Implementasi dan Pengujian Mekanisme Pencatatan Data Sensor Aman Berbasis Ascon AEAD128 dan Ascon Hash256

Sonya Putri Fadilah - 18223138
Program Studi Sistem dan Teknologi Informasi
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: sonyaputri0099@gmail.com , 18223138@std.stei.itb.ac.id

Abstract*—Perangkat sensor pada sistem Internet of Things menghasilkan data secara berkala yang dapat digunakan untuk pemantauan lingkungan, industri, kesehatan, dan sistem otomatisasi. Data tersebut perlu dilindungi karena dapat berisi informasi penting dan rentan terhadap pembacaan, perubahan, penghapusan, penyisipan, atau pengacakan urutan log. Makalah ini mengimplementasikan dan menguji mekanisme pencatatan data sensor aman berbasis Ascon AEAD128 dan Ascon Hash256. Ascon AEAD128 digunakan untuk mengenkripsi isi record sensor dan memvalidasi integritas metadata melalui associated data, sedangkan Ascon Hash256 digunakan untuk membentuk hash chain antar-record. Implementasi dilakukan dalam bahasa Python menggunakan data sensor simulasi dengan variasi 100, 1000, dan 10000 record. Pengujian mencakup verifikasi normal, kesesuaian hasil dekripsi, pengukuran waktu proses, overhead ukuran data, serta deteksi manipulasi ciphertext, tag autentikasi, timestamp, device ID, penghapusan record, penyisipan record, dan pertukaran urutan record. Hasil pengujian menunjukkan bahwa seluruh data normal berhasil diverifikasi dan seluruh skenario manipulasi dapat terdeteksi. Namun, mekanisme menghasilkan overhead ukuran yang besar karena penyimpanan komponen keamanan dalam format JSON dan heksadesimal.

***Keywords*—Ascon AEAD128; Ascon Hash256; lightweight cryptography; sensor logging; hash chain; authenticated encryption**

I. Pendahuluan

Perangkat sensor banyak digunakan dalam sistem *Internet of Things* untuk mengumpulkan data dari lingkungan fisik secara berkala. Data tersebut dapat berupa suhu, kelembapan, tekanan udara, intensitas cahaya, status perangkat, atau parameter lain yang digunakan dalam pemantauan lingkungan, industri, kesehatan, maupun sistem otomatisasi. Dalam banyak kasus, data sensor tidak langsung diproses, tetapi disimpan terlebih dahulu sebagai log sebelum dikirim ke server atau dianalisis lebih lanjut. Oleh karena itu, keamanan log sensor menjadi aspek penting karena data tersebut dapat memengaruhi hasil pemantauan dan pengambilan keputusan.

Ancaman terhadap log sensor tidak hanya berupa pembacaan data oleh pihak tidak berwenang, tetapi juga perubahan isi *record*, pengubahan metadata, penghapusan *record*, penyisipan *record* palsu, dan pertukaran urutan *record*. Jika manipulasi tersebut tidak terdeteksi, sistem dapat menggunakan data yang tidak valid. Sebagai contoh, perubahan pada data suhu, kelembapan, atau tekanan dapat menyebabkan kesalahan analisis pada sistem pemantauan gudang, pertanian, atau mesin industri. Dengan demikian, mekanisme keamanan log sensor perlu menjaga kerahasiaan isi *record*, integritas metadata, dan keterurutan antar-*record*.

Pada perangkat sensor dan IoT, pemilihan algoritma kriptografi perlu mempertimbangkan keterbatasan komputasi, memori, daya, dan *bandwidth*. *Lightweight cryptography* menjadi relevan karena dirancang untuk memberikan perlindungan kriptografis dengan beban yang lebih sesuai untuk perangkat terbatas. Salah satu algoritma yang relevan adalah Ascon, yaitu keluarga algoritma kriptografi ringan yang mencakup *authenticated encryption* dan *hash function*. Dalam makalah ini, Ascon AEAD128 digunakan untuk mengamankan setiap *record* sensor, sedangkan Ascon Hash256 digunakan untuk membentuk hubungan antar-*record* melalui *hash chain*.

Makalah ini mengusulkan mekanisme pencatatan data sensor aman berbasis kombinasi Ascon AEAD128 dan Ascon Hash256. Nilai sensor diamankan menggunakan *authenticated encryption* sehingga isi data menjadi rahasia dan perubahan terhadap *record* dapat dideteksi. Selain itu, setiap *record* dikaitkan dengan *record* sebelumnya melalui *hash chain* agar manipulasi struktur log, seperti penghapusan, penyisipan, atau perubahan urutan *record*, dapat diketahui saat proses verifikasi. Mekanisme ini juga menggunakan *trusted anchor* yang menyimpan informasi akhir rantai hash sebagai acuan verifikasi.

Kontribusi utama makalah ini adalah merancang struktur *secure log* untuk data sensor, mengimplementasikan proses enkripsi, dekripsi, pembentukan tag autentikasi, pembentukan *hash chain*, dan verifikasi log, serta menguji mekanisme terhadap beberapa skenario manipulasi. Pengujian dilakukan menggunakan data sensor simulasi berukuran 100, 1000, dan 10000 *record*. Evaluasi mencakup kesesuaian hasil dekripsi, deteksi manipulasi, waktu proses, dan *overhead* ukuran data.

II. Dasar Teori

A. Lightweight Cryptography

Lightweight cryptography merupakan cabang kriptografi yang dirancang untuk perangkat dengan sumber daya terbatas, seperti sensor IoT, RFID, *smart card*, *wireless sensor network*, dan *embedded microcontroller*. Perangkat tersebut umumnya memiliki keterbatasan pada kapasitas memori, daya, kemampuan prosesor, serta *bandwidth* komunikasi. Oleh karena itu, algoritma kriptografi yang digunakan pada perangkat terbatas perlu memiliki beban komputasi dan ukuran implementasi yang relatif kecil, tetapi tetap memberikan tingkat keamanan yang memadai [5].

