Pengembangan Aplikasi Manajemen Kata Sandi Terdistribusi Dengan Skema Pembagian Data Rahasia

Edbert Eddyson Gunawan, 13522039[1,2]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13522039@std.stei.itb.ac.id, [2]edbert.e.gunawan@protonmail.com

***Abstrak*—Perkembangan ekosistem digital yang pesat meningkatkan kebutuhan akan sistem manajemen identitas dan kontrol akses yang aman. Kata sandi masih menjadi mekanisme autentikasi utama, namun penggunaan ulang kata sandi dan lemahnya pengelolaan kredensial menimbulkan risiko keamanan yang signifikan. Aplikasi manajemen kata sandi hadir sebagai solusi, tetapi sebagian besar mengadopsi arsitektur terpusat yang rentan terhadap Single Point of Failure (SPoF). Makalah ini mengusulkan dan mengimplementasikan sebuah sistem manajemen kata sandi dengan arsitektur terdistribusi yang mengombinasikan algoritma AES-128 untuk enkripsi data dan Shamir's Secret Sharing (SSS) untuk distribusi kunci kriptografis. Kunci enkripsi utama dibagi menggunakan skema ambang batas (2,3) dan didistribusikan ke server, perangkat pengguna, serta media pemulihan, sehingga proses dekripsi membutuhkan minimal dua entitas yang berbeda. Hasil pengujian menunjukkan bahwa sistem mampu meningkatkan keamanan dengan menghilangkan SPoF, sekaligus menjaga ketersediaan data melalui mekanisme pemulihan berbasis Interpolasi Lagrange. Selain itu, seluruh proses kriptografi dilakukan di sisi klien, mendukung prinsip zero-knowledge dan memastikan kinerja yang efisien pada perangkat dengan sumber daya terbatas.**

***Kata kunci*— Shamir's Secret Sharing, Manajemen Kata Sandi, Kriptografi, AES-128, Sistem Terdistribusi, Keamanan Informasi**

I. PENDAHULUAN

Dengan semakin berkembangnya ekosistem digital dan internet, manajemen identitas dan kontrol akses menjadi elemen krusial dalam keamanan informasi. Salah satu mekanisme autentikasi yang paling mendasar adalah kata sandi (*password*). Kata sandi digunakan untuk melindungi privasi data, aset finansial, dan identitas digital pengguna. Namun, meningkatnya jumlah layanan daring (*online services*) mendorong pengguna memiliki banyak akun. Sering kali pengguna menggunakan kata sandi yang lemah atau berulang. Untuk mengatasi hal ini, penggunaan aplikasi pengelola kata sandi (*password manager*) menjadi solusi standar untuk mengelola kredensial.

Mayoritas aplikasi manajemen kata sandi komersial saat ini mengadopsi arsitektur terpusat (*centralized architecture*), yaitu satu entitas penyedia layanan menyimpan seluruh basis data kredensial pengguna. Meskipun data tersebut dienkripsi, pendekatan ini menciptakan *Single Point of Failure* (SPoF). Jika basis data server pusat diretas, penyerang memiliki akses ke seluruh *ciphertext* pengguna. Sebaliknya, jika pengguna menyimpan data hanya di perangkat lokal (laptop/HP) lalu perangkat tersebut hilang atau rusak, seluruh akses data hilang selamanya.

Oleh karena itu, diperlukan mekanisme yang dapat memecah risiko penyimpanan kunci tanpa mengorbankan ketersediaan data. Mekanisme Shamir's Secret Sharing (SSS) menjadi sangat memungkinkan sebuah rahasia (kunci enkripsi utama) dibagi menjadi n bagian (*shares*), dengan k bagian yang diperlukan untuk merekonstruksi rahasia tersebut.

Penerapan SSS dalam sistem ini bertujuan untuk mencapai dua properti keamanan sekaligus yaitu keamanan ganda (*double verification*) dan ketersediaan data (*availability/fault tolerance*). Dengan menetapkan suatu *threshold* proses *login* atau dekripsi tidak dapat dilakukan hanya dengan satu *share*. Artinya, jika seorang peretas berhasil membobol server dan mencuri *share* yang ada di sana, data pengguna tetap aman karena peretas tidak memiliki *share* kedua yang tersimpan diperangkat pengguna. Hal ini memaksa penyerang untuk mendapatkan akses ke 2 entitas terpisah secara bersamaan. Berbeda dengan sistem *multi-signature* yang mewajibkan seluruh kunci hadir, SSS menawarkan fleksibilitas. Dengan skema jika salah satu *share* hilang (misalnya server mati atau perangkat hilang), pengguna masih dapat memulihkan kunci menggunakan kombinasi dua *share* lainnya (misalnya perangkat + *recovery key*). Hal ini menjamin layanan tetap dapat digunakan (*available*) meskipun salah satu komponen mengalami kegagalan.

