Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

Tugas 4 II4021 Kriptografi – Semester II tahun 2025/2026

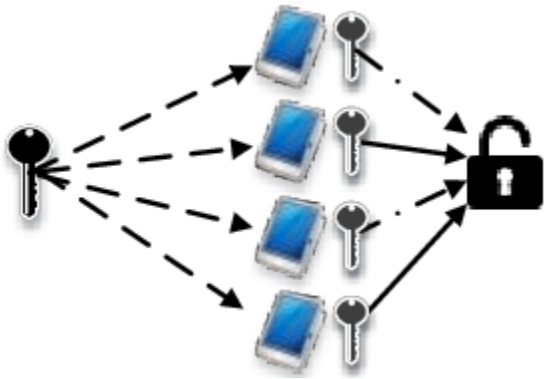
Implementasi Shamir Secret Sharing pada Aplikasi Password Manager Terdistribusi

---

Batas pengumpulan	**:** Minggu, 31 Mei 2026 Pukul 23.59
Tempat pengumpulan	**:** Form Pengumpulan
Anggota kelompok	**:** 2-3 orang
Sheets kelompok	**:** Sheets Kelompok
QnA	**:** QnA Tugas II4021 Kriptografi

Latar Belakang

Raka adalah mahasiswa dengan banyak akun digital, mulai dari email, media sosial, hingga portal kampus. Setiap akun memiliki password berbeda, sehingga mengingat dan mengelolanya secara manual menjadi sulit.



Gambar 1. Contoh Skema Secret Sharing (source: Aleksandr Ometov)

Untuk menyelesaikan masalah ini, Raka sedang mengembangkan *password manager* sendiri. Sistem ini membagi akses ke data penting menjadi beberapa bagian agar tetap aman dan mudah dipulihkan. Misalnya, ia bisa menyimpan satu bagian di komputernya dan bagian lain sebagai cadangan di ponsel. Dengan pendekatan ini, kehilangan satu perangkat tidak akan menghalangi Raka untuk membuka *vault*, sekaligus menjaga keamanannya.

Spesifikasi dan Ketentuan Program

Gambaran Umum Sistem

Aplikasi yang dibangun merupakan *password manager* terdistribusi untuk menyimpan data akun pengguna dalam sebuah *vault* terenkripsi. Data yang disimpan mencakup nama layanan, *username* atau email, password, dan catatan opsional. Sistem menggunakan arsitektur *client-server*, dengan klien sebagai tempat utama proses kriptografi dan server sebagai tempat penyimpanan data terenkripsi.

Seluruh isi *vault* dienkripsi menggunakan AES-128-GCM. Kunci utama (*master key*) untuk membuka *vault* tidak disimpan secara utuh, tetapi dibagi menjadi beberapa *share* menggunakan Shamir Secret Sharing. Pada tugas ini, sistem menggunakan skema ambang batas (2, 3), sehingga *vault* hanya dapat dibuka apabila minimal dua *share* valid berhasil digabungkan.

Sistem mendukung dua mode akses, yaitu mode normal dan mode backup. Mode normal menggunakan kombinasi *local share* dan *server share*, sedangkan mode backup menggunakan *local share* dan *recovery share*. Dengan rancangan ini, server tidak menyimpan *master key* secara utuh dan tidak menerima isi *vault* dalam bentuk asli.

Arsitektur dan Penyimpanan Data

Sistem menggunakan arsitektur *client-server*. Klien menjalankan proses kriptografi, sedangkan server digunakan sebagai tempat penyimpanan data yang digunakan oleh sistem. Pada tugas ini, server wajib menggunakan SQLite untuk menyimpan data.

Dalam sistem ini, pembagian tanggung jawab antara klien dan server harus dibuat secara jelas. Klien berperan sebagai tempat utama proses kriptografi, sedangkan server hanya berperan sebagai media penyimpanan data terenkripsi dan data pendukung yang diperlukan untuk akses normal.