Kebutuhan *lightweight cryptography* muncul karena algoritma kriptografi umum tidak selalu efisien untuk perangkat sangat terbatas. Pada konteks sensor dan IoT, sistem tidak hanya membutuhkan enkripsi, tetapi juga mekanisme untuk mendeteksi manipulasi data. Karena itu, *authenticated encryption* menjadi salah satu kebutuhan penting. Dalam makalah ini, *lightweight cryptography* digunakan sebagai dasar pemilihan Ascon karena mekanisme yang dirancang ditujukan untuk skenario pencatatan data sensor yang membutuhkan perlindungan kerahasiaan, integritas, dan autentikasi.

B. Authenticated Encryption with Associated Data

Authenticated Encryption with Associated Data (AEAD) adalah skema kriptografi simetris yang menggabungkan enkripsi dan autentikasi data [6]. AEAD memberikan perlindungan kerahasiaan terhadap *plaintext* serta perlindungan integritas dan autentikasi terhadap ciphertext maupun *associated data*. *Associated data* adalah data yang tidak dienkripsi, tetapi tetap dilindungi integritasnya. Jika *associated data* berubah, proses verifikasi tag akan gagal dan *ciphertext* tidak boleh didekripsi.

Secara umum, proses enkripsi AEAD dapat dituliskan sebagai berikut:

$$C_i \,||\, T_i = \text{Enc\_K}(N_i, P_i, AD_i) \quad (1)$$

dengan K sebagai kunci rahasia, N_i sebagai *nonce* untuk *record* ke-i, P_i sebagai *plaintext*, AD_i sebagai *associated data*, C_i sebagai ciphertext, dan T_i sebagai *authentication tag*. Pada proses dekripsi, sistem menerima K, N_i, C_i, T_i, dan AD_i. Jika tag valid, *plaintext* dikembalikan. Jika tag tidak valid, dekripsi ditolak. Sifat ini membuat AEAD sesuai untuk data sensor karena nilai sensor dapat dienkripsi, sedangkan metadata seperti *sequence number*, *timestamp*, dan *device* ID dapat tetap terbaca tetapi tidak dapat dimodifikasi tanpa terdeteksi.

C. Ascon AEAD128 dan Ascon Hash256

Ascon adalah keluarga algoritma kriptografi ringan yang mencakup *authenticated encryption*, *hash function*, dan *extendable output function*. Dalam standar NIST SP 800-232, keluarga Ascon mencakup Ascon-AEAD128, Ascon-Hash256, Ascon-XOF128, dan Ascon-CXOF128 [1]. Ascon menggunakan struktur berbasis *permutation* dengan *state* internal 320 bit [2]. Desain ini ditujukan untuk memberikan keamanan modern dengan implementasi yang relatif ringan pada perangkat terbatas.

Dalam makalah ini, Ascon AEAD128 digunakan untuk mengamankan isi setiap *record* sensor. Algoritma ini menerima kunci, *nonce, plaintext,* dan *associated data*, lalu menghasilkan *ciphertext* dan tag autentikasi. *Plaintext* pada rancangan ini berisi nilai sensor, sedangkan *associated data* berisi metadata yang harus tetap dapat dibaca tetapi tidak boleh diubah.

Ascon Hash256 digunakan untuk menghasilkan *digest* 256 bit dari data masukan [2]. Pada makalah ini, fungsi hash tersebut tidak digunakan sebagai *message authentication code*, melainkan sebagai komponen pembentuk hash chain. Dengan demikian, Ascon Hash256 berfungsi untuk mengaitkan satu *record* dengan *record* sebelumnya sehingga perubahan struktur log dapat terdeteksi saat proses verifikasi.

D. Hash Chain dan Trusted Anchor

Hash chain adalah mekanisme yang menghubungkan sekumpulan data secara berurutan menggunakan fungsi hash. Setiap *record* menyimpan nilai hash dari *record* sebelumnya, sehingga perubahan pada satu *record* dapat memengaruhi validitas hubungan dengan *record* berikutnya. Pada pencatatan log, *hash chain* dapat membuat log bersifat *tamper-evident*, yaitu manipulasi terhadap isi atau urutan log dapat diketahui saat proses verifikasi [7].

Secara sederhana, hubungan antar-*record* dalam *hash chain* dapat ditulis sebagai berikut:

$$\text{previous\_hash}_i = H_{\{i-1\}} \quad (2)$$

$$H_i = \text{Hash}(\text{record}_i \,||\, \text{previous\_hash}_i) \quad (3)$$

dengan H_i sebagai *hash record* ke-i. Untuk *record* pertama, previous_hash dapat diisi dengan nilai awal tertentu yang disebut genesis hash. Dalam rancangan makalah ini, hash record dihitung dari metadata, *nonce, previous hash, ciphertext,* dan *authentication tag*. Dengan demikian, perubahan terhadap salah satu komponen *record* dapat menyebabkan hash hasil perhitungan ulang berbeda dari hash yang tersimpan.

Meskipun hash chain dapat mendeteksi perubahan pada struktur log, *hash chain* biasa bukanlah MAC berkunci. Jika penyerang dapat mengubah seluruh log dan menghitung ulang semua hash, maka *hash chain* saja tidak cukup untuk memberikan bukti integritas. Oleh karena itu, mekanisme ini menggunakan *trusted anchor*, yaitu nilai acuan yang disimpan pada tempat yang lebih terpercaya. *Anchor* pada makalah ini menyimpan jumlah *record* dan *final record hash*:

$$\text{Anchor} = (n, H_n) \quad (4)$$

dengan n sebagai jumlah *record* dan H_n sebagai *hash record* terakhir. *Anchor* penting untuk mendeteksi manipulasi seperti penghapusan *record* terakhir, karena penghapusan bagian akhir log dapat membuat urutan *record* yang tersisa tetap terlihat benar, tetapi jumlah *record* dan *final record hash* tidak lagi sesuai dengan *anchor*.