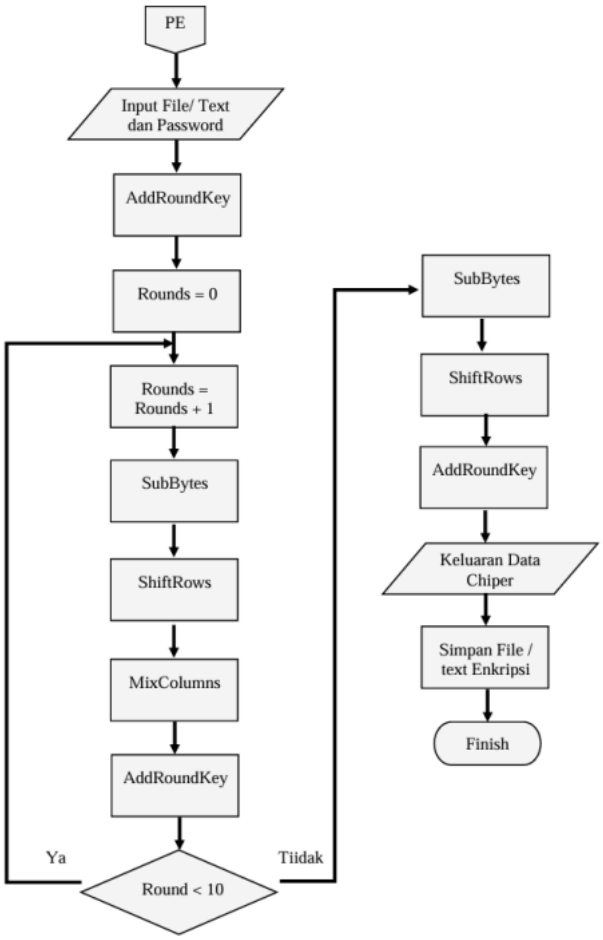
Berangkat dari masalah arsitektur terpusat dan kebutuhan akan sistem yang *fault* tolerance namun aman, makalah ini mengembangkan sebuah sistem manajemen kata sandi dengan arsitektur terdistribusi menggunakan Shamir's Secret Sharing.

II. LANDASAN TEORI

A. Algoritma Enkripsi AES

Algoritma Advanced Encryption Standard (AES) merupakan salah satu algoritma kriptografi modern yang bersifat simetri dan termasuk dalam kategori *block cipher*, dengan representasi data berbasis heksadesimal dalam

proses komputasinya. AES mendukung tiga variasi panjang kunci, yaitu 128 bit, 192 bit, dan 256 bit. Dalam praktiknya, ukuran kunci yang paling umum digunakan adalah 128 bit (AES-128).

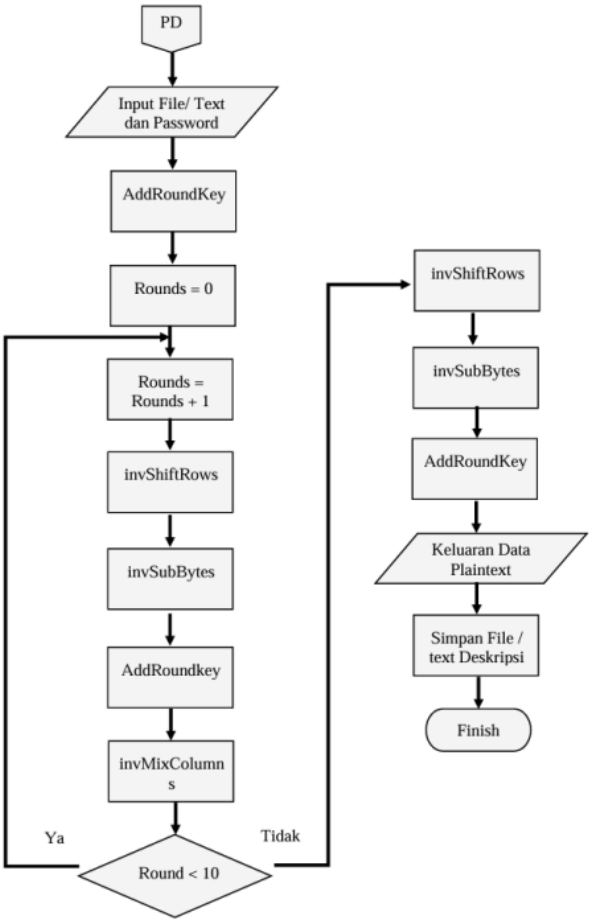


Gambar 1. Proses enkripsi AES (Sumber: Pribadi)

Proses enkripsi dan dekripsi pada algoritma AES terdiri atas dua tahap utama. Tahap pertama adalah pembangkitan sub-kunci (key schedule) yang berasal dari kunci utama, sedangkan tahap kedua adalah proses enkripsi atau dekripsi data menggunakan sub-kunci tersebut. Pada AES-128, jumlah sub-kunci yang dihasilkan adalah 10 sub-kunci, dengan masing-masing sub-kunci memiliki panjang 128 bit.

