Analisis Performa FF1 Format-Preserving Encryption dan Envelope Encryption untuk Perlindungan Data PII pada Backend Microservice

Muhammad Omar Berliansyah - 18223055
Program Studi Sistem dan Teknologi Informasi
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
Email: muhammadomarberliansyah@gmail.com, 18223055@std.stei.itb.ac.id

***Abstrak*—Data PII pada layanan backend tidak cukup hanya disembunyikan. Beberapa field, seperti NIK, nomor telepon, dan nomor rekam medis, tetap perlu berbentuk angka agar mudah divalidasi dan dicari. Makalah ini menguji prototipe dua layanan yang memakai FF1 format-preserving encryption untuk kolom numerik, AES-GCM untuk teks bebas, envelope encryption untuk pemisahan DEK/KEK, blind index untuk pencarian NIK, dan Record MAC untuk integritas data FF1. Data yang dipakai adalah data pasien sintetis. Pengujian dilakukan dengan k6, query PostgreSQL, uji rewrap, dan enam simulasi serangan. Dari tiga kali run, mode FF1-envelope mencatat p95 latency 183,89 ms, p99 latency 223,72 ms, dan throughput 84,57 permintaan per detik pada operasi create. Ukuran baris rata-ratanya naik 187,20% dari plaintext, masih lebih kecil daripada AES-GCM-only yang naik 228,89%. Rewrap 10.000 record selesai dalam 117,74 detik tanpa error. Semua simulasi keamanan lulus dalam batas model ancaman kompromi basis data.**

***Kata Kunci*—FF1, format-preserving encryption, AES-GCM, envelope encryption, blind index, PII, key management service**

I. Pendahuluan

Sistem backend untuk layanan kesehatan, layanan publik, dan manajemen identitas menyimpan banyak data yang dekat dengan identitas seseorang. Pada contoh layanan pasien, data itu dapat berupa NIK, nomor telepon, nomor rekam medis, nama, alamat, dan catatan diagnosis. Menyimpan semuanya sebagai plaintext memang paling mudah untuk aplikasi. Masalahnya, jika basis data bocor, data yang sama juga langsung mudah dibaca oleh penyerang.

AES-GCM bisa menyelesaikan sebagian masalah karena memberi kerahasiaan dan integritas. Namun, hasil enkripsinya tidak mempertahankan bentuk data awal. NIK yang semula 16 digit tidak lagi terlihat seperti NIK setelah dienkripsi. Untuk sistem yang sudah punya validasi format, constraint kolom, atau integrasi lama, perubahan bentuk seperti ini bisa menjadi pekerjaan tambahan yang cukup mengganggu.

FF1 format-preserving encryption (FPE) dipakai untuk bagian yang butuh format tetap. FF1 dapat mengenkripsi string numerik sambil menjaga radix dan panjang output [1]. Dalam prototipe ini, NIK 16 digit tetap menjadi 16 digit setelah dienkripsi. Akan tetapi, FF1 bukan AEAD dan tidak membawa tag autentikasi. Karena itu, saya menambahkan Record MAC untuk mendeteksi perubahan pada ciphertext FF1.

Bagian lain yang tidak kalah penting adalah manajemen kunci. Jika satu kunci jangka panjang dipakai langsung untuk semua record, rotasi kunci menjadi berat. Envelope encryption membuat alurnya lebih masuk akal: data dienkripsi dengan DEK per record, lalu DEK itu dibungkus oleh KEK yang dikelola KMS. Saat KEK berubah, sistem cukup melakukan rewrap DEK. Payload PII tidak perlu dienkripsi ulang.

Makalah ini berangkat dari implementasi, bukan hanya pembahasan konsep. Prototipe backend microservice dibangun untuk membandingkan tiga mode penyimpanan: plaintext, AES-GCM-only, dan FF1-envelope. Dari sana, pengujian diarahkan ke performa create, read, dan search, ukuran penyimpanan, biaya rewrap, serta respons sistem terhadap beberapa simulasi serangan.

Pertanyaan penelitian pada makalah ini adalah sebagai berikut:

1) Berapa overhead latensi dan throughput yang ditambahkan FF1-envelope dibandingkan plaintext dan AES-GCM-only?
2) Seberapa besar overhead penyimpanan akibat ciphertext, nonce, tag, wrapped DEK, blind index, key version, dan metadata Record MAC?
3) Berapa waktu graceful rewrap KEK untuk skala 1.000, 5.000, dan 10.000 record?
4) Bagaimana prototipe merespons inspeksi dump basis data, pertukaran ciphertext pada AES-GCM, manipulasi pada FF1, pencarian dengan blind index, dan skenario KEK yang dihancurkan?

