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*—Makalah ini menyajikan analisis komparatif antara ASCON-128 (Standar NIST LWC) dan AES-128-GCM pada simulasi lingkungan terbatas. Kami berfokus pada latensi CPU, penggunaan memori, dan overhead kode untuk aplikasi kunci sepeda pintar. Hasil pengujian menunjukkan bahwa ASCON-128 memiliki keunggulan dalam penggunaan memori yang lebih rendah, menjadikannya kandidat kuat untuk perangkat IoT dengan sumber daya sangat terbatas, meskipun AES-128-GCM unggul dalam throughput pada perangkat dengan akselerasi perangkat keras.**

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

I. Pendahuluan

Internet of Things (IoT) telah berkembang pesat, menghubungkan miliaran perangkat ke internet. Namun, perangkat-perangkat ini seringkali memiliki keterbatasan sumber daya komputasi, memori, dan daya baterai. Kriptografi standar seperti AES (Advanced Encryption Standard) seringkali terlalu berat untuk diimplementasikan pada perangkat kelas bawah tanpa akselerasi perangkat keras khusus.

Pada tahun 2023, NIST mengumumkan standar baru untuk Kriptografi Ringan (Lightweight Cryptography/LWC), yaitu keluarga algoritma ASCON [1]. Standar ini dirancang khusus untuk memberikan keamanan yang kuat pada lingkungan terbatas.

Dalam makalah ini, kami mengimplementasikan dan membandingkan kinerja ASCON-128 dan AES-128-GCM dalam konteks sistem kunci sepeda pintar (Smart Bicycle Lock). Sistem ini memerlukan autentikasi yang cepat dan aman untuk membuka kunci, dengan overhead daya dan memori seminimal mungkin. Fokus utama penelitian ini adalah mengukur trade-off antara latensi eksekusi dan konsumsi memori dari kedua algoritma tersebut.

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`.

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

- `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.

```
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.

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.

```
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 perintah 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.

IV. Hasil dan Pembahasan

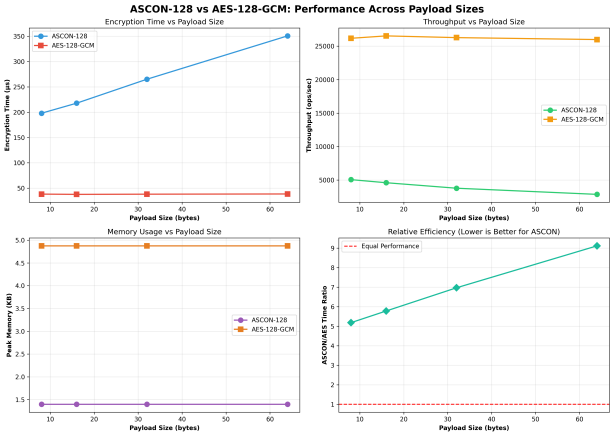
Bagian ini memaparkan hasil benchmarking kinerja antara ASCON-128 dan AES-128-GCM. Metrik yang dievaluasi adalah latensi waktu komputasi, penggunaan memori, dan throughput.

A. Analisis Latensi (Waktu Eksekusi)

Berdasarkan data pengujian dengan payload 64 byte (ukuran tipikal perintah unlock) yang diilustrasikan pada Gambar 4:

- **ASCON-128**: Rata-rata waktu enkripsi adalah 350.75 μs dan dekripsi 348.13 μs.
- **AES-128-GCM**: Rata-rata waktu enkripsi adalah 38.47 μs dan dekripsi 49.67 μs.

AES-GCM terlihat jauh lebih cepat (hampir 9x lipat) dalam simulasi ini. Hal ini disebabkan oleh pustaka `pycryptodome` yang mengutilisasi instruksi AES-NI pada prosesor Intel/AMD modern dan implementasi *backend* C yang teroptimasi. Di sisi lain, implementasi ASCON berjalan murni pada Python (*pure-python reference*), yang mensimulasikan kinerja pada perangkat tanpa akselerasi perangkat keras.