III. Rancangan Mekanisme Secure Logging

A. Struktur Data Sensor

Data sensor yang digunakan pada mekanisme ini berbentuk *record* terstruktur. Setiap *record* merepresentasikan satu pembacaan sensor pada waktu tertentu. Atribut yang digunakan terdiri dari *sequence number, timestamp, device* ID*, temperature, humidity, pressure*, dan *light*. *Sequence number* digunakan untuk menunjukkan urutan pencatatan, *timestamp* menunjukkan waktu pencatatan, *device* ID menunjukkan identitas perangkat sensor, sedangkan *temperature, humidity, pressure,* dan *light* merupakan nilai sensor yang perlu dilindungi.

Secara umum, satu *record* data sensor sebelum diamankan dapat ditulis sebagai berikut:

$$\text{record}_i = (\text{seq}_i, \text{time}_i, \text{dev}_i, \text{temp}_i, \text{hum}_i, \text{press}_i, \text{light}_i) \quad (5)$$

dengan seq_i sebagai *sequence number,* time_i sebagai *timestamp*, dev_i sebagai *device* ID, serta temp_i, hum_i, press_i, dan light_i sebagai nilai sensor pada *record* ke-i. Pada rancangan ini, metadata dan nilai sensor diperlakukan secara berbeda agar mekanisme tetap mendukung kebutuhan pencarian dan verifikasi.

B. Pembagian Plaintext dan Associated Data

Rancangan ini membagi setiap record menjadi dua bagian, yaitu *plaintext* dan *associated data*. *Plaintext* berisi nilai sensor yang perlu dijaga kerahasiaannya, sedangkan *associated data* berisi metadata yang tetap dapat dibaca tetapi harus dilindungi integritasnya. Pembagian tersebut ditunjukkan pada Tabel I.

TABLE I. Pembagian Komponen Record Sensor

Komponen	Atribut	Perlakuan
Plaintext	temperature, humidity, pressure, light	Dienkripsi menggunakan Ascon AEAD128
Associated data	sequence number, timestamp, device ID, previous hash	Tidak dienkripsi, tetapi diautentikasi
Komponen kriptografis	nonce, ciphertext, authentication tag, record hash	Disimpan pada secure log

Pembagian ini dilakukan karena metadata seperti *sequence number, timestamp,* dan *device* ID sering kali masih dibutuhkan untuk identifikasi, pencarian, atau pengurutan log. Namun, metadata tersebut tidak boleh dapat diubah tanpa terdeteksi. Oleh karena itu, metadata ditempatkan sebagai *associated data*. Jika *timestamp, device* ID, atau *sequence number* dimodifikasi, proses verifikasi tag autentikasi akan gagal.

Associated data pada *record* ke-i dirumuskan sebagai berikut:

$$\text{AD}_i = \text{metadata}_i \,||\, \text{previous\_hash}_i \quad (6)$$

dengan metadata_i berisi *sequence number, timestamp*, dan *device* ID. Nilai previous_hash_i ikut dimasukkan ke dalam *associated data* agar hubungan dengan *record* sebelumnya juga terikat pada proses autentikasi per-*record*.

C. Proses Enkripsi Per Record

Setiap *record* diamankan menggunakan Ascon AEAD128. Proses enkripsi menerima kunci rahasia K, *nonce* N_i, *plaintext* P_i, dan *associated data* AD_i. Output yang dihasilkan adalah *ciphertext* C_i dan *authentication tag* T_i. Proses tersebut dapat dituliskan sebagai berikut:

$$\text{C}_i \,||\, \text{T}_i = \text{Enc\_K}(\text{N}_i, \text{P}_i, \text{AD}_i) \quad (7)$$

Pada rancangan ini, *plaintext* P_i berisi nilai *temperature, humidity, pressure,* dan *light*. Sementara itu, AD_i berisi metadata dan *previous hash*. Dengan mekanisme AEAD, nilai sensor menjadi rahasia karena dienkripsi menjadi ciphertext. Pada saat yang sama, *associated data* tetap tidak terenkripsi, tetapi perubahan terhadap *associated data* dapat dideteksi melalui validasi *authentication tag*.

Nonce pada implementasi eksperimen diturunkan secara deterministik dari *associated data* dan *previous hash* menggunakan Ascon Hash256. Tujuannya adalah agar eksperimen dapat direproduksi dan hasil pengujian dapat dibandingkan kembali. Secara konseptual, *nonce* eksperimen dapat ditulis sebagai berikut:

$$\text{N}_i = \text{Hash}(\text{"nonce"} \,||\, \text{AD}_i)[0{:}16] \quad (8)$$

Rumus (8) menunjukkan bahwa nonce diambil dari 16 byte pertama hasil hash. Namun, pada implementasi nyata, *nonce* harus dikelola dengan kebijakan yang menjamin keunikan untuk setiap proses enkripsi dengan kunci yang sama. Penggunaan *nonce* deterministik pada makalah ini hanya ditujukan untuk kebutuhan eksperimen dan *reproducibility*.

Alur umum mekanisme *secure logging* yang diusulkan ditunjukkan pada Fig. 1. Mekanisme dimulai dari pemisahan data sensor menjadi *plaintext* dan *associated data*, kemudian dilanjutkan dengan enkripsi menggunakan Ascon AEAD128, pembentukan *ciphertext* dan *authentication tag*, serta pembentukan *hash chain* menggunakan Ascon Hash256.

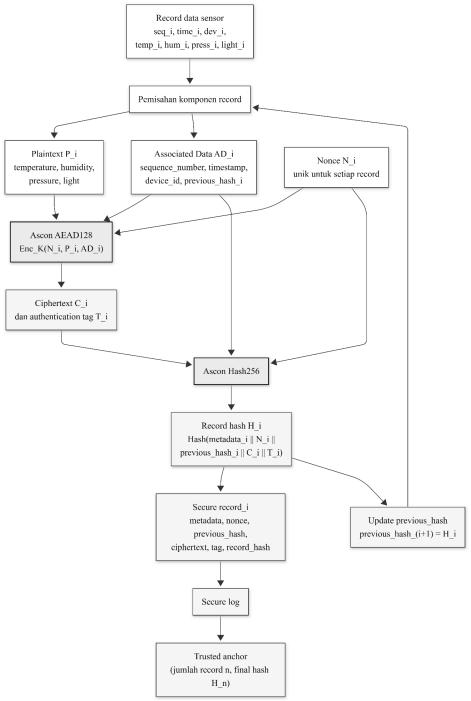

Fig. 1. Alur mekanisme secure logging berbasis Ascon AEAD128 dan Ascon Hash256.