Klien bertanggung jawab untuk melakukan operasi berikut:

1. Membangkitkan *master key* secara acak.
2. Membagi *master key* menjadi tiga *share* menggunakan Shamir Secret Sharing dengan skema ambang batas (2, 3).
3. Menurunkan kunci dari *master password* menggunakan *Key Derivation Function* (KDF).
4. Mengenkripsi dan mendekripsi *local share*.
5. Merekonstruksi *master key* dari minimal dua *share* yang valid.
6. Mengenkripsi dan mendekripsi vault menggunakan AES-128-GCM.
7. Membuat dan memperbarui *backup vault* lokal dalam bentuk terenkripsi.

8. Membangkitkan *password* otomatis menggunakan *Cryptographically Secure Pseudorandom Number Generator* (CSPRNG), apabila fitur pembangkitan *password* digunakan.

Server bertanggung jawab untuk melakukan operasi berikut:
1. Menyimpan *server share*.
2. Menyimpan *vault* terenkripsi.
3. Menyimpan *nonce* yang digunakan pada enkripsi *vault* dengan AES-128-GCM.
4. Menyimpan *metadata* yang diperlukan, seperti identitas pengguna.
5. Mengirimkan *server share*, *vault* terenkripsi, dan *nonce* kepada klien pada mode akses normal.
6. Menerima pembaruan *vault* terenkripsi dan *nonce* baru dari klien setelah terjadi perubahan data pada mode normal.

Untuk menjaga prinsip *zero-knowledge*, *server* tidak boleh menerima atau menyimpan data berikut:
1. *Master key* dalam bentuk utuh.
2. *Local share*.
3. *Recovery share*.
4. Isi *vault* dalam bentuk plaintext.
5. *Password* pengguna dalam bentuk plainteks.
6. Kunci turunan dari *master password*.

Server juga tidak boleh melakukan operasi berikut:
1. Merekonstruksi *master key*.
2. Mengenkripsi atau mendekripsi isi *vault*.
3. Membuka, membaca, atau memodifikasi isi *vault* dalam bentuk asli.

Dengan pembagian ini, seluruh proses kriptografi sensitif berada di sisi klien, sedangkan *server* hanya menyimpan *ciphertext* dan satu *share*. Oleh karena itu, kompromi terhadap *server* saja tidak cukup untuk membuka *vault* pengguna.

Data yang disimpan pada masing-masing komponen ditunjukkan pada Tabel 1.

Tabel 1. Penyimpanan Data pada Komponen Sistem

Lokasi Penyimpanan	Data yang Disimpan	Keterangan
Klien	Local share yang terenkripsi	Local share tidak disimpan dalam bentuk asli. Local share dienkripsi menggunakan kunci turunan dari master password.
Klien	Nonce enkripsi local share	Digunakan untuk mendekripsi local share ketika pengguna memasukkan master

