunak

Kode program diorganisir ke dalam modul-modul berikut:

- `src/bicycle_lock_terminal.py`[1]: Antarmuka terminal utama yang menangani registrasi sepeda dan simulasi proses `lock/unlock`.
- `src/crypto_engine/ascon_wrapper.py`[1]: Implementasi kelas `AsconLock` yang membungkus pustaka `ascon` native Python.
- `src/crypto_engine/aes_wrapper.py`[1]: Implementasi kelas `AESLock` menggunakan pustaka `pycryptodome` dengan mode GCM.

B. Implementasi Kriptografi

Bagian ini mendetailkan implementasi wrapper untuk kedua algoritma yang digunakan dalam simulasi.

1) Implementasi ASCON-128: ASCON-128 diimplementasikan menggunakan pustaka `ascon` native Python (Gambar 1). Wrapper ini menangani pembuatan *nonce* unik (128-bit) untuk setiap operasi enkripsi guna mencegah *replay attack*. *Associated Data* (AD) disertakan untuk menjamin integritas konteks perintah.

2) Implementasi AES-128-GCM: Sebagai pembanding standar industri, AES-128-GCM diimplementasikan menggunakan pustaka `pycryptodome`. Berbeda dengan ASCON, AES-GCM menggunakan *nonce* standar 96-bit yang diperlihatkan pada Gambar 2.

[1]Kode sumber lengkap (simulasi dan kriptografi) tersedia di: https://github.com/SandWithCheese/makalah-kripto

```python
class AsconLock(BaseCipher):
    def encrypt_command(self, plaintext,
        associated_data):
        # Generate unique 128-bit nonce
        nonce = os.urandom(16)

        # ASCON-128 encryption
        ciphertext = ascon_encrypt(
            key=self.key,
            nonce=nonce,
            associateddata=associated_data,
            plaintext=plaintext,
            variant="Ascon-128"
        )
        return nonce, ciphertext
```

Gambar 1. Potongan kode implementasi enkripsi ASCON-128.

```python
class AESLock(BaseCipher):
    def encrypt_command(self, plaintext,
        associated_data):
        # Generate unique 96-bit nonce (Standard GCM
            )
        nonce = os.urandom(12)

        cipher = AES.new(self.key, AES.MODE_GCM,
            nonce=nonce)
        cipher.update(associated_data)

        # Encrypt and generate tag
        ciphertext, tag = cipher.encrypt_and_digest(
            plaintext)
        return nonce, ciphertext + tag
```

Gambar 2. Potongan kode implementasi enkripsi AES-128-GCM.

C. Protokol Komunikasi Aman

Untuk memastikan keamanan perintah ”Unlock”, setiap pesan dienkripsi menggunakan skema AEAD.

1) **Pembangkitan Kunci**: Saat registrasi, kunci acak 128-bit dibangkitkan menggunakan `os.urandom(16)`.
2) **Struktur Pesan**: Format pesan adalah `UNLOCK:<bike_id>:<timestamp>`.
3) **Nonce Unik**: Untuk setiap operasi enkripsi, *nonce* baru dibangkitkan.
4) **Associated Data**: ID sepeda disertakan sebagai AD yang tidak dienkripsi namun diautentikasi.

D. Simulasi Berbasis Terminal

Untuk memvalidasi fungsionalitas sistem secara menyeluruh, dikembangkan aplikasi simulasi berbasis terminal (*Command Line Interface*). Simulasi ini memodelkan interaksi nyata antara pengguna dan sistem kunci pintar.

1) Alur Kerja Simulasi: Aplikasi terminal menyediakan antarmuka interaktif yang mencakup fitur-fitur berikut:

- **Registrasi Perangkat**: Pengguna dapat mendaftarkan sepeda dengan ID unik. Sistem akan membangkitkan dan menyimpan kunci enkripsi khusus untuk ID tersebut.
- **Simulasi Unlock**: Fitur ini mensimulasikan pembangkitan token akses oleh aplikasi mobile. Token berisi per-

intah waktu-nyata (real-time) yang diamankan dengan algoritma terpilih (ASCON atau AES).