D. Pembentukan Hash Chain

Setelah *ciphertext* dan *authentication tag* diperoleh, sistem membentuk *hash chain* menggunakan Ascon Hash256. *Hash chain* digunakan untuk menghubungkan *record* saat ini dengan *record* sebelumnya. Setiap *record* menyimpan *previous hash*, yaitu hash dari *record* sebelumnya. Untuk *record* pertama, *previous hash* diisi dengan *genesis hash,* yaitu nilai awal berupa 32 byte nol.

Hubungan antar-*record* dirumuskan sebagai berikut:

$$\text{previous\_hash}_i = H_{\{i-1\}} \tag{9}$$

$$H_i = \text{Hash}(\text{metadata}_i \,||\, N_i \,||\, \text{previous\_hash}_i \,||\, C_i \,||\, T_i) \tag{10}$$

dengan H_i sebagai *record* hash pada record ke-i. Rumus (10) menunjukkan bahwa *hash record* dihitung dari metadata, *nonce, previous hash, ciphertext,* dan *authentication tag*. Dengan demikian, perubahan pada salah satu komponen tersebut akan mengubah hasil hash. Jika *record* dihapus, disisipkan, atau ditukar urutannya, hubungan *previous hash* antar-*record* akan menjadi tidak sesuai.

Struktur *secure log* yang dihasilkan untuk setiap *record* adalah sebagai berikut:

secure_record$_i$ = {
sequence_number,
timestamp,
device_id,
nonce,
previous_hash,
ciphertext,
authentication_tag,
record_hash
}

Struktur ini membuat *secure log* memiliki dua lapisan perlindungan. Lapisan pertama adalah AEAD yang menjaga kerahasiaan dan integritas setiap *record*. Lapisan kedua adalah *hash chain* yang menjaga keterkaitan antar-*record*. Kombinasi keduanya membuat log bersifat *tamper-evident*, yaitu manipulasi terhadap isi maupun struktur log dapat terdeteksi saat proses verifikasi.

E. Trusted Anchor

Hash chain biasa belum cukup jika penyerang dapat mengubah seluruh log dan menghitung ulang semua hash. Untuk mengatasi kelemahan tersebut, rancangan ini menggunakan *trusted anchor*. *Anchor* merupakan nilai acuan yang diasumsikan disimpan di tempat yang lebih tepercaya, misalnya server, media terpisah, atau sistem verifikasi pusat. *Anchor* menyimpan jumlah *record* dan hash terakhir dari rantai log.

Anchor dirumuskan sebagai berikut:

$$\text{Anchor} = (n, H_n) \tag{11}$$

dengan n sebagai jumlah *record* dan H_n sebagai *record hash* terakhir. Anchor berperan penting untuk mendeteksi manipulasi pada bagian akhir log. Misalnya, jika record terakhir dihapus, *sequence number* pada *record* yang tersisa masih dapat terlihat berurutan. Namun, jumlah *record* dan *final record hash* tidak lagi sesuai dengan *anchor*. Dengan demikian, penghapusan *record* terakhir tetap dapat terdeteksi.

F. Proses Verifikasi dan Pembacaan Log

Proses verifikasi dilakukan dengan membaca seluruh *secure log*, kemudian memeriksa tiga aspek utama. Pertama, sistem memeriksa *sequence number* untuk memastikan urutan record sesuai. Kedua, sistem menghitung ulang *record hash* dan memeriksa kecocokan *previous hash* antar-*record*. Ketiga, sistem melakukan dekripsi menggunakan Ascon AEAD128 untuk memvalidasi *authentication tag* dan mengembalikan *plaintext*.

Tahapan verifikasi dapat diringkas sebagai berikut. Sistem membaca *secure log* dan *trusted anchor*. Untuk setiap *record*, sistem memeriksa apakah *sequence number* sesuai dengan posisi *record*. Selanjutnya, sistem memeriksa apakah *previous hash* pada *record* saat ini sama dengan *record hash* sebelumnya. Sistem kemudian menghitung ulang *record hash* berdasarkan metadata, *nonce, previous hash, ciphertext*, dan *authentication tag*. Setelah itu, sistem melakukan dekripsi menggunakan kunci, *nonce, ciphertext, tag,* dan *associated data*. Jika seluruh pemeriksaan berhasil, log dinyatakan valid dan *plaintext* sensor dapat dikembalikan. Jika salah satu pemeriksaan gagal, *record* atau struktur log dianggap telah dimanipulasi.

Mekanisme ini dirancang untuk mendeteksi beberapa bentuk manipulasi. Perubahan *ciphertext*, *authentication tag, timestamp*, atau *device* ID akan merusak validitas AEAD dan *hash record.* Penghapusan *record*, penyisipan *record*, dan pertukaran urutan *record* akan merusak *sequence number* atau hubungan *hash chain*. Penghapusan *record* terakhir dapat dideteksi melalui ketidaksesuaian jumlah *record* atau *final record hash* terhadap *trusted anchor.*

IV. Implementasi dan Pengujian

A. Lingkungan Implementasi

Mekanisme *secure logging* diimplementasikan menggunakan bahasa pemrograman *Python*. Implementasi dibuat sebagai prototipe eksperimen perangkat lunak untuk menguji rancangan pencatatan data sensor aman berbasis Ascon AEAD128 dan Ascon Hash256. Program terdiri dari beberapa komponen utama, yaitu implementasi fungsi Ascon, pembentukan *secure log*, generator dataset simulasi, modul eksperimen, serta *unit test.*

Komponen program yang digunakan adalah sebagai berikut. File ascon.py berisi implementasi Ascon AEAD128 dan Ascon Hash256. File secure_logger.py berisi fungsi untuk memisahkan *record* sensor menjadi *plaintext* dan *associated data,* melakukan enkripsi, membentuk *hash chain,* membuat *trusted anchor,* melakukan dekripsi, serta memverifikasi log. File dataset.py digunakan untuk membangkitkan data sensor simulasi. File experiments.py digunakan untuk menjalankan eksperimen performa dan skenario manipulasi. File run_all.py digunakan sebagai program utama untuk menjalankan seluruh eksperimen dan menghasilkan file hasil berupa CSV serta grafik.