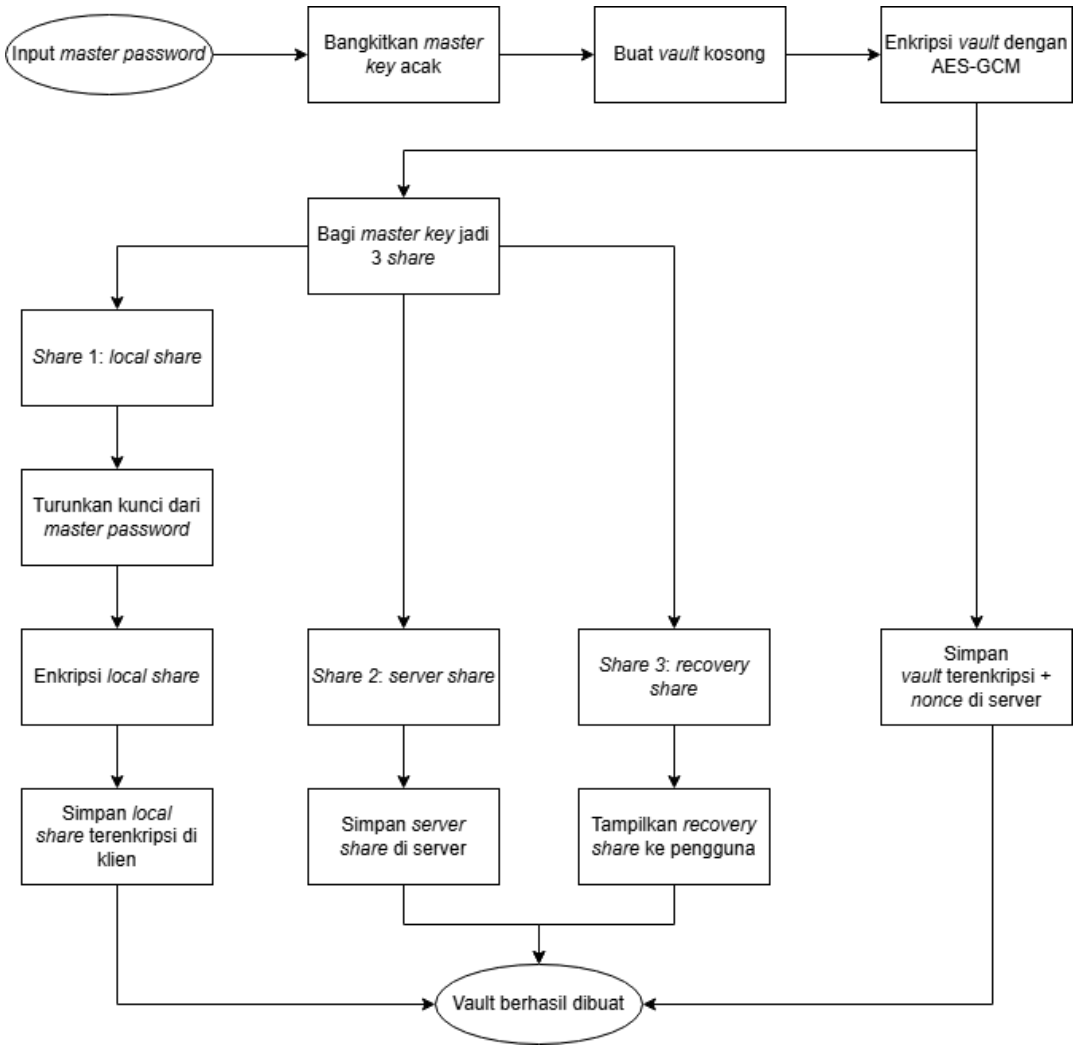
		password.
Klien	Salt KDF dan parameter KDF	Digunakan untuk menurunkan kunci dari master password.
Klien	Backup vault lokal terenkripsi	Digunakan pada mode backup ketika server tidak dapat diakses.
Klien	Nonce backup vault	Digunakan untuk mendekripsi backup vault lokal.
Server	Server share	Salah satu share hasil pembagian master key. Share ini tidak cukup untuk membuka vault tanpa share lain.
Server	Vault terenkripsi	Isi vault yang telah dienkripsi menggunakan AES-128-GCM di sisi klien.
Server	Nonce vault	Nonce yang digunakan pada enkripsi vault terakhir.
Server	Metadata pengguna	Data pendukung seperti identitas pengguna atau informasi lain yang diperlukan untuk pengelolaan vault.
Pengguna	Recovery share	Share cadangan yang ditampilkan kepada pengguna dan harus disimpan secara mandiri.

Pembuatan Vault

Alur pembuatan *vault* dimulai ketika pengguna memasukkan *master password*. Klien kemudian membangkitkan *master key* secara acak, membuat *vault* kosong, dan mengenkripsinya menggunakan AES-128-GCM. Setelah itu, *master key* dibagi menjadi tiga *share* menggunakan Shamir Secret Sharing dengan skema (2, 3), sebagaimana ditunjukkan pada Gambar 2.

Tiga *share* yang dihasilkan terdiri atas *local share*, *server share*, dan *recovery share*. *Local share* dienkripsi menggunakan kunci turunan dari *master password*, lalu disimpan di klien. *Server share* disimpan di server bersama *vault* terenkripsi dan *nonce*. *Recovery share* ditampilkan kepada pengguna sebagai cadangan. *Recovery share* ditampilkan dalam format teks yang dapat disalin atau disimpan pengguna. Format share harus memuat koordinat *share* dan nilai *share*, misalnya dalam bentuk base64, hex, atau JSON. Dengan alur ini, *master key* tidak pernah disimpan secara utuh, dan server hanya memegang satu *share* sehingga tidak dapat membuka *vault* sendiri.