- **Verifikasi Token**: Mensimulasikan logika pada mikrokontroler kunci (Lock MCU). Sistem memvalidasi token, mendekripsi pesan, dan memeriksa kesesuaian waktu serta integritas data.
- **Uji Manipulasi Data**: Simulasi secara otomatis menyertakan demonstrasi keamanan dengan memodifikasi token yang valid (tampering). Sistem harus mampu mendeteksi perubahan ini dan menolak akses, mendemonstrasikan fitur integritas AEAD.

Implementasi logika pembuatan token dapat dilihat pada Gambar 3.

```
def generate_unlock_token(self, bike_id):
    # Create secure command structure
    timestamp = datetime.now().isoformat()
    command = f"UNLOCK:{bike_id}:{timestamp}"

    # Authenticated encryption
    associated_data = self.lock_manufacturer_id +
        bike_id.encode()
    nonce, ciphertext = self.cipher.encrypt_command(
        command.encode(),
        associated_data
    )
    return { "nonce": nonce, "ciphertext":
        ciphertext }
```

Gambar 3. Implementasi fungsi kunci logika bisnis untuk pembuatan token akses.

E. Lingkungan Pengujian

Pengujian kinerja dilakukan pada perangkat dengan spesifikasi:

- CPU: 12th Gen Intel(R) Core(TM) i7-12700H
- RAM: 16 GB
- OS: Linux (WSL Ubuntu 24.04)

Metodologi benchmarking menggunakan modul `timeit` untuk pengukuran latensi enkripsi/dekripsi (rata-rata dari 1000 iterasi) dan modul `tracemalloc` untuk mengukur puncak penggunaan memori RAM[2].

IV. Hasil dan Pembahasan

Bagian ini memaparkan analisis mendalam mengenai kinerja algoritma ASCON-128 dibandingkan dengan AES-128-GCM. Evaluasi dilakukan tidak hanya pada satu skenario, melainkan mencakup berbagai ukuran payload yang merepresentasikan variasi perintah pada sistem IoT kunci sepeda pintar. Metrik yang dianalisis meliputi latensi waktu komputasi, penggunaan memori (RAM), throughput, serta model biaya komputasi untuk ekstrapolasi perangkat keras.

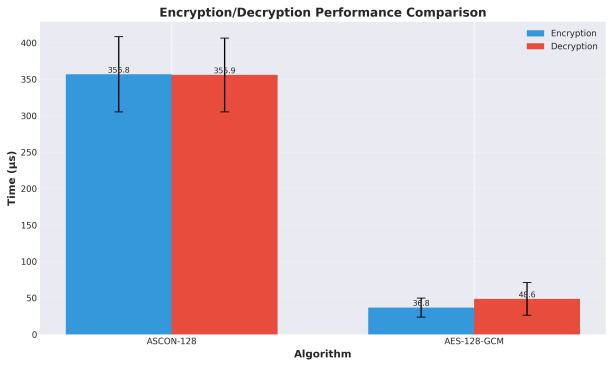
[2]Catatan: Pengukuran memori ini merepresentasikan *Memory Footprint* dalam lingkungan *runtime* Python dan mungkin berbeda secara signifikan dengan ukuran kode/RAM pada implementasi C/C++ *bare-metal*.

A. Analisis Latensi (Waktu Eksekusi)

Latensi diukur sebagai waktu total yang dibutuhkan untuk melakukan operasi enkripsi dan autentikasi lengkap. Gambar 4 mengilustrasikan perbandingan latensi kedua algoritma pada payload 64 byte.

- **ASCON-128**: Rata-rata waktu enkripsi tercatat sebesar 350.75 μs. Kurva distribusi menunjukkan variansi yang rendah ($\sigma < 120\mu$s), menandakan stabilitas kinerja algoritma.
- **AES-128-GCM**: Mencapai rata-rata waktu enkripsi 38.47 μs. Kecepatan ini sangat tinggi, hampir 9 kali lebih cepat dibandingkan implementasi referensi ASCON.