Kontribusi makalah ini ada pada tiga hal. Pertama, rancangan backend dua layanan yang memisahkan data pasien dari manajemen kunci. Kedua, implementasi FF1, AES-GCM, envelope encryption, blind index, dan Record MAC dalam satu alur kerja. Ketiga, evaluasi performa, penyimpanan, rewrap, dan simulasi keamanan berdasarkan artefak eksperimen. Pemetaan ke NIST dan ISO dipakai sebagai rujukan teknis, bukan sebagai klaim sertifikasi atau kepatuhan formal.

II. Metodologi penelitian

A. Metode penelitian

Penelitian ini dimulai dari studi literatur singkat, lalu dilanjutkan dengan perancangan sistem, implementasi, dan eksperimen. Studi literatur dipakai untuk menentukan batas teknis: FF1 untuk data numerik berformat tetap, AES-GCM untuk teks bebas, AES pembungkusan kunci untuk DEK, dan KMS untuk memisahkan pengelolaan KEK dari basis data pasien. Setelah itu, rancangan sistem dibuat dengan membagi tanggung jawab antara main service, KMS service, dan dua basis data.

Implementasi memakai dua service FastAPI. Main service menangani API pasien, enkripsi, dekripsi, pencarian, endpoint benchmark, dan rewrap worker. KMS service membuat DEK, membungkus dan membuka DEK, merotasi KEK, serta menolak operasi untuk kunci yang sudah dihancurkan. PostgreSQL dipakai untuk menyimpan rekam data pasien dan metadata KMS. Seluruh komponen dijalankan dengan Docker Compose.

Bagian eksperimen dibuat sesederhana mungkin agar hasilnya mudah ditelusuri. k6 dipakai untuk mengukur latency dan throughput API. Query PostgreSQL dipakai untuk mengukur ukuran baris. Script rewrap mengukur biaya rotasi KEK pada 1.000, 5.000, dan 10.000 record. Untuk sisi keamanan, script Python menjalankan skenario yang meniru kompromi basis data parsial.

B. Ruang lingkup dan batasan

Prototipe hanya memakai data pasien sintetis. Tidak ada PII nyata yang digunakan. Eksperimen dijalankan pada Docker Compose lokal agar variabel luar tidak terlalu banyak masuk ke hasil. Karena itu, angka yang muncul di makalah ini sebaiknya dibaca sebagai hasil eksperimen lokal, bukan angka performa produksi. Frontend juga tidak dibuat karena tidak menambah bukti kriptografi; sumber angka tetap berasal dari k6, SQL, dan script evaluasi.

Model ancaman dibatasi pada kompromi basis data parsial. Penyerang boleh membaca tabel pasien, termasuk ciphertext, wrapped DEK, nonce, tag, blind index, key version, dan Record MAC. Namun, penyerang tidak diasumsikan menguasai KMS service, material KEK, environment variable, runtime memory, atau host machine. Makalah ini juga tidak mengevaluasi SQL injection, OAuth, access control produksi, side-channel attack, dan memory scraping. KMS yang dipakai adalah service prototipe, bukan HSM.

C. Rancangan eksperimen

Tabel I merangkum komponen eksperimen. Setiap skenario utama k6 dijalankan tiga kali, lalu script analisis menghitung rerata p50, p95, p99, permintaan per detik (RPS), dan tingkat error.

Benchmark create membandingkan mode plaintext, AES-GCM-only, dan FF1-envelope. Benchmark operasi membandingkan create, read, dan search pada mode FF1-envelope. Untuk penyimpanan, sistem memasukkan 1.000 record per mode lalu mengukur ukuran baris dengan PostgreSQL. Uji rewrap memakai tiga ukuran data: 1.000, 5.000, dan 10.000 record.

TABLE I
Komponen Eksperimen

Komponen	Tujuan	Alat
Benchmark k6	Latensi API, p95, p99, dan RPS	k6
Query penyimpanan	Ukuran baris rata-rata dan overhead	PostgreSQL
Benchmark rewrap	Biaya rotasi KEK	Script/k6
Simulasi serangan	Perilaku terhadap model ancaman	Python

Evaluasi keamanan mencakup dump basis data, pertukaran ciphertext, manipulasi pada FF1, dynamic tweak dan blind index, blind index search, serta penghancuran KEK.