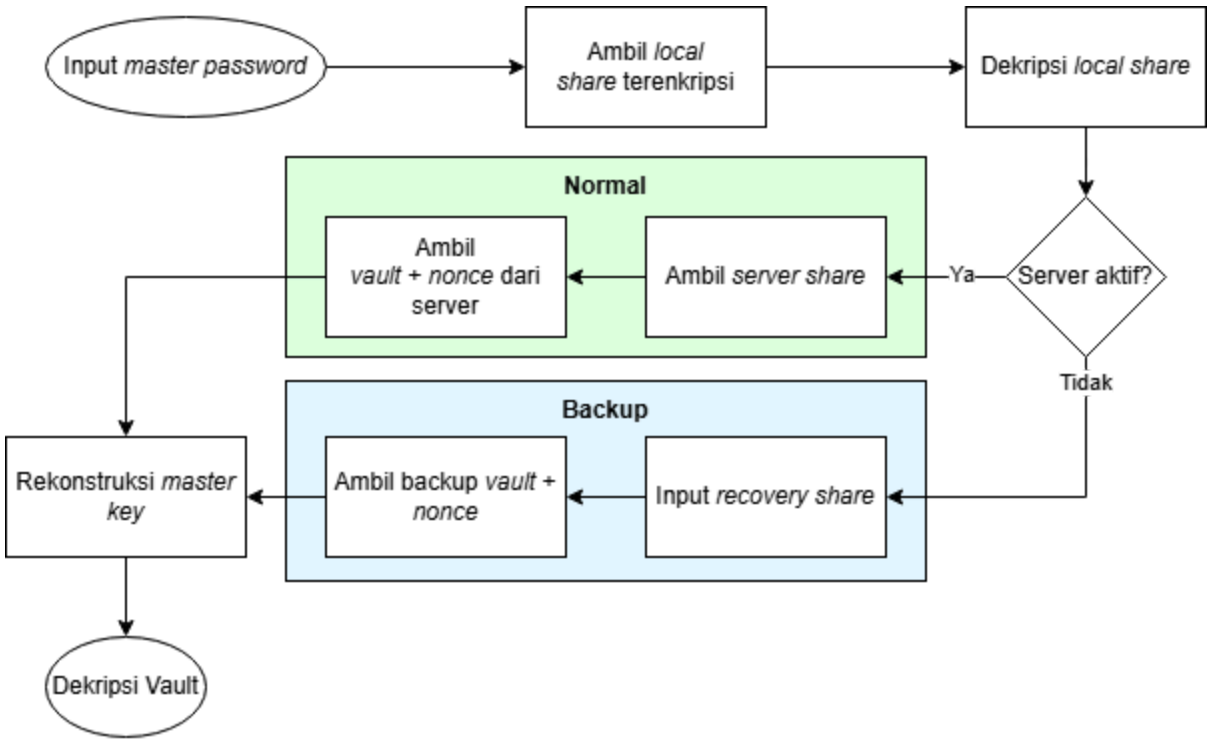
Recovery share harus ditampilkan hanya satu kali pada saat pembuatan *vault* atau pada kondisi yang ditentukan oleh implementasi. Pengguna bertanggung jawab untuk menyimpan *recovery share* secara aman. Apabila *recovery share* hilang, maka mode backup tidak dapat digunakan.



Gambar 2. Diagram Alur Pembuatan Vault

Alur Akses dan Pemulihan Vault

Alur akses *vault* dimulai ketika pengguna memasukkan *master password* untuk membuka *local share* yang tersimpan dalam bentuk terenkripsi. Setelah *local share* diperoleh, sistem akan berjalan dalam dua kemungkinan alur, yaitu akses normal atau akses pemulihan, sebagaimana ditunjukkan pada Gambar 3.



Gambar 3. Diagram Alur Akses dan Pemulihan Vault

Pada mode normal, klien mengambil *server share*, *vault* terenkripsi, dan *nonce vault* dari *server*. Setelah itu, klien menggabungkan *local share* dan *server share* untuk merekonstruksi *master key*. *Master key* hasil rekonstruksi kemudian digunakan untuk mendekripsi *vault* menggunakan AES-128-GCM.