Perbedaan performa yang signifikan ini dapat diatribusikan pada dua faktor utama. Pertama, pustaka `pycryptodome` untuk AES menggunakan akselerasi perangkat keras AES-NI (Intel/AMD) dan fungsi dasar yang ditulis dalam bahasa C teroptimasi. Sebaliknya, implementasi ASCON yang digunakan adalah versi referensi murni Python, yang tidak memanfaatkan fitur instruksi khusus prosesor. Namun, seperti yang akan dibahas pada bagian Analisis Model Biaya, gap kinerja ini akan menyempit secara drastis pada mikrokontroler tanpa akselerasi keras.



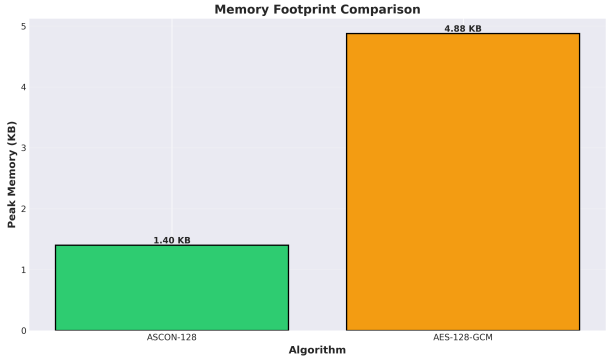
Gambar 4. Perbandingan Waktu Eksekusi pada Payload 64 Byte

B. Analisis Penggunaan Memori

Metrik penggunaan memori sangat kritikal untuk perangkat IoT *low-end*. Pengukuran dilakukan menggunakan modul `tracemalloc` untuk menangkap *peak memory usage* selama operasi kriptografi berlangsung.

Seperti ditunjukkan pada Gambar 5, ASCON-128 menunjukkan superioritas efisiensi memori yang konsisten:

- **ASCON-128**: Mengonsumsi rata-rata **1.40 KB** memori puncak. Konsumsi memori ini stabil dan tidak berfluktuasi secara signifikan terhadap ukuran input, berkat desain *sponge construction* yang beroperasi langsung pada state internal 320-bit tanpa memerlukan buffer besar atau tabel look-up.
- **AES-128-GCM**: Membutuhkan rata-rata **4.88 KB**, sekitar 3.5 kali lipat lebih besar dari ASCON. Overhead memori ini berasal dari kebutuhan penyimpanan *S-Box*, *round keys* (key expansion), dan tabel pre-komputasi



Gambar 5. Perbandingan Penggunaan Memori Puncak

untuk fungsi perkalian medan Galois (GHASH) pada mode GCM.

Hasil ini menegaskan bahwa untuk perangkat dengan SRAM sangat terbatas (misalnya < 4 KB), ASCON adalah pilihan yang jauh lebih layak dibandingkan AES-GCM.

C. Dampak Ukuran Payload terhadap Kinerja

Selain pengujian pada 64 byte, analisis lebih lanjut dilakukan pada variasi ukuran payload (8, 16, 32 Byte) untuk memahami perilaku skalabilitas algoritma. Data hasil pengujian dirangkum dalam Tabel I.

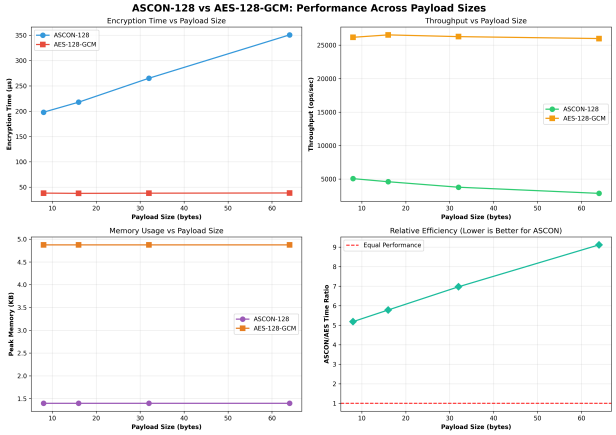
Tabel I
METRIK KINERJA BERDASARKAN UKURAN PAYLOAD