Implementasi ini tidak ditujukan sebagai *library* produksi. Program dibuat untuk kepentingan eksperimen akademik dan demonstrasi mekanisme. Oleh karena itu, hasil performa yang diperoleh merepresentasikan implementasi *Python* yang dijalankan pada lingkungan pengujian tertentu, bukan performa Ascon pada implementasi *embedded* atau implementasi C yang dioptimalkan.

B. Dataset Simulasi

Pengujian dilakukan menggunakan data sensor simulasi. Data ini tidak berasal dari perangkat IoT fisik, tetapi dibangkitkan oleh program untuk merepresentasikan pembacaan sensor secara berkala. Setiap *record* memiliki atribut *sequence number, timestamp, device* ID, *temperature, humidity, pressure,* dan *light*. *Timestamp* dibuat meningkat satu menit untuk setiap *record*, sedangkan *sequence number* menunjukkan urutan pencatatan.

Dataset simulasi digunakan karena tujuan utama makalah ini adalah menguji mekanisme keamanan log, bukan mengevaluasi akurasi sensor fisik. Dengan dataset simulasi, jumlah *record* dapat diatur sesuai kebutuhan eksperimen. Pada pengujian ini digunakan tiga ukuran dataset, yaitu 100 record, 1000 record, dan 10000 record. Ketiga ukuran tersebut digunakan untuk mewakili skenario kecil, sedang, dan besar.

Contoh struktur data sensor sebelum diamankan adalah sequence_number, timestamp, device_id, temperature, humidity, pressure, light.

Setelah diamankan, data sensor tidak lagi disimpan dalam bentuk nilai sensor asli. Nilai sensor dienkripsi menjadi *ciphertext*, sedangkan metadata tetap disimpan bersama *nonce*, *previous hash, authentication tag*, dan *record hash* dalam *secure log.*

C. Parameter Eksperimen

Eksperimen menggunakan satu kunci rahasia berukuran 128 bit untuk proses Ascon AEAD128. *Nonce* pada eksperimen diturunkan secara deterministik dari *associated data* dan *previous hash* menggunakan Ascon Hash256. Pemilihan *nonce* deterministik dilakukan agar hasil eksperimen dapat direproduksi. Namun, dalam implementasi nyata, *nonce* harus dikelola dengan kebijakan yang menjamin keunikan untuk setiap enkripsi dengan kunci yang sama.

Parameter utama eksperimen ditunjukkan pada Tabel II.

TABLE II. Parameter Eksperimen

Parameter	Nilai
Algoritma enkripsi	Ascon AEAD128
Algoritma *hash*	Ascon Hash256
Ukuran kunci	128 bit
Ukuran *nonce*	128 bit
Ukuran *hash*	256 bit
Format *secure log*	JSON Lines
Encoding komponen biner	Heksadesimal
Jumlah *record*	100, 1000, 10000
Data uji	Data sensor simulasi
Bahasa pemrograman	Python

D. Skenario Pengujian

Pengujian dilakukan dalam dua kelompok, yaitu pengujian normal dan pengujian manipulasi. Pengujian normal bertujuan memastikan bahwa data sensor dapat diamankan, diverifikasi, dan didekripsi kembali sesuai data awal. Pengujian manipulasi bertujuan memastikan bahwa perubahan terhadap isi *record*, metadata, atau struktur log dapat terdeteksi.

Skenario manipulasi yang digunakan ditunjukkan pada Tabel III.

TABLE III. Skenario Pengujian Manipulasi

No.	Skenario	Tujuan
1	*Ciphertext* diubah 1 byte	Menguji deteksi manipulasi isi data terenkripsi
2	*Authentication tag* diubah	Menguji validasi tag autentikasi
3	*Timestamp* diubah	Menguji perlindungan *associated data*
4	*Device* ID diubah	Menguji perlindungan metadata perangkat
5	Satu *record* dihapus	Menguji deteksi penghapusan *record* di tengah log
6	*Record* terakhir dihapus	Menguji fungsi *trusted anchor*
7	*Record* duplikat disisipkan	Menguji deteksi penyisipan *record*
8	Urutan dua record ditukar	Menguji deteksi perubahan urutan log

Pada skenario perubahan *ciphertext, tag, timestamp, dan device* ID, sistem diharapkan mendeteksi manipulasi melalui validasi AEAD dan perhitungan ulang *record hash*. Pada skenario penghapusan, penyisipan, dan pertukaran urutan *record*, sistem diharapkan mendeteksi manipulasi melalui pemeriksaan *sequence number, previous hash, record hash*, dan *trusted anchor.*

E. *Metrik Evaluasi*

Evaluasi dilakukan menggunakan beberapa metrik. Pertama, status verifikasi normal digunakan untuk memastikan secure log valid ketika tidak dimanipulasi. Kedua, kesesuaian hasil dekripsi digunakan untuk memastikan bahwa *plaintext* hasil dekripsi sama dengan data sensor awal. Ketiga, deteksi manipulasi digunakan untuk melihat apakah setiap skenario serangan berhasil terdeteksi. Keempat, waktu proses digunakan untuk mengukur waktu enkripsi, dekripsi, verifikasi *hash chain*, dan verifikasi penuh. Kelima, *overhead* ukuran digunakan untuk membandingkan ukuran file data sensor asli dengan ukuran *secure log*.