Pada mode backup, klien tidak mengambil data dari *server*. Klien menggunakan *local share* yang berhasil didekripsi, *recovery share* yang dimasukkan oleh pengguna, *backup vault* lokal terenkripsi, dan *nonce backup vault*. *Local share* dan *recovery share* digunakan untuk merekonstruksi *master key*, kemudian *master key* digunakan untuk mendekripsi *backup vault* lokal.

Mode backup hanya digunakan untuk akses darurat ketika *server* tidak dapat diakses. Pada mode ini, pengguna hanya dapat melihat data *password*. Sistem tidak melakukan penambahan, pengubahan, atau penghapusan data *password* pada mode backup agar tidak terjadi konflik versi data antara *backup* lokal dan vault yang tersimpan di *server*.

Pada mode normal maupun mode backup, *master key* hasil rekonstruksi digunakan untuk mendekripsi *vault* menggunakan AES-128-GCM. Jika dekripsi gagal, akses ditolak dan data password tidak ditampilkan.

Perbedaan antara mode normal dan mode backup ditunjukkan pada Tabel 2.

Tabel 2. Perbedaan Mode Normal dan Mode Backup

Aspek	Mode Normal	Mode Backup
Kombinasi *share*	*Local share* + *server share*	*Local share* + *recovery share*
Sumber *vault*	*Vault* terenkripsi dari server	Backup *vault* terenkripsi di klien
Operasi yang didukung	Melihat, menambah, mengubah dan menghapus data password	Melihat data password
Pembaruan data	*Vault* dienkripsi ulang dengan *nonce* baru, lalu disimpan ke server dan backup lokal	Tidak melakukan pembaruan data
Tujuan utama	Akses biasa	Akses darurat saat server tidak dapat diakses

Penambahan Data Password ke dalam Vault

Penambahan data password dilakukan setelah *vault* berhasil dibuka pada **mode normal**. Setiap data password yang ditambahkan ke dalam *vault* minimal memuat atribut sebagaimana ditunjukkan pada Tabel 3.

Tabel 3. Struktur Minimal Data Password

Atribut	Status	Contoh
`nama_layanan`	Wajib	GitHub
`username`	Wajib	diero@example.com
`password`	Wajib	password_tersimpan
`catatan`	Opsional	Akun utama

Penambahan, pengubahan, dan penghapusan data *password* dilakukan terhadap isi *vault* di sisi klien. Setelah perubahan dilakukan, seluruh isi *vault* dienkripsi ulang sebagai satu kesatuan dan disimpan kembali ke server sebagai BLOB di SQLite.

Saat menambahkan data password, pengguna diberi dua pilihan untuk menentukan nilai password yang akan disimpan:

1. **Input manual**, yaitu pengguna memasukkan password sendiri.
2. **Pembangkitan otomatis**, yaitu pengguna memasukkan panjang password, lalu klien membangkitkan password menggunakan CSPRNG.

Untuk fitur pembangkitan otomatis, password yang dihasilkan dapat terdiri atas kombinasi huruf besar, huruf kecil, angka, dan simbol. Algoritma CSPRNG yang digunakan dibebaskan, tetapi harus dijelaskan pada laporan, termasuk cara bilangan acak dihasilkan dan cara hasilnya dipetakan menjadi karakter password.

Bonus: Kriptografi Visual untuk *Recovery Share*

Fitur *recovery share* dapat diimplementasikan menggunakan kriptografi visual. *Recovery share* terlebih dahulu diubah menjadi QR code, lalu QR tersebut dibagi menjadi dua gambar *share* dengan skema (2, 2). Gabungan kedua gambar *share* harus dapat menghasilkan QR code yang dapat dipindai kembali dan memuat *recovery share* yang sama.