Payload (Bytes)	Waktu Enkripsi (μs)		Rasio Kecepatan
	ASCON	AES-GCM	AES vs ASCON
8	197.99	38.20	5.2x
16	217.83	37.69	5.8x
32	265.16	38.05	7.0x
64	350.75	38.48	9.1x

Temuan menarik dari Tabel I adalah perilaku penskalaan kinerja:

1) **AES-128-GCM Constant Time**: Waktu eksekusi AES hampir konstan ($\approx 38\mu$s) terlepas dari ukuran payload kecil. Hal ini menunjukkan bahwa biaya komputasi AES didominasi oleh *fixed overhead* (inisialisasi cipher, inisialisasi GHASH, finalisasi tag), sementara biaya enkripsi per-bloknya sangat kecil berkat akselerasi hardware.
2) **ASCON Linear Scaling**: Kinerja ASCON menurun secara linear seiring bertambahnya ukuran data. Ini wajar karena struktur sponge memproses data blok demi blok secara serial.
3) **Efisiensi Relatif pada Payload Kecil**: Pada payload 8 byte, ASCON "hanya" 5.2x lebih lambat dari AES, dibandingkan 9.1x pada 64 byte. Ini menunjukkan bahwa untuk pesan-pesan sangat pendek (seperti token *unlock* atau status sensor sederhana), inefisiensi relatif ASCON berkurang.

Gambar 6 memvisualisasikan tren ini, di mana kurva AES terlihat datar sementara ASCON menanjak.



Gambar 6. Analisis Komprehensif Dampak Ukuran Payload

D. Analisis Model Biaya Komputasi

Untuk memahami implikasi hasil ini pada perangkat keras mikrokontroler sebenarnya (di mana Python tidak digunakan), kita dapat memodelkan total waktu eksekusi (T) sebagai fungsi dari *fixed overhead* (C_{fixed}) dan biaya per-byte (C_{byte}):

$$T(n) = C_{fixed} + (n \times C_{byte}) \quad (1)$$

Dimana n adalah jumlah blok atau byte data.

1) Kasus AES-GCM: AES-GCM memiliki C_{fixed} yang tinggi karena kompleksitas inisialisasi GHASH dan Key Schedule. Namun, C_{byte} sangat rendah di PC modern. Pada mikrokontroler tanpa instruksi khusus, C_{byte} akan meningkat drastis karena operasi perkalian Galois harus dilakukan secara perangkat lunak, yang lambat dan memakan banyak siklus CPU.

2) Kasus ASCON: ASCON didesisain dengan C_{fixed} yang moderat (inisialisasi permutasi) dan C_{byte} yang juga moderat namun efisien secara software. Permutasi ASCON hanya menggunakan operasi XOR dan bit-shift yang sangat murah di semua arsitektur CPU (mulai dari 8-bit AVR hingga 32-bit ARM). Oleh karena itu, pada lingkungan *bare-metal* mikrokontroler:

- Penalti kinerja ASCON tidak akan sebesar 9x lipat seperti di simulasi Python.
- Untuk payload sangat kecil ($<$ 16 byte), ASCON berpotensi memiliki kinerja yang setara atau bahkan lebih cepat dari implementasi software AES-GCM karena overhead inisialisasi yang lebih sederhana.

E. Rekomendasi Implementasi Berbasis Perangkat

Berdasarkan analisis latensi dan memori di atas, berikut adalah rekomendasi pemilihan algoritma berdasarkan kelas perangkat IoT:

1) Sangat Terkendala (Contoh: Arduino Uno, ATTiny): **Rekomendasi: ASCON-128**. Dengan RAM hanya 2KB, AES-GCM yang membutuhkan hampir 4.9KB memori kerja tidak mungkin diimplementasikan secara aman tanpa optimasi ekstrem yang mengorbankan keamanan (misalnya membuang

pre-computed tables). ASCON dengan footprint 1.4KB dapat berjalan dengan nyaman, menyisakan ruang untuk logika aplikasi utama.