III. Dasar teori

A. PII dan Enkripsi Tingkat Kolom

PII merujuk pada data yang dapat mengarah ke identitas seseorang. Dalam prototipe ini, bentuknya dibuat sebagai rekam data pasien sintetis: NIK, nomor telepon, nomor rekam medis, nama, alamat, dan catatan diagnosis. Enkripsi tingkat kolom dipakai agar perlindungan terjadi sebelum data masuk ke basis data. Jadi, sekalipun tabel pasien dibaca langsung, kolom sensitif tidak tampil sebagai plaintext biasa.

B. FF1 format-preserving encryption

FF1 adalah metode FPE dari NIST untuk mengenkripsi nilai pada alfabet berhingga [1]. Karena NIK, nomor telepon, dan nomor rekam medis berbentuk angka, prototipe memakai radix 10. Kolom numerik dengan panjang n memiliki ukuran domain:

$$|\mathcal{D}| = 10^n. \tag{1}$$

Sebagai contoh, NIK 16 digit tetap berada dalam domain desimal 16 digit setelah enkripsi FF1. FF1 diterapkan pada NIK, nomor telepon, dan nomor rekam medis. Nama, alamat, dan catatan diagnosis tidak memakai FF1 karena ketiganya berupa teks bebas.

Prototipe membentuk tweak dinamis dengan HMAC-SHA256:

$$T = \mathrm{HMAC}_{K_t}(table \parallel column \parallel record\_id \parallel scheme\_version). \tag{2}$$

Tweak memuat identitas record dan konteks field, tetapi tidak memuat versi KEK. Dengan pilihan ini, rotasi KEK tidak memaksa sistem mengenkripsi ulang kolom FF1.

C. AES-GCM dan data terasosiasi

AES-GCM dipakai untuk kolom teks bebas. Mode ini termasuk AEAD dan dispesifikasikan dalam NIST SP 800-38D [2]. Selain menyembunyikan plaintext, AES-GCM juga memeriksa apakah ciphertext atau data terasosiasi berubah. Prototipe memakai nonce 96-bit dan tag 128-bit untuk setiap kolom AES-GCM.

Data terasosiasi mengikat ciphertext ke record dan kolomnya. AAD dibentuk dari ID record, nama kolom, dan versi

skema. Versi KEK sengaja tidak dimasukkan karena rewrap KEK tidak boleh merusak tag AES-GCM pada payload PII. Jika penyerang memindahkan ciphertext ke record lain, verifikasi tag akan gagal.

D. Envelope Encryption dan Pembungkusan Kunci

Envelope encryption membagi peran kunci menjadi dua. DEK dipakai untuk mengenkripsi data record, sedangkan KEK dipakai untuk membungkus DEK. Basis data utama hanya menyimpan wrapped DEK. Material KEK tetap berada di KMS service.

Prototipe memakai AES-KW-256 untuk pembungkusan kunci. Rujukannya adalah NIST SP 800-38F [3]. Saat rotasi KEK, main service meminta KMS membuka wrapped DEK lama dan membungkusnya kembali dengan KEK baru. Kolom PII terenkripsi tidak berubah.

E. Blind Index dan Record MAC

Sistem memakai blind index agar pencarian NIK tidak perlu mendekripsi semua baris. NIK dinormalisasi lebih dulu, lalu sistem menghitung:

$$BI_N = \mathrm{HMAC}_{K_b}(N). \tag{3}$$

Query basis data mencari nilai deterministik ini. Desain tersebut membuat pencarian kesamaan nilai bisa dilakukan, tetapi ada harga yang harus diterima: nilai yang sama menghasilkan blind index yang sama.

Karena FF1 tidak memiliki tag autentikasi, prototipe menambahkan Record MAC:

$$MAC_r = \mathrm{HMAC}_{K_m}(id_r \parallel C_{nik} \parallel C_{hp} \parallel C_{mrn} \parallel v_{kek} \parallel v_s). \tag{4}$$

Main service memverifikasi MAC ini sebelum mendekripsi kolom FF1. Jika ciphertext FF1 diubah, nilai MAC tidak lagi cocok.

F. Backend Microservice dan Model Ancaman

Main service dan KMS service dipisahkan. Main service dapat membaca basis data pasien, tetapi tidak membaca basis data KMS. KMS service menyimpan metadata kunci, tetapi tidak membaca rekam data pasien. Pemisahan ini membantu saat skenario yang diuji hanya berupa dump basis data. Jika penyerang sudah menguasai runtime aplikasi atau KMS, pembahasan makalah ini tidak lagi cukup.

IV. Rancangan dan Implementasi Sistem

A. Arsitektur Microservice

Prototipe dibagi menjadi empat komponen: main service, KMS service, main database, dan KMS database. Alurnya dibuat sederhana seperti pada Gambar 1. Client atau k6 hanya memanggil main service. Main service menyimpan rekam data pasien di main database dan memanggil KMS service saat membutuhkan DEK, unwrap, atau rewrap. KMS service sendiri hanya menyimpan metadata KEK di KMS database.