Overhead ukuran data dihitung menggunakan rumus berikut:

$$\text{Overhead (\%)} = \left(\frac{(\text{ukuran secure log} - \text{ukuran plaintext})}{\text{ukuran plaintext}}\right) \times 100\% \quad (12)$$

Metrik ini penting karena mekanisme *secure logging* menambahkan beberapa komponen keamanan pada setiap *record*, yaitu *nonce, ciphertext, authentication tag, previous hash,* dan *record hash*. Dengan demikian, evaluasi tidak hanya menunjukkan keberhasilan mekanisme dalam mendeteksi manipulasi, tetapi juga menunjukkan biaya tambahan dari sisi waktu proses dan ukuran penyimpanan.

V. HASIL DAN ANALISIS

A. *Hasil Verifikasi Normal*

Pengujian pertama dilakukan untuk memastikan bahwa *secure log* yang tidak dimanipulasi dapat diverifikasi dan didekripsi kembali dengan benar. Hasil pengujian menunjukkan bahwa seluruh dataset berukuran 100, 1000, dan 10000 record memiliki status verifikasi normal valid. Selain itu, hasil dekripsi pada seluruh dataset sesuai dengan *plaintext* awal. Hal ini menunjukkan bahwa proses enkripsi, penyimpanan *secure log*, pembacaan ulang, verifikasi, dan dekripsi berjalan sesuai dengan rancangan.

Hasil ini juga menunjukkan bahwa pembagian data menjadi *plaintext* dan *associated data* tidak mengganggu proses pemulihan data. Nilai sensor yang dijadikan *plaintext* dapat dikembalikan setelah proses dekripsi, sedangkan metadata yang dijadikan *associated data* tetap dapat digunakan pada proses verifikasi. Dengan demikian, mekanisme yang diusulkan dapat menjaga kerahasiaan nilai sensor tanpa menghilangkan kemampuan sistem untuk membaca metadata penting seperti *sequence number, timestamp*, dan *device* ID.

B. *Hasil Performa dan Overhead Ukuran Data*

Pengujian performa dilakukan pada tiga ukuran dataset, yaitu 100, 1000, dan 10000 *record*. Metrik yang diukur meliputi waktu enkripsi, waktu dekripsi, waktu verifikasi penuh, ukuran *plaintext*, ukuran *secure log, overhead* ukuran data, kesesuaian hasil dekripsi, dan status verifikasi normal. Hasil pengujian ditunjukkan pada Tabel IV.

TABLE IV. HASIL PENGUJIAN PERFORMA DAN OVERHEAD

Jumlah record	Enkripsi (ms)	Dekripsi (ms)	Verifikasi penuh (ms)	Ukuran plaintext (KB)	Ukuran secure log (KB)	Overhead (%)	Dekripsi sesuai	Status
100	679,224	133,770	544,410	5,976	50,684	748,18	Ya	Valid
1000	7250,648	938,152	4142,246	60,136	507,929	744,64	Ya	Valid
10000	64046,497	8231,148	45269,618	610,240	5088,605	733,87	Ya	Valid

Berdasarkan Tabel IV, waktu proses meningkat ketika jumlah *record* bertambah. Pada 100 *record*, waktu enkripsi sebesar 679,224 ms, sedangkan pada 10000 *record* waktu enkripsi meningkat menjadi 64046,497 ms. Pola serupa juga terlihat pada waktu dekripsi dan verifikasi penuh. Hal ini wajar karena setiap *record* diproses secara individual melalui pembentukan *associated data,* pembuatan *nonce*, enkripsi AEAD, pembentukan *tag autentikasi*, perhitungan *hash chain*, dan proses verifikasi.

Fig. 2 menunjukkan perbandingan waktu proses untuk setiap jumlah *record*. Grafik tersebut memperlihatkan bahwa enkripsi membutuhkan waktu lebih besar dibandingkan dekripsi. Selain itu, verifikasi penuh juga membutuhkan waktu cukup besar karena proses ini tidak hanya memeriksa *hash chain,* tetapi juga melakukan validasi AEAD terhadap setiap *record*.

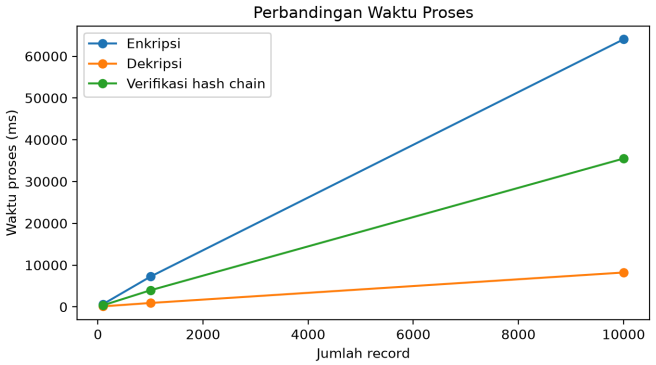

Fig. 2. Grafik waktu proses enkripsi, dekripsi, dan verifikasi penuh.

Hasil pada Tabel IV tidak boleh diartikan sebagai performa umum Ascon pada seluruh platform. Implementasi yang diuji merupakan implementasi eksperimen berbasis Python, bukan implementasi C atau *embedded* yang dioptimalkan. Oleh karena itu, waktu proses yang diperoleh lebih tepat digunakan untuk membandingkan perilaku mekanisme terhadap peningkatan jumlah *record* pada lingkungan pengujian yang sama.

Dari sisi ukuran data, *secure log* memiliki ukuran jauh lebih besar dibandingkan data *plaintext*. Pada 100 *record*, ukuran *plaintext* sebesar 5,976 KB, sedangkan ukuran *secure log* sebesar 50,684 KB dengan *overhead* 748,18%. Pada 10000 *record*, ukuran *plaintext* sebesar 610,240 KB, sedangkan ukuran *secure log* sebesar 5088,605 KB dengan *overhead* 733,87%. *Overhead* yang besar ini terjadi karena *secure log* tidak hanya menyimpan *ciphertext*, tetapi juga menyimpan metadata, *nonce, authentication tag, previous hash,* dan *record hash* pada setiap *record*. Selain itu, implementasi menggunakan format JSON Lines dan *encoding*

heksadesimal untuk komponen biner, sehingga ukuran penyimpanan menjadi lebih besar dibandingkan format biner.