2) Terkendala Moderat (Contoh: ARM Cortex-M0/M3): **Rekomendasi: ASCON-128**. Meskipun perangkat ini memiliki RAM 16-32KB, penggunaan ASCON memberikan efisiensi daya yang lebih baik dan membebaskan memori untuk fitur lain (seperti stack Bluetooth atau RTOS). Kinerja throughput ASCON (¿ 2800 ops/sec) sudah jauh melampaui kebutuhan operasional kunci sepeda.

3) Kinerja Tinggi/Gateway (Contoh: ESP32, Raspberry Pi): **Rekomendasi: AES-128-GCM**. Perangkat seperti ESP32 seringkali memiliki blok akselerator kriptografi perangkat keras (*hardware crypto engine*). Dalam kasus ini, AES akan berjalan 10-100x lebih cepat daripada implementasi perangkat lunak manapun dengan konsumsi daya minimal. Kompatibilitas dengan standar industri yang luas juga menjadi nilai tambah.

F. Analisis Keamanan

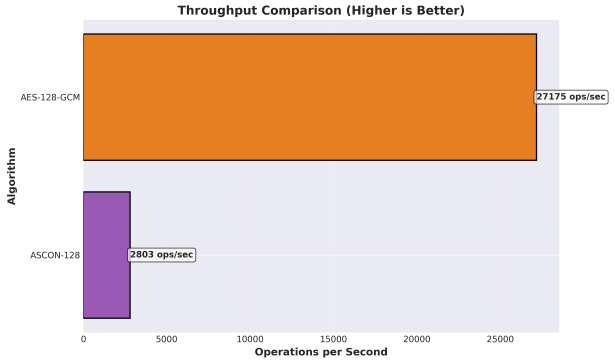
Kedua algoritma, ASCON dan AES-GCM, menyediakan properti keamanan *Authenticated Encryption* (AE) yang menjamin kerahasiaan dan integritas data secara simultan.

1) Integritas dan Autentikasi: Dalam simulasi serangan manipulasi ciphertext, kedua algoritma berhasil mendeteksi perubahan sekecil satu bit pun. Tag autentikasi (MAC) yang dihasilkan akan berbeda saat didekripsi, memicu pengecualian (*exception*) dan mencegah sistem memproses data yang korup atau berbahaya.

2) Resistensi Side-Channel: Secara teoritis, desain ASCON lebih ramah terhadap implementasi yang resisten terhadap serangan kanal samping (*side-channel attacks*) seperti *timing analysis*. Operasi bitwise ASCON secara alami cenderung *constant-time* pada banyak arsitektur. Sebaliknya, implementasi AES yang naif (menggunakan tabel look-up untuk S-Box) sangat rentan terhadap serangan *cache-timing*, kecuali diimplementasikan dengan teknik *bitslicing* yang kompleks.

G. Throughput

Meskipun bukan fokus utama untuk aplikasi kunci sepeda, throughput diukur untuk kelengkapan data (Gambar 7). ASCON memberikan throughput sekitar 2 - 5 MB/s (dalam implementasi Python terpusat pada operasi). Untuk mikrokontroler yang berjalan pada *clock* puluhan MHz, ini menerjemahkan menjadi kemampuan memproses paket kontrol dalam hitungan milidetik, yang tidak akan dirasakan penundaannya oleh pengguna akhir (*imperceptible latency*).