Proses pembangkitan sub-kunci diawali dengan mempartisi kunci utama menjadi empat kolom (word). Selanjutnya, setiap word diproses melalui empat operasi berurutan, yaitu RotWord, SubWord, operasi XOR dengan konstanta Rcon, dan operasi XOR dengan kolom sebelumnya. Rangkaian operasi ini diulangi secara iteratif untuk menghasilkan seluruh sub-kunci yang diperlukan.

Dalam proses enkripsi maupun dekripsi, AES-128 menggunakan sepuluh sub-kunci yang telah dibangkitkan sebelumnya serta data masukan berupa *plaintext* atau *ciphertext* yang telah dipartisi menjadi blok-blok berukuran 128 bit. Gambar 1 menunjukkan flowchart proses enkripsi pada algoritma AES-128, sedangkan Gambar 2 menggambarkan flowchart proses dekripsi pada algoritma AES-128.



Gambar 2.Proses dekripsi AES (Sumber: Pribadi)

B. Sistem Terdistribusi

Teorema CAP, yang diperkenalkan oleh Eric Brewer, menyatakan bahwa dalam sebuah sistem komputer terdistribusi, mustahil untuk menjamin ketiga properti berikut secara bersamaan dan sempurna. Sistem harus memilih maksimal dua dari tiga properti ini yakni *consistency*, *availability*, dan *partition tolerance*. *Consistency* didefinisikan untuk setiap pembacaan data (*read*) akan mendapatkan data yang paling baru. Artinya, semua node dalam sistem melihat data yang sama pada waktu yang sama. Sementara *availability* berarti setiap permintaan (*request*) yang diterima oleh *node* yang tidak *error* harus menghasilkan respons *non-error*, namun tanpa jaminan bahwa respons tersebut memuat data versi terbaru. Yang diutamakan adalah sistem selalu "hidup" dan bisa melayani pengguna. *Partition tolerance* didefinisikan sebagai sistem terus beroperasi meskipun terjadi kegagalan jaringan yang menyebabkan pesan antar *node* hilang atau tertunda (partisi jaringan).

Dalam konteks sistem aplikasi terdistribusi manajemen kata sandi, *Partition Tolerance* (P) adalah sesuatu yang wajib ada (tidak bisa dihindari karena jaringan internet tidak bisa dijamin 100% sempurna). Maka pilihannya tinggal antara CP (Konsistensi + Toleransi Partisi) atau AP (Ketersediaan + Toleransi Partisi).

Sistem ini memilih *Partition Tolerance* (P) karena kunci dibagi dan disimpan di beberapa lokasi (*server*, perangkat lokal, dan *recovery key*) yang masing-masing dapat mengalami partisi, seperti koneksi internet terputus, *server downtime*, atau perangkat pengguna hilang/rusak, sehingga sistem harus tetap berfungsi meskipun salah satu entitas tidak dapat diakses; tanpa toleransi partisi, kegagalan satu komponen akan menyebabkan sistem gagal total. Selain itu, sistem juga mengutamakan *Availability* (A) karena tujuan utama adalah memastikan pengguna tetap dapat mengakses data penting melalui *password manager*, sehingga ketika satu *node* gagal (misalnya server mati atau diretas), kata sandi tetap dapat direkonstruksi dari *share* lain seperti perangkat dan *recovery key*. Jika sistem lebih memilih Konsistensi (CP), maka pada saat server tidak tersedia akses akan ditolak demi mencegah inkonsistensi, namun hal ini bertentangan dengan tujuan utama sistem yang menekankan ketersediaan layanan meskipun terjadi kegagalan komponen.