Walaupun *overhead* ukuran cukup besar, nilai *overhead* tersebut masih dapat dijelaskan sebagai konsekuensi dari mekanisme keamanan tambahan. Jika mekanisme ini dikembangkan lebih lanjut, *overhead* ukuran dapat dikurangi dengan menggunakan format penyimpanan yang lebih ringkas, seperti format biner atau CBOR, serta mengurangi metadata yang tidak diperlukan.

C. Hasil Deteksi Manipulasi

Pengujian manipulasi dilakukan untuk memastikan bahwa mekanisme dapat mendeteksi perubahan pada isi *record*, metadata, dan struktur log. Skenario pengujian dilakukan pada *secure log* berisi 100 *record*. Hasil deteksi manipulasi ditunjukkan pada Tabel V.

TABLE V. HASIL PENGUJIAN DETEKSI MANIPULASI

Skenario	Terdeteksi	Terdeteksi oleh AEAD	Terdeteksi oleh hash chain/anchor	Terdeteksi oleh sequence	Status akhir
Normal / tanpa manipulasi	Tidak	Tidak	Tidak	Tidak	Valid
Ciphertext diubah 1 byte	Ya	Ya	Ya	Tidak	Gagal diverifikasi
Authentication tag diubah	Ya	Ya	Ya	Tidak	Gagal diverifikasi
Timestamp diubah	Ya	Ya	Ya	Tidak	Gagal diverifikasi
Device ID diubah	Ya	Ya	Ya	Tidak	Gagal diverifikasi
Satu *record* dihapus	Ya	Tidak	Ya	Ya	Gagal diverifikasi
Record terakhir dihapus	Ya	Tidak	Ya	Tidak	Gagal diverifikasi
Record duplikat disisipkan	Ya	Tidak	Ya	Ya	Gagal diverifikasi
Urutan dua record ditukar	Ya	Tidak	Ya	Ya	Gagal diverifikasi

Berdasarkan Tabel V, seluruh skenario manipulasi berhasil terdeteksi. Pada skenario *ciphertext* diubah, *authentication tag* diubah, *timestamp* diubah, dan *device* ID diubah, manipulasi terdeteksi oleh AEAD dan *hash chain*. Hal ini terjadi karena perubahan pada *ciphertext*, tag, atau *associated data* menyebabkan validasi tag autentikasi gagal. Selain itu, perubahan komponen *record* juga menyebabkan nilai *record* hash hasil perhitungan ulang berbeda dari *record hash* yang tersimpan.

Pada skenario satu *record* dihapus, *record* duplikat disisipkan, dan urutan dua *record* ditukar, manipulasi terutama terdeteksi oleh pemeriksaan *sequence number* dan *hash chain.* AEAD tidak secara langsung mendeteksi manipulasi jenis ini apabila isi *record* yang tersisa masih valid secara individual. Hal ini menunjukkan bahwa AEAD bekerja pada tingkat per-*record*, sedangkan manipulasi struktur log perlu dideteksi oleh mekanisme antar-*record* seperti *sequence number, previous hash, record hash,* dan *trusted anchor.*

Skenario *record* terakhir dihapus menunjukkan fungsi penting *trusted anchor*. Ketika *record* terakhir dihapus, urutan *sequence number* pada *record* yang tersisa masih dapat terlihat benar, sehingga pemeriksaan *sequence* tidak mendeteksi kesalahan. Namun, jumlah *record* dan *final record hash* tidak lagi sesuai dengan *anchor* tepercaya. Oleh karena itu, manipulasi tetap dapat terdeteksi oleh *hash chain/anchor.* Hasil ini menunjukkan bahwa *anchor* diperlukan untuk memperkuat *hash chain*, terutama untuk mendeteksi perubahan pada bagian akhir log.

D. Analisis Keamanan Mekanisme

Mekanisme yang diusulkan memberikan dua lapisan perlindungan. Lapisan pertama adalah Ascon AEAD128 yang menjaga kerahasiaan nilai sensor dan memverifikasi integritas *associated data*. Dengan AEAD, data sensor seperti *temperature, humidity, pressure,* dan *light* tidak disimpan dalam bentuk *plaintext*, tetapi dalam bentuk *ciphertext*. Metadata seperti *sequence number, timestamp, device* ID, dan *previous hash* tetap dapat dibaca, tetapi perubahan terhadap metadata tersebut menyebabkan validasi tag gagal.

Lapisan kedua adalah Ascon Hash256 yang digunakan untuk membentuk *hash chain.* Lapisan ini membuat setiap *record* bergantung pada *record* sebelumnya. Jika satu *record* diubah, dihapus, disisipkan, atau ditukar, hubungan previous hash dan *record hash* menjadi tidak konsisten. Dengan demikian, *hash chain* membantu mendeteksi manipulasi struktur log yang tidak selalu dapat dideteksi oleh AEAD per-*record*.

Namun, *hash chain* yang digunakan pada mekanisme ini bukan *message authentication code* berkunci. Artinya, apabila penyerang dapat mengubah seluruh log dan juga mengganti *anchor*, maka penyerang berpotensi menghitung ulang seluruh *hash chain*. Oleh karena itu, anchor harus disimpan pada tempat yang lebih tepercaya, misalnya server, media terpisah, atau sistem verifikasi pusat. Dengan asumsi *anchor* tidak dapat dimodifikasi oleh penyerang, perubahan terhadap jumlah *record* atau *final record hash* dapat terdeteksi.