Prosedur Pengerjaan

Berikut adalah ketentuan pengerjaan tugas ini:

1. Batas pengisian kelompok di Sheets kelompok: **Rabu, 20 Mei 2026 pukul 23.59**
2. Implementasi aplikasi dilakukan berbasis **CLI**. Kakas yang digunakan untuk *client* dan *server* **dibebaskan**.
3. Penggunaan library untuk enkripsi, dekripsi, dan secret sharing diperbolehkan.
4. Program harus ditulis dengan struktur yang rapi, mudah dibaca, dan disertai komentar yang jelas pada bagian yang diperlukan.
5. Dilarang menyalin kode program secara langsung dari internet maupun menggunakan keluaran mentah (*raw output*) dari *Generative AI*.
6. Antarmuka aplikasi dibebaskan, tetapi harus tetap fungsional, mudah digunakan, dan memenuhi alur minimum yang telah ditentukan.

Pengumpulan

Pengumpulan dilakukan dengan melakukan ***release*** pada repositori Github sebelum tenggat waktu pengerjaan. Revisi pengumpulan dapat dilakukan dengan membuat release baru.

1. Repositori Perangkat Lunak

Repositori harus memuat:

a. Kode sumber (*source code*).
b. Berkas eksekutabel (*executable file*), jika relevan.
c. Dokumen laporan
d. README yang minimal berisi:
    i. Nama dan deskripsi program.
    ii. Teknologi yang digunakan (*tech stack*).
    iii. Dependensi.

iv. Tata cara menjalankan program.
v. *Environment/configuration* yang digunakan.

2. Video Demo

Video demo berdurasi maksimal 10 menit, berisi:

a. Deskripsi singkat program.
b. Penjelasan singkat rancangan, terutama bagian yang dibebaskan atau dirancang sendiri.
c. Demo kasus uji utama sesuai ketentuan pengujian.

Apabila terdapat proses yang memerlukan waktu lama untuk dijalankan, video demo dapat dipotong pada bagian tersebut selama alur demonstrasi tetap jelas.

3. Laporan

Laporan diunggah pada repositori dengan nama **NIM1_NIM2_NIM3_Tugas4_II4021.pdf**, berisi:

a. Foto anggota kelompok di *cover* laporan sebagai pengganti logo gajah.
b. Pernyataan tidak melakukan kecurangan yang ditandatangani dengan format sebagai berikut:

Kami menyatakan bahwa kode program yang dihasilkan bukan merupakan hasil salinan mentah (*raw output*) dari *Generative AI*, melainkan hasil pengembangan dan penulisan mandiri. [Tanda tangan Mahasiswa 1] [Tanda tangan Mahasiswa 2] [Tanda tangan Mahasiswa 3] [Nama Mahasiswa 1] [Nama Mahasiswa 2] [Nama Mahasiswa 3]

c. Teori singkat mengenai Shamir Secret Sharing, AES-GCM, KDF, CSPRNG, dan konsep relevan lainnya termasuk bonus.
d. Perancangan dan implementasi, berisi penjelasan rancangan dan implementasi program, **bukan penyalinan isi codebase**. Setiap bagian yang dijelaskan harus disertai nama file yang relevan pada repository. Pastikan untuk menjelaskan implementasi bagian yang dibebaskan dan bonus.
e. Pengujian program dan analisis hasil, dengan minimal kasus uji mencakup:
    i. Uji pembuatan *vault*:
        1. Buat vault baru dengan *master password* valid, perlihatkan bahwa *master key* berhasil dibangkitkan, *vault* kosong berhasil dienkripsi, dan data awal berhasil disimpan.
        2. Perlihatkan bahwa *master key* dibagi menjadi *tiga share* menggunakan Shamir Secret Sharing dengan skema (2, 3), yaitu *local share*, *server share*, dan *recovery share*.
        3. Perlihatkan bahwa *recovery share* ditampilkan dalam format teks yang memuat koordinat share dan nilai share.