C. Skema Pembagian Rahasia Shamir

Shamir's Secret Sharing (SSS), yang diperkenalkan oleh Adi Shamir pada tahun 1979, menggunakan konsep *threshold cryptography*. SSS didefinisikan sebagai skema (k, n), sebuah rahasia S (kunci enkripsi utama basis data kata sandi) dibagi menjadi n bagian independen yang disebut *shares* (S_1, S_2, ... , S_n).Properti fundamental dari skema ini adalah ambang batas k. Rahasia S hanya dapat direkonstruksi jika dan hanya jika sembarang sub-himpunan yang terdiri dari minimal k *share* digabungkan. Hal ini menjamin *robustness*: Sistem tetap dapat berfungsi dan data dapat dipulihkan meskipun $n - k$ *share* hilang atau rusak (misalnya akibat kegagalan perangkat keras), dan *perfect Secrecy*: pengetahuan terhadap $k - 1$ *share* tidak memberikan informasi apa pun mengenai rahasia *S*, baik secara komputasional maupun statistik.

Mekanisme pembangkitan *share* didasarkan pada polinomial acak $q(x)$ berderajat $k - 1$ yang dikonstruksi dalam bilangan riil atau modular:

$$q(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{k-1}\, x^{k-1} \ (mod\ p)$$

Dalam polinomial ini, elemen kunci adalah a_0, yang ditetapkan sebagai nilai rahasia S. Koefisien lainnya $(a_1, \ldots, a_{k-1})$ dipilih secara acak dari distribusi seragam. Setiap partisipan atau lokasi penyimpanan kemudian diberikan pasangan koordinat (x_i, y_i), dengan $y_i = q(x_i)$. Karena koefisien-koefisien tersebut acak, setiap *share* individu tampak sebagai data acak (*noise*) yang tidak memiliki korelasi yang dapat diamati dengan rahasia aslinya.

D. Interpolasi Lagrange

Proses pemulihan kunci rahasia dari pecahan-pecahan *share* yang tersebar dilakukan menggunakan metode *Interpolasi Lagrange*. Secara teoritis, teorema dasar aljabar menyatakan bahwa k titik koordinat unik (x_i, y_i) cukup untuk mendefinisikan secara tunggal sebuah polinomial berderajat $k - 1$. Alih-alih menyelesaikan sistem persamaan linear yang kompleks (seperti metode eliminasi Gauss) untuk mencari seluruh koefisien polinomial, *Interpolasi Lagrange* memungkinkan penghitungan langsung nilai fungsi pada titik tertentu. Karena rahasia *S* tersimpan pada konstanta a_0 atau $q(0)$, algoritma hanya perlu mengevaluasi polinomial Lagrange pada $x = 0$. Persamaan rekonstruksi Lagrange didefinisikan sebagai jumlahan berbobot dari share yang tersedia:

$$S = f(0) = \sum_{j=1}^{k} y_j \cdot L_j(0) \ mod\ p$$

Dengan $L_j(x)$ adalah basis polinomial Lagrange:

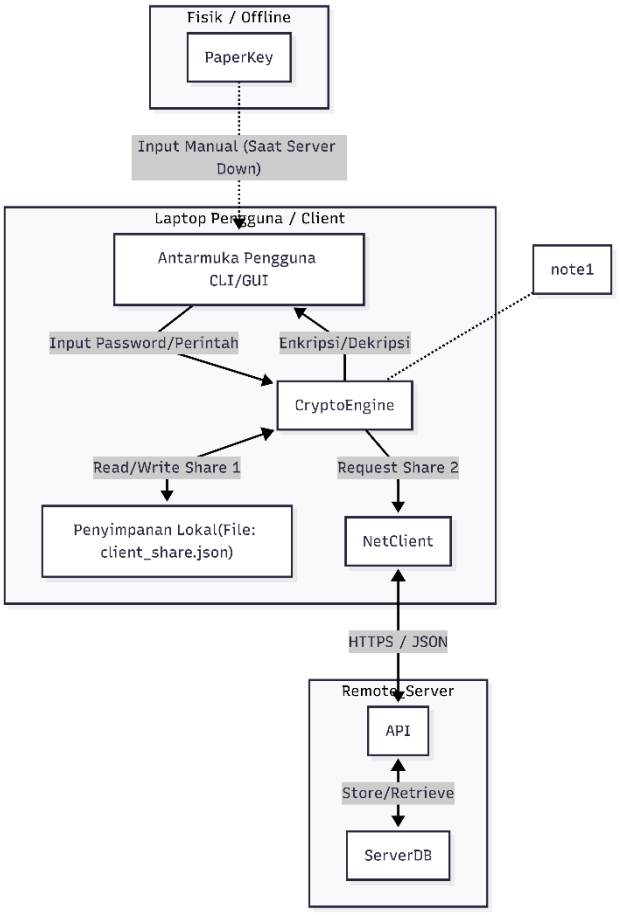
$$L_j(0) = \prod_{i=1, i \neq j}^{k} \frac{0 - x_i}{x_j - x_i}$$

Secara komputasi, operasi ini sangat ringan, memungkinkan dekripsi cepat pada perangkat pengguna dengan sumber daya terbatas (seperti ponsel atau laptop) untuk melakukan rekonstruksi kunci secara instan tanpa membebani prosesor. Hal ini mendukung arsitektur terdistribusi di mana proses dekripsi terjadi di sisi klien (*client-side*), menjamin bahwa kunci rahasia yang utuh tidak pernah terekspos ke jaringan atau server.