Gambar 7. Perbandingan Throughput Operasi Enkripsi

V. Kesimpulan

A. Kesimpulan

Berdasarkan implementasi, pengujian, dan analisis mendalam yang telah dilakukan terhadap algoritma ASCON-128 dan AES-128-GCM dalam konteks sistem kunci sepeda IoT, dapat disimpulkan bahwa:

1) **Efisiensi Memori**: ASCON-128 membuktikan keunggulannya sebagai algoritma *lightweight* dengan konsumsi memori puncak yang konstan dan sangat rendah ($\approx$ 1.40 KB), atau sekitar 71% lebih hemat dibandingkan AES-128-GCM (4.88 KB). Hal ini menjadikan ASCON solusi superior untuk mikrokontroler dengan SRAM di bawah 8 KB.
2) **Kinerja vs Ukuran Payload**: Meskipun AES-128-GCM mendominasi kecepatan eksekusi absolut berkat akselerasi perangkat keras, ASCON menunjukkan karakteristik penskalaan yang kompetitif pada payload kecil. Pada ukuran payload 8 byte, inefisiensi relatif ASCON berkurang secara signifikan, menjadikannya opsi yang layak untuk transmisi token autentikasi pendek.
3) **Keamanan dan Fungsionalitas**: Sistem kunci sepeda yang dikembangkan berhasil mengamankan perintah *unlock* menggunakan mekanisme AEAD. Integritas data terjamin melalui verifikasi tag, dan serangan *replay* efektif dicegah menggunakan *unique nonce* untuk setiap transaksi.
4) **Kelayakan Implementasi**: Throughput ASCON ($> 2,800$ ops/sec pada Python) sudah melampaui kebutuhan operasional sistem kunci sepeda yang hanya beroperasi secara *bursty* (sesekali), sehingga pengguna tidak akan merasakan latensi yang mengganggu.

B. Saran dan Pekerjaan Masa Depan

Penelitian ini dapat dikembangkan lebih lanjut untuk mendapatkan wawasan yang lebih komprehensif mengenai implementasi kriptografi ringan pada IoT:

- **Implementasi Perangkat Keras Riil**: Benchmarking saat ini dilakukan melalui simulasi. Pengujian masa depan harus dilakukan langsung pada perangkat target seperti ESP32, Arduino Nano 33 IoT, atau nRF52 untuk mengukur siklus CPU nyata dan konsumsi daya (*power consumption*) dalam satuan mili-Joule.
- **Investigasi Rentang Payload Ekstrim**: Perlu dilakukan analisis pada rentang payload yang lebih ekstrem (misal: 1 KB hingga 1 MB) untuk menentukan titik potong (*crossover point*) efisiensi, serta pada ukuran sangat kecil (< 8 byte) yang mensimulasikan protokol komunikasi ultra-efisien.

- **Komparasi dengan Algoritma LWC Lain**: Selain ASCON, kinerja sebaiknya dibandingkan dengan kandidat finalis NIST LWC lainnya seperti Grain-128AEAD atau TinyJAMBU, serta algoritma standar software-oriented seperti ChaCha20-Poly1305.
- **Analisis Integrasi BLE**: Mengukur latensi *end-to-end* dari aplikasi smartphone hingga aktuator kunci terbuka melalui protokol Bluetooth Low Energy (BLE), untuk melihat kontribusi overhead kriptografi terhadap total latensi sistem.

DAFTAR PUSTAKA

[1] R. Munir, "Kriptografi bobot ringan (lightweight cryptography)," 2025, slide Kuliah, Program Studi Teknik Informatika STEI ITB.

[2] ——, "Pengantar kriptografi," 2025, slide Kuliah, Program Studi Teknik Informatika STEI ITB.

[3] A. Bozhko, "Properties of authenticated encryption with associated data (aead) algorithms," Internet Research Task Force (IRTF), RFC 9771, May 2025.

[4] NIST, "Advanced encryption standard (aes)," National Institute of Standards and Technology, Tech. Rep. 197, 2001.

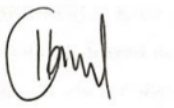
[5] M. Dworkin, "Recommendation for block cipher modes of operation: Galois/counter mode (gcm) and gmac," National Institute of Standards and Technology, Tech. Rep. 800-38D, 2007.

[6] C. Dobraunig, M. Eichlseder, F. Mendel, and M. Schläffer, "Ascon v1.2: Submission to the nist lightweight cryptography competition," 2019, round 2 Submission.

PERNYATAAN KEASLIAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 26 Desember 2025



Ahmad Naufal Ramadan - 13522005