**Memo — Encryption/Decryption Component**

**Group 3: Fnu Abdullah & Ahmed Hussain**

**Summary**

This memo describes the encryption/decryption component architecture. Symmetric keys are derived from user passwords using SHA256 with a per-file random salt and a secure iteration count, and AES with a random IV handles encryption. The design separates client-side actions from all secret credentials by using a server-side function (Supabase Edge Function) to generate and store per-user encryption keys. Row-level security (RLS) and service-role credentials protect the database. The decryption component verifies passwords with a server-side endpoint before decrypting files locally.

**Components & Dataflow**

1. **Client encryptor** 
   * Obtains a stable user\_id (persisted locally) and requests a per-user password from a protected server endpoint.
   * Uses that password locally as the symmetric key (via AES) to encrypt files in a designated test folder.
   * Marks the folder with a local flag file indicating “encrypted” for subsequent processes.
2. **Server-side key service (Edge Function)**
   * Receives a request containing user\_id plus a short-lived shared secret header.
   * Validates the short-lived shared secret, then generates a random password (server-side) and stores it in the Supabase table encryption\_keys using the always hidden service role key.
   * Returns the generated password to the caller.
3. **Database (Supabase)**
   * Stores { unique\_id, user\_id, password } in encryption\_keys.
   * RLS enabled; all permissive anon policies removed so only server-side service-role operations can read/write.
4. **Client decryptor (GUI)**
   * Runs on the same test machine. Presents a fullscreen GUI for the user to enter a password.
   * Sends user\_id + user-entered password to a server-side verify-password Edge Function that checks the provided password against stored value.
   * If the verify endpoint returns success, the client decrypts files using the same KDF + AES algorithm.

**Key management & security controls**

* **Secrets never exposed to client:** Supabase service-role key and DB credentials are kept only in the Edge Function environment variables (edge runtime / server). Clients never receive these keys.
* **RLS policies:** All public/anon permissions are removed. With RLS enforced, direct anon access cannot read/write the encryption\_keys table.
* **Brief shared key:** The setup script exports a short-lived EDGE\_SHARED\_KEY in the environment for the one-time server call, then unsets it immediately and removes installer files so the secret is not persisted to disk.
* **Server-only generation:** Passwords are generated server-side and stored with the service role.
* **Minimal server API surface:** Edge Function accepts only specific JSON request fields and checks the shared secret header; it does not expose broad DB operations.

**Cryptography**

* Symmetric keys are derived from the password using PBKDF2-HMAC-SHA256 with a per-file random salt and a secure iteration count. AES-CBC is used with a random IV.