III. IMPLEMENTASI

A. Arsitektur Aplikasi

Berdasarkan permasalahan yang sudah disampaikan, berikut rancangan arsitektur dari sistem manajemen kata sandi yang dibangun.



Gambar 3. Arsitektur aplikasi (Sumber: Pribadi)

Arsitektur Inti: Skema Ambang Batas (Threshold Scheme) Sistem ini dirancang berdasarkan prinsip kriptografi Shamir's Secret Sharing dengan konfigurasi (2,3). Arsitektur ini memecah Master Key (kunci utama) menjadi tiga bagian terpisah yang disebut shares, yang didistribusikan ke tiga domain keamanan berbeda: perangkat pengguna (Laptop), infrastruktur awan (Server VPS), dan media tambahan (Offline/Kertas). Agar sistem dapat mendekripsi brankas kata sandi (vault), minimal dua dari tiga shares tersebut harus digabungkan kembali. Desain ini menghilangkan Single Point of Failure (SPoF); peretas yang berhasil mengkompromikan salah satu lokasi (misalnya hanya server atau hanya laptop) tidak akan memiliki informasi yang cukup untuk merekonstruksi kunci enkripsi.

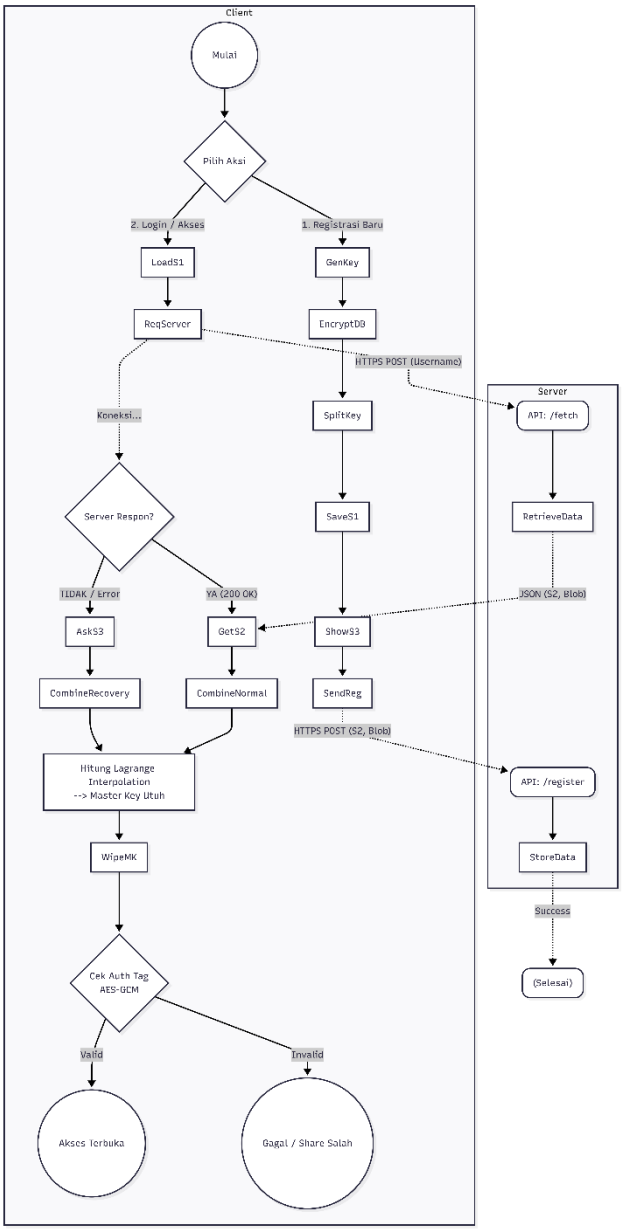
Komponen Klien: Pusat Operasi Kriptografi Sisi klien (Client Device) bertindak sebagai satu-satunya lingkungan tepercaya (*trusted environment*) agar data bisa didekripsi dengan aman. Di dalamnya terdapat CryptoEngine, komponen logis yang bertanggung jawab untuk menjalankan interpolasi Lagrange guna menyatukan *shares* dan membentuk Master Key secara sementara di memori (RAM). Klien menyimpan Share 1 secara lokal di dalam disk (LocalStore). Saat pengguna ingin masuk (login), klien akan terlebih dahulu membaca Share 1 ini. Karena Share 1 saja tidak cukup untuk membuka brankas, klien menggunakan modul NetClient untuk menghubungi server dan meminta *share* pelengkap.