Desain ini tidak menyertakan antarmuka depan. Alasannya cukup langsung: antarmuka depan tidak diperlukan untuk menjawab pertanyaan penelitian. Angka performa, ukuran penyimpanan, dan hasil simulasi keamanan semuanya berasal dari script otomatis.

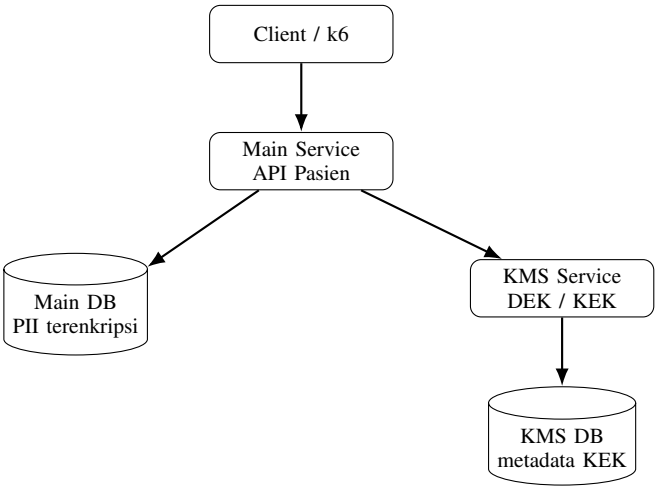

Fig. 1. Arsitektur prototipe backend microservice.

TABLE II
Field Terlindungi dan Algoritma

Field	Jenis	Algoritma	Alasan
NIK	16 digit	FF1 radix 10	Menjaga format numerik
Phone	10 sampai 15 digit	FF1 radix 10	Menjaga format numerik
MRN	10 sampai 12 digit	FF1 radix 10	Menjaga format numerik
Name	Teks	AES-GCM	Kolom teks bebas
Address	Teks	AES-GCM	Kolom teks bebas
Diagnosis	Teks	AES-GCM	Kolom teks bebas

B. Skema data dan pilihan algoritma

Pemilihan algoritma mengikuti bentuk datanya. Tabel II merangkum pembagian tersebut. FF1 dipakai pada kolom numerik yang harus tetap berupa angka. AES-GCM dipakai pada kolom teks bebas. Blind index hanya dipakai untuk NIK karena pencarian utama pada prototipe ini memang pencarian berdasarkan NIK.

Mode AES-GCM-only mengenkripsi semua kolom sensitif dengan AES-GCM. Mode ini berguna sebagai pembanding karena datanya terlindungi, tetapi format numeriknya hilang. Mode plaintext dipakai sebagai baseline performa dan ukuran penyimpanan.

C. Pipeline create, read, search, dan rewrap

Pada mode FF1-envelope, operasi create dimulai dari pembuatan ID record. Main service lalu meminta DEK ke KMS, menurunkan subkey, mengenkripsi kolom numerik dengan FF1, mengenkripsi kolom teks dengan AES-GCM, menghitung blind index, dan membuat Record MAC. Setelah itu,

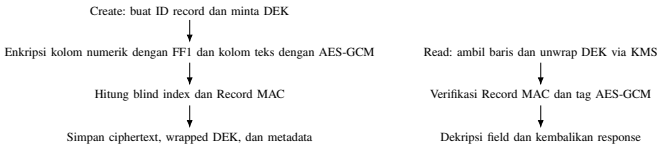

Fig. 2. Pipeline create dan read pada mode FF1-envelope.

TABLE III
PEMERIKSAAN PRESERVASI FORMAT FF1

Field	Plaintext	Ciphertext FF1	Terjaga
NIK	16 digit	16 digit	Ya
Phone	10 sampai 15 digit	10 sampai 15 digit	Ya
MRN	10 sampai 12 digit	10 sampai 12 digit	Ya

ciphertext dan metadata disimpan ke main database. Untuk read, urutannya dibalik: baris dibaca, DEK dibuka lewat KMS, MAC dan tag diverifikasi, lalu field didekripsi. Gambar 2 merangkum alur ini.

Search memakai blind index. Main service menormalisasi NIK input, menghitung HMAC blind index, lalu melakukan query ke kolom blind index. Jika baris ditemukan, sistem tetap menjalankan jalur read biasa. Dengan cara ini, sistem tidak perlu decrypt-scan seluruh record.