E. Keterbatasan Implementasi

Terdapat beberapa keterbatasan pada implementasi ini. Pertama, implementasi dibuat menggunakan *Python* sebagai prototipe eksperimen perangkat lunak, sehingga hasil performa tidak merepresentasikan performa Ascon pada perangkat IoT nyata atau implementasi yang dioptimalkan. Kedua, *nonce* pada eksperimen dibuat deterministik agar hasil dapat direproduksi. Pada sistem nyata, *nonce* harus dikelola dengan kebijakan yang menjamin keunikan untuk setiap enkripsi dengan kunci yang sama. Ketiga, secure log disimpan dalam format JSON Lines dengan *encoding* heksadesimal, sehingga *overhead* ukuran menjadi besar. Keempat, data yang digunakan merupakan data sensor simulasi, bukan data dari perangkat IoT fisik.

Meskipun demikian, keterbatasan tersebut tidak mengubah tujuan utama eksperimen. Implementasi ini tetap dapat menunjukkan bahwa kombinasi Ascon AEAD128 dan Ascon Hash256 dapat digunakan untuk membangun mekanisme

secure logging yang menjaga kerahasiaan isi *record*, melindungi integritas metadata, dan mendeteksi manipulasi terhadap struktur log.

VI. Kesimpulan

Makalah ini telah mengimplementasikan dan menguji mekanisme pencatatan data sensor aman berbasis Ascon AEAD128 dan Ascon Hash256. Mekanisme yang diusulkan menggunakan Ascon AEAD128 untuk menjaga kerahasiaan nilai sensor serta memvalidasi integritas metadata melalui *associated data.* Selain itu, Ascon Hash256 digunakan untuk membentuk *hash chain* antar-*record* agar perubahan terhadap struktur log dapat terdeteksi. Rancangan ini juga menggunakan trusted anchor yang menyimpan jumlah *record* dan *final record hash* untuk memperkuat verifikasi terhadap keseluruhan log.

Hasil pengujian menunjukkan bahwa *secure log* normal pada dataset 100, 1000, dan 10000 *record* berhasil diverifikasi dengan status valid. Hasil dekripsi juga sesuai dengan *plaintext* awal, sehingga proses enkripsi, penyimpanan, verifikasi, dan dekripsi berjalan sesuai rancangan. Pada pengujian manipulasi, seluruh skenario berhasil terdeteksi, yaitu perubahan *ciphertext*, perubahan *authentication tag*, perubahan *timestamp*, perubahan device ID, penghapusan *record*, penghapusan record terakhir, penyisipan *record* duplikat, dan pertukaran urutan *record*. Hasil ini menunjukkan bahwa AEAD efektif untuk mendeteksi perubahan pada isi *record* dan *associated data*, sedangkan *hash chain, sequence number,* dan *trusted anchor* efektif untuk mendeteksi manipulasi struktur log.

Dari sisi performa, waktu proses meningkat seiring bertambahnya jumlah *record*. Pada implementasi eksperimen berbasis *Python*, enkripsi dan verifikasi penuh membutuhkan waktu lebih besar dibandingkan dekripsi, terutama pada dataset 10000 *record*. Dari sisi ukuran, *secure log* menghasilkan overhead yang besar karena setiap record menyimpan metadata, *nonce, ciphertext, authentication tag, previous hash,* dan *record hash* dalam format JSON Lines dan heksadesimal. Oleh karena itu, mekanisme ini berhasil meningkatkan keamanan pencatatan data sensor, tetapi masih memiliki biaya tambahan dari sisi waktu proses dan ukuran penyimpanan.

Pengembangan lebih lanjut dapat dilakukan dengan mengimplementasikan mekanisme ini pada bahasa pemrograman yang lebih efisien, seperti C, atau pada perangkat *embedded* untuk memperoleh evaluasi performa yang lebih representatif. Selain itu, format penyimpanan dapat dioptimalkan menggunakan format biner atau CBOR untuk mengurangi *overhead* ukuran data. Pengelolaan *nonce* juga perlu disesuaikan dengan kebutuhan sistem nyata agar keunikan *nonce* tetap terjamin untuk setiap proses enkripsi dengan kunci yang sama.

Code Repository

Source code implementasi, skrip eksperimen, dataset simulasi, dan hasil pengujian tersedia pada repositori GitHub berikut:
https://github.com/sonyaaputri/ascon-sensor-secure-logging

Video Link at Youtube

Video penjelasan makalah:
https://www.youtube.com/watch?v=w1xI3AGU79w

Acknowledgment

Penulis mengucapkan puji dan syukur kepada Allah Swt. atas rahmat dan karunia-Nya sehingga makalah ini dapat diselesaikan dengan baik. Penulis juga mengucapkan terima kasih kepada Bapak Dr. Ir. Rinaldi Munir, M.T. selaku dosen pengampu mata kuliah II4021 Kriptografi atas materi dan pembelajaran yang diberikan selama perkuliahan. Ucapan terima kasih juga disampaikan kepada orang tua atas dukungan dan motivasi yang diberikan kepada penulis.

References

[1] M. Sönmez Turan, K. McKay, D. Chang, J. Kelsey, C. Bassham, J. Kang, and R. Moody, "Ascon-based lightweight cryptography standards for constrained devices," NIST Special Publication 800-232, National Institute of Standards and Technology, 2025.

[2] C. Dobraunig, M. Eichlseder, F. Mendel, and M. Schläffer, "Ascon v1.2: Lightweight authenticated encryption and hashing," Journal of Cryptology, vol. 34, no. 3, article 33, 2021.

[3] National Institute of Standards and Technology, "NIST selects 'lightweight cryptography' algorithms to protect small devices," NIST, 2023.

[4] M. Sönmez Turan, "Lightweight cryptography standardization," National Institute of Standards and Technology, 2023.

[5] R. Munir, "Kriptografi bobot ringan," Materi Kuliah II4021 Kriptografi, Program Studi Sistem dan Teknologi Informasi, Institut Teknologi Bandung, 2026.

[6] P. Rogaway, "Authenticated-encryption with associated-data," in Proceedings of the 9th ACM Conference on Computer and Communications Security, 2002, pp. 98-107.

[7] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, "Duplexing the sponge: Single-pass authenticated encryption and other applications," in Selected Areas in Cryptography, 2012, pp. 320-337.

PERNYATAAN

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, 19 Juni 2026

Sonya Putri Fadilah
18223138