Komponen Server: Penyimpanan Tanpa Pengetahuan (Zero-Knowledge) Server jarak jauh (Remote Server) berfungsi sebagai penyimpanan buta. Server ini menyimpan basis data kata sandi yang terenkripsi (*Encrypted Blob*) dan *Share* 2. Server berperan untuk menjamin ketersediaan data dan sinkronisasi antar-perangkat. Secara desain, server tidak pernah menerima Master Key utuh ataupun Share 1 dan Share 3. Hal ini memastikan penyedia layanan (atau penyerang yang meretas server) tidak memiliki kemampuan matematis untuk melihat isi data pengguna, karena mereka hanya memegang satu bagian dari kunci, sementara dibutuhkan dua bagian untuk dekripsi.

Mekanisme Pemulihan dan Toleransi Kesalahan Diagram ini juga mengilustrasikan mekanisme ketahanan sistem (resilience) melalui komponen Offline User. Pengguna memegang Share 3 dalam bentuk fisik (Paper Key) atau dalam bentuk lainnya. Jalur ini diaktifkan hanya saat kondisi darurat, misalnya ketika server mengalami gangguan (*down*) atau tidak dapat dihubungi. Dalam skenario ini, klien akan mendeteksi kegagalan koneksi dan meminta pengguna memasukkan Share 3 secara manual melalui antarmuka (UI). CryptoEngine kemudian akan menggabungkan Share 1 (dari penyimpanan lokal) dengan Share 3 (input manual) untuk memenuhi ambang batas 2 shares, sehingga brankas tetap dapat dibuka (available) meskipun infrastruktur server sedang lumpuh.

B. Alur Kerja Aplikasi

Aplikasi dirancang dengan arsitektur *client-server* yang mengimplementasikan skema Shamir's Secret Sharing (SSS) untuk manajemen kunci kriptografi. Alur kerja sistem dibagi menjadi dua proses utama: (A) Registrasi Kunci Baru dan (B) Mekanisme Otentikasi dan Pemulihan Akses. Proses Registrasi dan Distribusi ShareProses registrasi bertujuan untuk mengamankan basis data lokal dan mendistribusikan pecahan kunci (*key shares*) ke berbagai lokasi penyimpanan untuk menghindari titik kegagalan tunggal (*Single Point of Failure*). Pada tahap Inisiasi dan Enkripsi, pengguna memulai aplikasi dan memilih opsi "Registrasi Baru". Sistem membangkitkan *Master Key* secara acak yang kemudian digunakan untuk mengenkripsi basis data lokal menggunakan algoritma simetris (tersirat AES-GCM berdasarkan proses dekripsi). Pada tahap Kunci (*Key Splitting*) *Master Key* dipecah menggunakan algoritma SSS menjadi tiga bagian (shares) dengan ambang batas (k=2), yang berarti dibutuhkan minimal dua *share* untuk merekonstruksi kunci. Penyimpanan Share 1 Disimpan secara lokal di perangkat pengguna (*Client Storage*). Share 2 Dikirim melalui protokol HTTPS POST ke Server untuk disimpan dalam basis data jarak jauh. Server menerima data ini melalui endpoint API /register. Share 3 Ditampilkan kepada pengguna sebagai *Recovery Share* (kunci darurat) yang harus disimpan secara manual/*offline*.



Gambar 4. Alur logika aplikasi

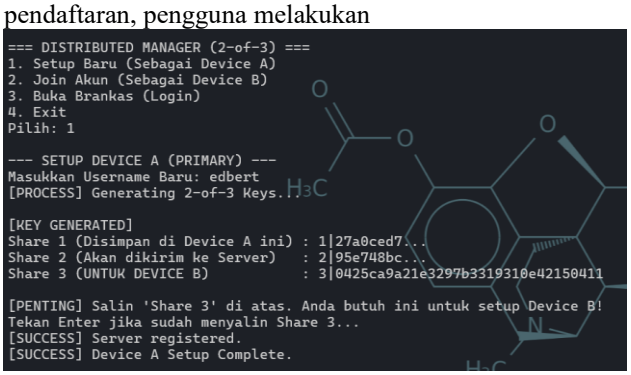
Mekanisme Otentikasi dan Akses DataProses login dirancang adaptif terhadap ketersediaan jaringan, memungkinkan sistem tetap berjalan meskipun server tidak dapat diakses (mekanisme *offline-first*). Pada tahap Pemuatan Share Lokal: Saat pengguna memilih "Login / Akses", sistem pertama-tama memuat S_1 dari penyimpanan lokal perangkat.Permintaan ke Server: Klien mengirimkan permintaan (*request*) ke API /fetch pada server untuk mengambil S_2 dan blob data terenkripsi.