Rewrap dimulai dari rotasi KEK. KMS membuat KEK aktif baru dan menandai versi lama sebagai retiring. Main service kemudian memproses baris secara batch. Untuk setiap baris, KMS membungkus ulang DEK dari KEK lama ke KEK baru. Kolom PII terenkripsi tetap sama.

D. Lingkungan implementasi

Prototipe memakai Python FastAPI, PostgreSQL, Docker Compose, k6, dan script analisis Python. Repository memuat script benchmark, simulasi serangan, query penyimpanan SQL, dan figure hasil eksperimen. Plaintext DEK hanya dipakai sementara saat operasi kriptografi berlangsung. Prototype Python ini tidak mengklaim secure memory zeroization.

V. HASIL DAN ANALISIS

A. Validasi FF1 dan preservasi format

Implementasi FF1 diuji dengan official sample vector. Setelah itu, pengujian dilanjutkan ke contoh NIK 16 digit. Hasilnya tetap 16 digit, tetap numerik, dan dapat didekripsi kembali. Saat NIK yang sama dimasukkan pada ID record yang berbeda, ciphertext FF1 yang muncul juga berbeda karena tweak memuat konteks record.

Hasil pada Tabel III sesuai dengan tujuan awal FF1, yaitu menjaga bentuk angka. Namun, ini baru menyelesaikan masalah format. Integritas tetap ditangani oleh Record MAC agar perubahan ciphertext FF1 tidak lolos tanpa terdeteksi.

B. Performa create

Hasil create pada Tabel IV cukup sesuai dugaan. Plaintext paling cepat karena tidak melakukan enkripsi. AES-GCM-only mulai menambah biaya dari generate DEK, pembungkusan

TABLE IV
PERFORMA CREATE RERATA TIGA RUN

Mode	p50	p95	p99	RPS	Error
Plaintext	46.38	80.54	107.98	199.12	0
AES-GCM Only	85.47	126.84	159.19	113.51	0
FF1-Envelope	112.76	183.89	223.72	84.57	0

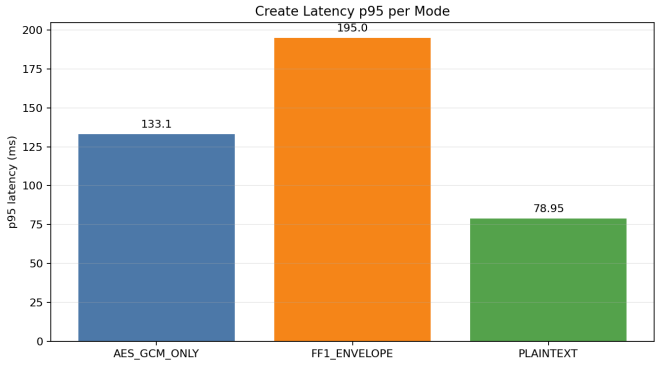

Fig. 3. Perbandingan p95 latency pada operasi create.

kunci, dan AES-GCM untuk semua kolom sensitif. FF1-envelope menjadi yang paling lambat karena jalurnya lebih panjang: generate DEK, wrapping, FF1, AES-GCM, blind index, dan Record MAC.

P95 latency naik dari 80,54 ms pada plaintext menjadi 126,84 ms pada AES-GCM-only dan 183,89 ms pada FF1-envelope. Throughput turun dari 199,12 RPS menjadi 113,51 RPS dan 84,57 RPS. Error tetap 0 pada semua mode, jadi penurunan ini lebih tepat dibaca sebagai biaya kriptografi dan metadata, bukan kegagalan permintaan.

Gambar 3 dan Gambar 4 memperlihatkan pola yang sama. FF1-envelope memang membayar biaya paling besar pada create, tetapi biaya itu datang dari fitur yang memang ditambahkan: preservasi format, pencarian NIK, isolasi kunci, dan deteksi manipulasi untuk kolom FF1.

C. Performa read dan search

Tabel V memperlihatkan perbedaan biaya antaroperasi pada mode FF1-envelope. Create paling mahal karena membuat rekam data baru dan menjalankan seluruh pipeline. Read lebih ringan karena hanya melakukan unwrap, verifikasi, dan dekripsi. Search sedikit lebih cepat lagi karena blind index membantu menemukan baris tanpa memindai dan mendekripsi semua data.

Di sini blind index terlihat berguna. Tanpa blind index, service harus mencoba banyak baris untuk menemukan satu NIK. Dengan blind index, basis data langsung menemukan kandidat baris. Konsekuensinya tetap ada: NIK yang sama akan menghasilkan blind index yang sama.