4. Perlihatkan bahwa server hanya menyimpan *server share*, *vault terenkripsi*, *nonce vault*, dan metadata yang diperlukan.

ii. Uji penyimpanan dan perlindungan *local share*:
   1. Perlihatkan bahwa *local share* tidak disimpan dalam bentuk asli.
   2. Perlihatkan bahwa *local share* berhasil dienkripsi menggunakan kunci turunan dari *master password*.
   3. Masukkan *master password* yang benar, perlihatkan bahwa *local share* berhasil didekripsi.
   4. Masukkan *master password* yang salah, perlihatkan bahwa *local share* gagal didekripsi atau akses ditolak.

iii. Uji akses normal:
   1. Buka *vault* dengan *master password* yang benar ketika server dapat diakses, perlihatkan bahwa sistem menggunakan kombinasi *local share* dan *server share*.
   2. Perlihatkan bahwa *master key* berhasil direkonstruksi dari dua share valid.
   3. Perlihatkan bahwa *vault* berhasil didekripsi dan isi *vault* dapat ditampilkan.
   4. Masukkan *master password* yang salah atau gunakan share yang tidak valid, perlihatkan bahwa dekripsi *vault* gagal dan data password tidak ditampilkan.

iv. Uji penambahan data password:
   1. Tambahkan data password dengan input manual, perlihatkan bahwa data berhasil masuk ke *vault*.
   2. Tambahkan data password dengan pembangkitan otomatis menggunakan CSPRNG, perlihatkan bahwa password berhasil dibangkitkan sesuai panjang yang diminta.
   3. Perlihatkan bahwa setelah penambahan data, *vault* dienkripsi ulang menggunakan *nonce* baru dan disimpan kembali ke server.
   4. Perlihatkan bahwa backup *vault* lokal juga diperbarui setelah perubahan pada mode normal.

v. Uji pengubahan dan penghapusan data password:
   1. Ubah salah satu data password yang tersimpan, perlihatkan bahwa perubahan berhasil tersimpan setelah *vault* dienkripsi ulang.
   2. Hapus salah satu data password yang tersimpan, perlihatkan bahwa data tersebut tidak lagi muncul setelah *vault* dibuka kembali.
   3. Perlihatkan bahwa pembaruan hanya dilakukan pada mode normal.

vi. Uji penyimpanan vault di server:
   1. Perlihatkan bahwa *vault* terenkripsi disimpan di SQLite sebagai BLOB.
   2. Perlihatkan bahwa server tidak menyimpan nama layanan, username, password, atau catatan sebagai baris plaintext terpisah.

3. Perlihatkan bahwa server tidak menerima *master key*, *local share*, *recovery share*, maupun isi *vault* dalam bentuk plaintext.

vii. Uji mode backup:
1. Simulasikan kondisi server tidak dapat diakses.
2. Buka *vault* menggunakan *master password* dan *recovery share*, perlihatkan bahwa sistem menggunakan kombinasi *local share* dan *recovery share*.
3. Perlihatkan bahwa sistem menggunakan backup *vault* lokal terenkripsi sebagai sumber *vault*.
4. Perlihatkan bahwa data password dapat dilihat pada mode backup.
5. Perlihatkan bahwa pengguna tidak dapat menambah, mengubah, atau menghapus data password pada mode backup.

viii. Uji kegagalan pemulihan:
1. Masukkan *recovery share* yang salah, perlihatkan bahwa rekonstruksi *master key* atau dekripsi *vault* gagal.
2. Gunakan backup *vault* yang tidak sesuai atau sudah dimodifikasi, perlihatkan bahwa dekripsi AES-128-GCM gagal karena validasi autentikasi tidak sesuai.
3. Perlihatkan bahwa sistem tidak menampilkan data password ketika proses dekripsi gagal.

ix. Uji kriptografi visual (*bonus*)
1. Tampilkan QR code awal dari *recovery share*.
2. Tampilkan dua gambar *share* hasil pembagian QR code.
3. Tampilkan hasil penggabungan kedua gambar *share*.
4. Perlihatkan bahwa QR code hasil penggabungan dapat dipindai dan menghasilkan *recovery share* yang sama.

f. Kesimpulan dari hasil implementasi,
g. Daftar pustaka,
h. Lampiran berisi:
i. Pranala repositori
ii. Pranala video demo. Pranala video demo tidak boleh menggunakan url shortener dan tidak boleh mengarah ke folder google drive (jika menggunakan google drive, harap langsung share link videonya).
iii. Pembagian tugas

Referensi

- Pengembangan Aplikasi Manajemen Kata Sandi Terdistribusi Dengan Skema Pembagian Data Rahasia