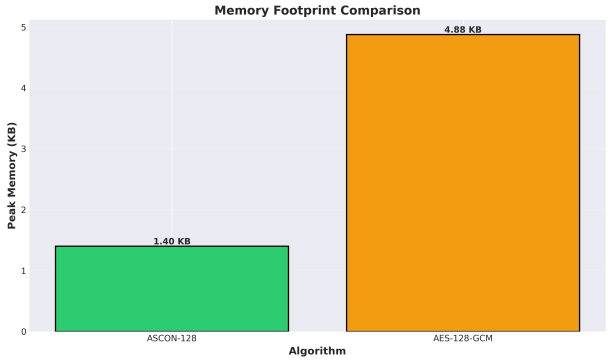
Gambar 4. Analisis Waktu Enkripsi Berdasarkan Ukuran Payload

B. Analisis Penggunaan Memori

Pengukuran memori puncak (*peak memory usage*) menunjukkan keunggulan signifikan ASCON:

- **ASCON-128**: Rata-rata penggunaan memori puncak hanya **1.40 KB**.
- **AES-128-GCM**: Rata-rata penggunaan memori puncak mencapai **4.88 KB**.

Hasil ini konsisten dengan desain ASCON yang memang ditujukan untuk perangkat terkekang (*constrained devices*) dengan RAM sangat terbatas, seperti yang ditunjukkan pada Gambar 5. AES-GCM membutuhkan tabel *lookup* yang lebih besar dan state management yang lebih kompleks.



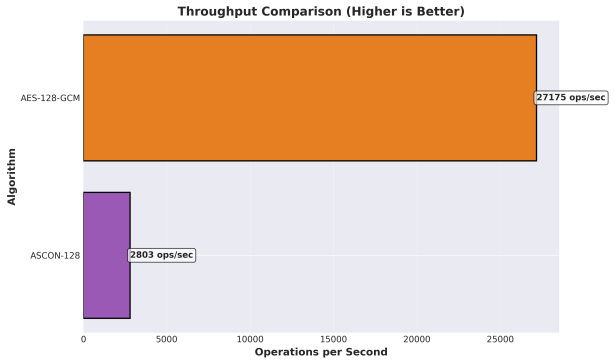
Gambar 5. Perbandingan Penggunaan Memori Puncak

C. Throughput

Throughput enkripsi pada payload 64 byte (Gambar 6):

- ASCON-128: $\approx$ 2,851 operasi/detik.
- AES-128-GCM: $\approx$ 25,989 operasi/detik.

Meskipun throughput AES lebih tinggi di PC, ASCON memberikan kinerja yang memadai ($>$ 2000 ops/sec) yang sudah sangat cukup untuk aplikasi kunci sepeda yang hanya beroperasi sesekali (tidak *real-time streaming*).



Gambar 6. Perbandingan Throughput Operasi Enkripsi

D. Analisis Keamanan

Kedua algoritma berhasil memverifikasi integritas data. Percobaan modifikasi ciphertext menghasilkan kegagalan autentikasi (tag mismatch), memastikan sistem aman dari serangan manipulasi pesan.

V. Kesimpulan

A. Kesimpulan

Berdasarkan implementasi dan pengujian yang dilakukan, dapat disimpulkan bahwa:

1) **ASCON-128 lebih efisien dalam penggunaan memori**, mengonsumsi hanya sekitar 1.4 KB peak RAM dibandingkan AES-GCM yang membutuhkan 4.9 KB. Ini menjadikan ASCON pilihan ideal untuk mikrokontroler low-end (seperti AVR atau ARM Cortex-M0) yang memiliki keterbatasan SRAM.
2) **AES-128-GCM menawarkan kecepatan eksekusi yang lebih tinggi** pada perangkat yang mendukung instruksi AES-NI. Namun, pada perangkat IoT tanpa akselerasi tersebut, keunggulan ini mungkin tidak signifikan atau justru berbalik karena kompleksitas tabel AES.
3) Implementasi sistem kunci sepeda pintar berhasil mengamankan perintah *unlock* menggunakan kedua algoritma dengan mekanisme `nonce` unik untuk mencegah *replay attacks*.

B. Saran

Untuk pengembangan selanjutnya, disarankan untuk:

- Melakukan pengukuran daya (*power consumption benchmarking*) pada perangkat keras IoT fisik (misalnya ESP32).
- Mengimplementasikan versi teroptimasi C dari ASCON untuk membandingkan kinerja secara lebih adil dengan AES OpenSSL/PyCryptodome.

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