D. Overhead penyimpanan

Penyimpanan memberi cerita yang agak berbeda dari performa. Pada Tabel VI, AES-GCM-only justru paling besar

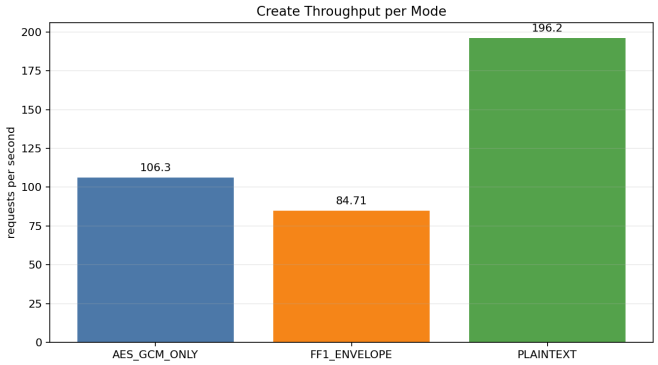

Fig. 4. Perbandingan throughput pada operasi create.

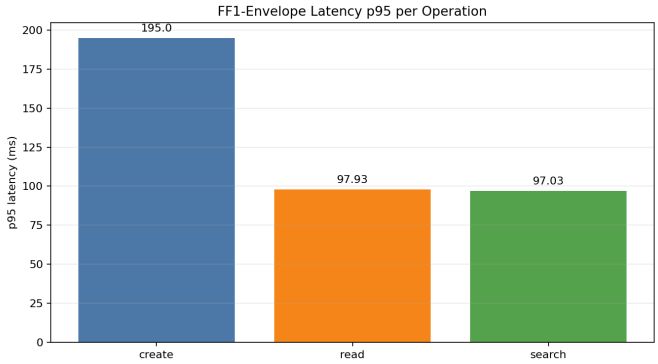

Fig. 5. p95 latency untuk create, read, dan search pada FF1-envelope.

TABLE V
PERFORMA OPERASI FF1-ENVELOPE

Operasi	p50	p95	p99	RPS
Create	112.76	183.89	223.72	84.57
Read	73.51	103.31	126.27	133.44
Search	69.80	97.20	118.51	140.14

TABLE VI
OVERHEAD PENYIMPANAN

Mode	Ukuran Baris Rata-rata	Overhead
Plaintext	191.92 B	0.00%
AES-GCM Only	631.20 B	228.89%
FF1-Envelope	551.20 B	187.20%

karena setiap kolom sensitif membutuhkan ciphertext, nonce, dan tag. FF1-envelope tetap menambah metadata, tetapi kolom numeriknya tidak membengkak sebesar ciphertext AES-GCM.

Baris FF1-envelope tetap menyimpan wrapped DEK, key version, blind index, Record MAC, dan metadata AES-GCM untuk kolom teks. Namun, kolom numerik tetap berupa string digit. Bagian inilah yang membuat overhead FF1-envelope lebih rendah daripada AES-GCM-only pada skema ini.

E. Rewrap KEK

Uji rewrap dijalankan pada tiga ukuran data. Seperti terlihat pada Tabel VII, 1.000, 5.000, dan 10.000 record berhasil diproses tanpa error. Total waktunya naik mengikuti jumlah record, sementara waktu rata-rata per record berada di sekitar 12 sampai 14 ms.

Rewrap hanya mengubah wrapped DEK dan metadata kunci. Payload PII tidak disentuh. Ini bagian yang membuat envelope encryption menarik untuk siklus hidup kunci, walaupun sistem produksi tetap perlu job scheduling, retry, rate limit, monitoring, dan integrasi managed KMS atau HSM.

F. Evaluasi keamanan

Tabel VIII merangkum simulasi serangan. Keenam skenario lulus dalam model ancaman yang didefinisikan.

Uji dump basis data tidak menemukan plaintext PII pada basis data utama. Pertukaran ciphertext ditolak karena AAD AES-GCM mengikat data ke konteks record dan kolom. Manipulasi FF1 juga ditolak karena Record MAC berubah setelah ciphertext numerik dimodifikasi. Uji dynamic tweak memperlihatkan bahwa ciphertext FF1 untuk NIK yang sama dapat berbeda pada record yang berbeda, sementara blind index tetap sama agar search masih bisa berjalan. Pada uji penghancuran KEK, record lama gagal didekripsi. Perilaku ini sesuai dengan prinsip fail-closed.

G. Pemetaan standar

Tabel IX dipakai untuk menunjukkan dasar rujukan desain. Isinya bukan klaim sertifikasi. Pemetaan ini hanya menjelaskan bahwa pilihan algoritma dan alur siklus hidup kunci tidak dibuat asal-asalan.