Percabangan Logika Ketersediaan Server. Saat Kondisi Normal (Server Online), jika server merespons dengan status 200 OK, sistem mengunduh S_2. Proses rekonstruksi kunci dilakukan dengan menggabungkan kombinasi ($S_1 + S_2$). Saat Kondisi Kegagalan (Server Offline/Error), jika server tidak merespons atau terjadi kesalahan jaringan, sistem beralih ke mode pemulihan (*failover*). Aplikasi akan meminta pengguna memasukkan S_3 secara manual. Proses rekonstruksi kunci dilakukan menggunakan kombinasi ($S_1 + S_3$).

Rekonstruksi Kriptografi dan Validasi Setelah dua *share* berhasil dikumpulkan (baik melalui jalur normal maupun pemulihan), sistem menjalankan proses kriptografi inti: yaitu Interpolasi Lagrange. Sistem menghitung fungsi polinomial menggunakan metode Interpolasi Lagrange berdasarkan dua titik koordinat (*share*) yang didapat untuk menghasilkan kembali Master Key yang utuh. Segera setelah Master Key digunakan untuk proses dekripsi sementara, kunci tersebut dihapus dari memori (RAM) untuk mencegah kebocoran data (memory dump attacks). Tahap terakhir yaitu validasi dekripsi (AES-GCM): Master Key digunakan untuk mendekripsi data. Validitas kunci diverifikasi melalui pengecekan Authentication Tag pada algoritma AES-GCM. Jika tag valid, akses diberikan (Success). Jika tag invalid (indikasi share salah atau data korup), akses ditolak (Fail).

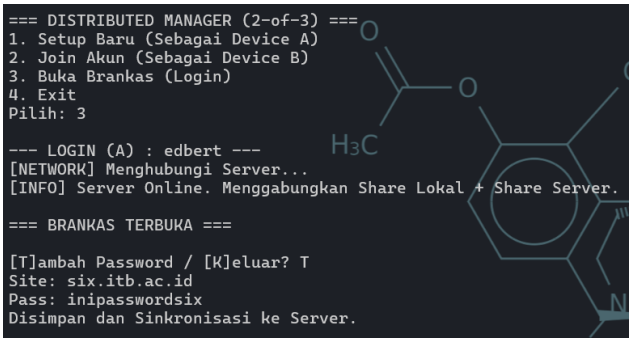
IV. PENGUJIAN

Dalam makalah ini, terdapat skenario pengujian yang akan dijalankan sebuah alur saat pengguna melakukan pendaftaran, pengguna melakukan



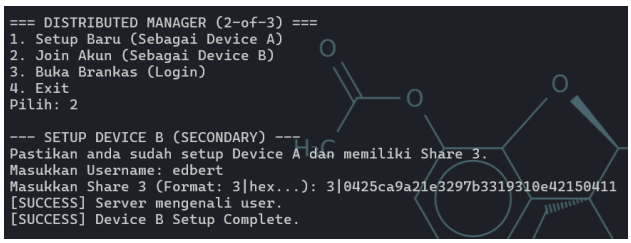
Gambar 5. Device A - pendaftaran

Pada Gambar 5. *Server* sudah berjalan, dan pengguna dengan *device A* dmelakukan pendaftaran nama pengguna. Setelah itu, SSS akan membangkitkan *shares*. *Shares* device 3 digunakan untuk *device B*.



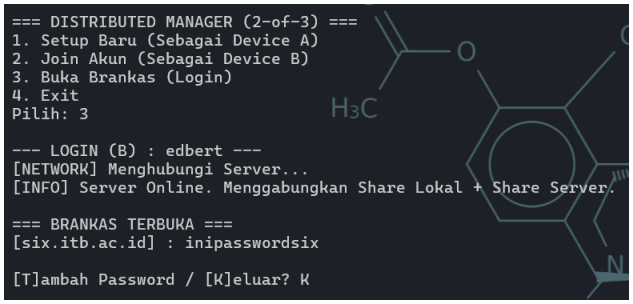
Gambar 6. Device A - menambah kata sandi

Pada Gambar 6. *Device A* menambah kata sandi baru pada aplikasi. Kata sandi ini kemudian dienkripsi saat disimpan.