FF1 dirujukkan ke NIST SP 800-38G, AES-GCM ke NIST SP 800-38D, pembungkusan kunci ke NIST SP 800-38F, dan siklus hidup kunci ke NIST SP 800-57 Part 1 [1]–[4]. ISO/IEC 27701:2025 dipakai sebagai kerangka privasi untuk pemrosesan PII [5]. Bahasa yang dipakai sengaja dibatasi pada pemetaan teknis, bukan kepatuhan formal.

VI. DISKUSI

A. Kompromi Performa dan Keamanan

Hasil eksperimen menunjukkan biaya yang cukup nyata. FF1-envelope lebih lambat daripada plaintext dan AES-GCM-only pada operasi create. Ukuran barisnya juga lebih besar daripada plaintext. Biaya ini muncul karena mode tersebut melakukan pekerjaan tambahan: membuat DEK, membungkus DEK, menjalankan FF1, menjalankan AES-GCM, menghitung blind index, dan membuat Record MAC.

Biaya tersebut masih punya alasan. FF1 menjaga format kolom numerik. AES-GCM melindungi kolom teks bebas dan menolak ciphertext yang dipindahkan ke konteks lain. Envelope encryption membuat rotasi KEK lebih ringan karena payload PII tidak perlu dienkripsi ulang. Blind index membuat pencarian NIK tetap praktis. Record MAC menutup kekurangan FF1 dari sisi integritas.

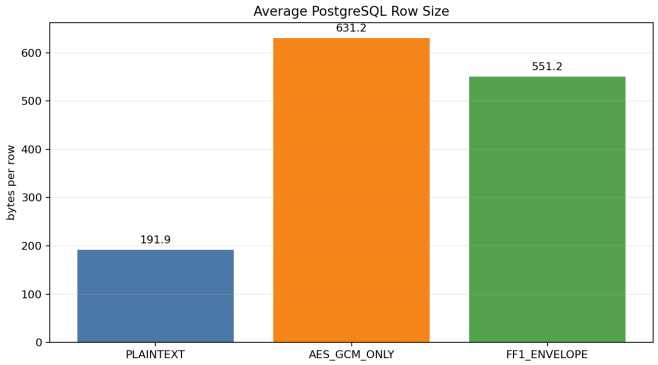

Fig. 6. Overhead penyimpanan pada tiga mode penyimpanan.

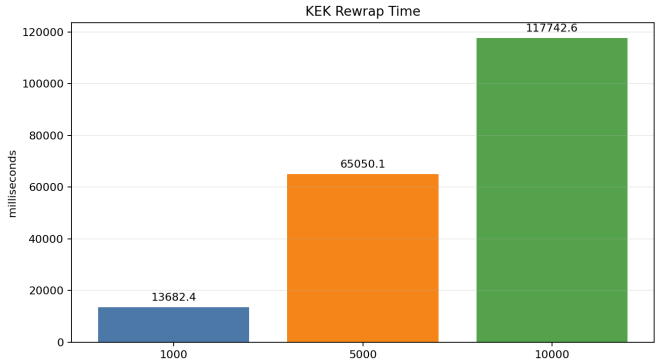

Fig. 7. Waktu rewrap untuk 1.000, 5.000, dan 10.000 record.

TABLE VII
BIAYA REWRAP KEK

Record	Total Time (s)	Avg./Record (ms)	Error
1,000	13.68	13.68	0
5,000	65.05	13.01	0
10,000	117.74	11.77	0

TABLE VIII
EVALUASI KEAMANAN

Skenario	Hasil teramati	Status
Dump basis data	Plaintext PII tidak ditemukan pada raw DB/debug dump	PASS
Pertukaran ciphertext	Ciphertext AES-GCM hasil swap ditolak dengan status 400	PASS
Manipulasi FF1	Ciphertext FF1 yang diubah ditolak dengan status 400	PASS
Dynamic tweak dan blind index	NIK sama menghasilkan dua nilai FF1, tetapi satu nilai blind index	PASS
Blind index search	Pencarian NIK mengembalikan record terlindungi	PASS
Penghancuran KEK	Record dengan KEK yang dihancurkan tidak dapat didekripsi	PASS

TABLE IX
PEMETAAN STANDAR

Rujukan	Implementasi	Bukti evaluasi
NIST SP 800-38G	FF1 radix 10 untuk NIK, nomor telepon, dan MRN	Preservasi format dan vector test
NIST SP 800-38D	AES-GCM dengan AAD berbasis record dan kolom	Pertukaran ciphertext gagal
NIST SP 800-38F	AES-KW-256 untuk wrapped DEK	DB hanya menyimpan wrapped DEK
NIST SP 800-57 Part 1	KEK versioning, rotation, destroy, dan rewrap	Rewrap 1k, 5k, dan 10k record tanpa error
ISO/IEC 27701:2025	Field encryption dan controlled search	Search berjalan tanpa decrypt-scan

AES-GCM-only terlihat menarik dari sisi performa karena lebih ringan daripada FF1-envelope. Namun, mode itu mengubah bentuk kolom numerik. Plaintext tetap paling cepat dan paling kecil, tetapi tidak memberi perlindungan saat dump basis data terjadi. Jadi, pilihan mode bergantung pada prioritas sistem: performa saja, atau performa yang masih menerima kebutuhan format, search terbatas, dan siklus hidup kunci.

B. *Batas model ancaman*

Evaluasi ini sengaja dibatasi pada kompromi basis data. Jika penyerang sudah menguasai main service, KMS service, runtime memory, atau environment secret, asumsi makalah ini tidak lagi berlaku. Prototipe juga belum membahas side-channel, timing attack, SQL injection, atau access control produksi. KMS yang dipakai masih berupa service software untuk eksperimen, bukan HSM.

Blind index juga perlu dibaca dengan hati-hati. Ia membuat search bisa berjalan, tetapi membuka kebocoran kesamaan nilai. Jika dua baris memiliki NIK yang sama, blind index-nya juga sama. Untuk NIK, risiko ini relatif lebih kecil karena nilainya semestinya unik. Untuk field yang sering berulang, seperti label diagnosis, pendekatan yang sama akan jauh lebih sensitif.

C. *Implikasi desain*

Dari eksperimen ini, pembagian perannya cukup jelas. FF1 cocok untuk kolom numerik berformat tetap. AES-GCM cocok untuk kolom teks bebas. AAD perlu mengikat ciphertext ke konteks record dan field. Record MAC diperlukan karena FF1 tidak mengautentikasi data. Envelope encryption membantu saat sistem membutuhkan rotasi kunci tanpa mengenkripsi ulang semua kolom terlindungi.

Untuk produksi, desain ini jelas belum selesai. Sistem nyata sebaiknya memakai managed KMS atau HSM, autentikasi antarlayanan, logging operasional, retry control, dan benchmark pada skala yang lebih besar. Pada skala eksperimen kecil sampai menengah di makalah ini, prototipe sudah cukup untuk menunjukkan trade-off kriptografi yang ingin diuji.

VII. KESIMPULAN

Makalah ini menguji prototipe backend microservice yang menggabungkan FF1 format-preserving encryption dan envelope encryption untuk PII pasien sintetis. FF1 berhasil menjaga format NIK, nomor telepon, dan nomor rekam medis. AES-GCM melindungi kolom teks bebas, sedangkan envelope encryption memisahkan DEK dan KEK agar rotasi KEK dapat dilakukan melalui rewrap. Hasilnya tidak gratis: FF1-envelope

menambah latency dan overhead penyimpanan dibandingkan plaintext. Namun, mode ini juga memberi preservasi format, search terbatas, deteksi manipulasi, dan dukungan siklus hidup kunci. Rewrap 10.000 record selesai dalam 117,74 detik tanpa error, dan keenam simulasi keamanan lulus dalam model ancaman kompromi basis data. Dengan batasan tersebut, pendekatan ini layak dipertimbangkan untuk sistem yang perlu melindungi PII tanpa merusak format numerik yang sudah dipakai aplikasi.

VIDEO LINK AT YOUTUBE

Link Video Penjelasan (dapat diklik)

GITHUB

Github Repository (dapat diklik)

REFERENCES

[1] M. Dworkin, "Recommendation for Block Cipher Modes of Operation: Methods for Format-Preserving Encryption," NIST Special Publication 800-38G, National Institute of Standards and Technology, 2016.

[2] M. Dworkin, "Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC," NIST Special Publication 800-38D, National Institute of Standards and Technology, 2007.

[3] M. Dworkin, "Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping," NIST Special Publication 800-38F, National Institute of Standards and Technology, 2012.

[4] E. Barker, "Recommendation for Key Management: Part 1 - General," NIST Special Publication 800-57 Part 1 Revision 5, National Institute of Standards and Technology, 2020.

[5] ISO/IEC, "ISO/IEC 27701:2025 Information security, cybersecurity and privacy protection - Privacy information management systems - Requirements and guidance," International Organization for Standardization, 2025.

[6] Grafana Labs, "k6 Documentation." [Online]. Available: https://grafana.com/docs/k6/

[7] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. Boca Raton, FL, USA: CRC Press, 1996.

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

Muhammad Omar Berliansyah
18223055