Gambar 7. Device B - proses recovery

Pada Gambar 7. *Device A* diasumsikan mengalami kegagalan. Namun meskipun terjadi kegagalan terdapat *device B* yang dapat memasukkan *shares* baru sehingga dengan *shares* 1 pada server, berhasil merekonstruksi kembali.



Gambar 8. Device B - melihat daftar password

Pada Gambar 8. *Device B* dapat mengakses kembali kunci yang sudah disimpan sebelumnya.

V. KESIMPULAN

Pada makalah ini telah dikembangkan sebuah aplikasi manajemen kata sandi yang mengintegrasikan algoritma AES-128 untuk enkripsi data tersimpan dengan algoritma Shamir's Secret Sharing (SSS) sebagai mekanisme distribusi kunci kriptografis. Berdasarkan hasil perancangan dan pengujian, penerapan skema ambang batas (2,3) terbukti mampu menghilangkan risiko Single Point of Failure (SPoF) yang umum dijumpai pada arsitektur terpusat, karena kunci enkripsi dipecah menjadi tiga *share* yang tersebar pada server, perangkat pengguna, dan media pemulihan, sehingga kompromi terhadap satu entitas saja tidak cukup untuk mendekripsi data. Selain itu, sistem menunjukkan tingkat toleransi kesalahan dan ketersediaan yang tinggi, sejalan dengan prinsip Availability dan Partition Tolerance dalam Teorema CAP, di mana mekanisme pemulihan berbasis Interpolasi Lagrange memungkinkan rekonstruksi kunci dan pemulihan akses data meskipun terjadi kegagalan server atau kehilangan perangkat, selama minimal dua *share* masih tersedia. Dari sisi performa, penggunaan aritmatika *finite field* dan algoritma AES-128 memungkinkan proses enkripsi dan dekripsi dijalankan secara efisien di sisi klien, sekaligus menjaga privasi pengguna melalui prinsip zero-knowledge dan memastikan aplikasi tetap responsif pada perangkat dengan sumber daya terbatas.

VI. UCAPAN TERIMA KASIH

Segala puji dan rasa syukur penulis panjatkan ke hadirat Tuhan Yang Maha Esa atas berkat dan rahmat-Nya yang melimpah sehingga penulis dapat menyelesaikan makalah ini dengan baik. Penulis mengucapkan terima kasih yang sebesar-besarnya kepada Dr. Ir. Rinaldi, M.T., selaku dosen mata kuliah IF4020 Kriptografi (kelas K01), atas ilmu dan bimbingan yang telah diberikan sehingga sangat membantu dalam penyusunan makalah ini. Selain itu, penulis juga menyampaikan apresiasi dan terima kasih yang tulus kepada orang tua penulis atas dukungan dan motivasi yang tiada henti selama proses penyusunan makalah ini.

REFERENSI

[1] A. Shamir, "How to share a secret," Communications of the ACM, vol. 22, no. 11, hal. 612–613, Nov. 1979.

[2] C. Chen, M. Felix, S. Otremba, and N. Sykes, "Distributed and Secure Password Manager," MIT 6.857 Computer and Network Security Project, Cambridge, MA, USA, Spring 2021.

[3] E. A. Brewer, "Towards robust distributed systems," in *Proc. 19th Annu. ACM Symp. Principles of Distributed Computing (PODC)*, Portland, OR, USA, 2000, pp. 7.

[4] J. Daemen dan V. Rijmen, *The Design of Rijndael: AES - The Advanced Encryption Standard*. Berlin, Heidelberg: Springer-Verlag, 2002.

[5] National Institute of Standards and Technology (NIST), "Advanced Encryption Standard (AES)," U.S. Department of Commerce, Washington, D.C., FIPS PUB 197, Nov. 2001.

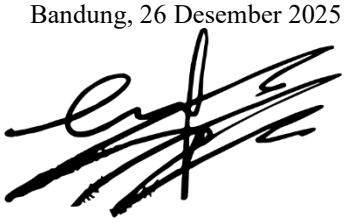
[6] R. Munir, "Skema Pembagian Data Rahasia (Secret Sharing Scheme)," Bahan Kuliah IF4021 Kriptografi, Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung, 2025.W.-K. Chen, *Linear Networks and Systems* (Book style). Belmont, CA: Wadsworth, 1993, pp. 123–135.

[7] W. Stallings, *Cryptography and Network Security: Principles and Practice*, edisi ke-7. Hoboken, NJ: Pearson, 2017.

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, 26 Desember 2025



Edbert Eddyson Gunawan, 13522039