SafeVault

Client-side Media Encryption Platform

Adeeb Ali
`aliadee2003@gmail.com`

October 27, 2025

Encrypt locally. Own your privacy.

Abstract

SafeVault is a privacy-first, open-source web application that enables users to securely encrypt and decrypt their photos and videos entirely in the browser. When a user uploads a media file, SafeVault converts it into an encrypted code-file that can be stored safely on the user's device. Later, the code-file can be uploaded back and, with the correct password/key, restored to the original media. This document describes the idea, architecture, security approach, tech stack, workflow and contribution plan.

1 Motivation

I believe privacy should be simple, accessible, and under everyone's control. SafeVault aims to give individuals an easy and dependable way to keep personal media private without relying on third-party cloud providers. I will be actively working on this project in December (during my winter vacation) and will document every step openly. This project will always remain **100% open source**. Anyone may contribute at any time; every contribution adds value to the tech ecosystem in India and worldwide.

2 Goals

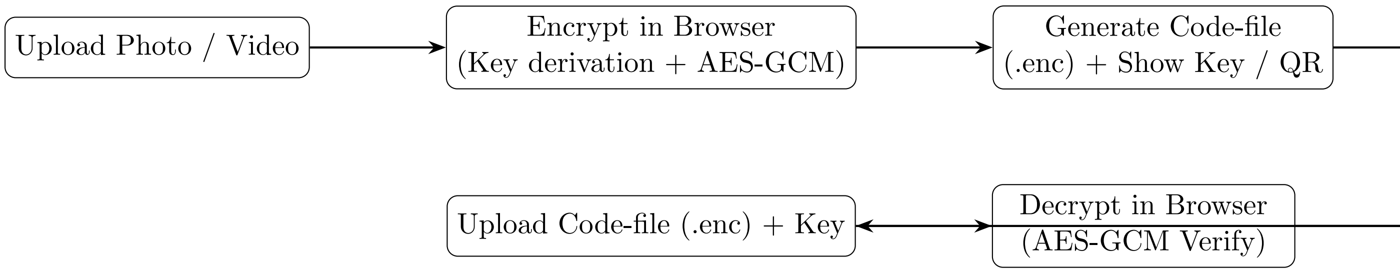
- Provide a simple, mobile-friendly UI that lets users encrypt photos and videos in the browser.
- Ensure all encryption and decryption occur client-side so plaintext never leaves the user's device.
- Produce a portable `.enc` (code) file and a user-visible key/password for safe long-term storage.
- Keep the project open-source, well-documented, and easy for contributors to join.
- Deliver a world-class web experience through community collaboration.

3 High-level Architecture

- **Client (Browser)**: UI, file read, key derivation, encryption (AES-GCM), code-file generation, decryption.
- **Optional Backend (Node.js + Express)**: host static site, optionally store encrypted blobs (ciphertext only), serve metadata or backups (never plaintext or passwords).

- **Storage Options**: Local device file system (primary), optional cloud blob storage (S3/R2) of ciphertext only.

Flow diagram:



Upload Photo / Video
Encrypt in Browser (Key derivation + AES-GCM)
Generate Code-file (.enc) + Show Key / QR
Upload Code-file (.enc) + Key
Decrypt in Browser (AES-GCM Verify)

4 Encryption Design (summary)

1. **Key derivation:** derive a 256-bit encryption key from user password/passphrase using a secure KDF (PBKDF2 with high iterations or Argon2 if using wasm).
2. **Cipher:** AES-GCM (authenticated encryption) with a random 96-bit IV per encryption.
3. **Salt:** store a random salt per file (used for key derivation) inside the code-file (not secret).
4. **Code-file format:** JSON wrapper (versioned) containing: version, kdf, kdf-params, salt (base64), iv (base64), mime type, ciphertext (base64). Then base64/URL-safe encode the JSON if you want a single-line token, or save as a `.enc` file.
5. **Authentication:** AES-GCM tag verifies integrity — wrong password or tampered ciphertext will fail to decrypt.

5 User Experience (UX) Flow

1. User visits the Upload page.
2. User selects photo/video.
3. Browser prompts (optionally) to set a password or generate a random 256-bit key (shown as hex and QR).
   Recommendation: offer both — random-key (recommended) and passphrase (optional).
4. File is read in browser, encrypted, and the `.enc` file is prepared for download.
5. User downloads and stores the `.enc` file on their phone or computer. They are shown clear instructions to back up the key/password (e.g., copy to password manager, save QR, print).
6. To restore, user uploads the `.enc` file and provides the key/password — the browser decrypts and shows the media.

6 Tech Stack (recommended)

- **Frontend:** Next.js (React) + TypeScript + Tailwind CSS
- **Client Crypto:** Web Crypto API (SubtleCrypto) for AES-GCM + PBKDF2; optionally `argon2-browser` (WASM) for Argon2
- **Backend (optional):** Node.js + Express (only for hosting and optional ciphertext storage)
- **Database (optional):** MongoDB Atlas (for metadata or user accounts; do not store keys)
- **Hosting:** Vercel (frontend), Render/Railway (backend) — free tiers available for initial development

7 Security Notes & Best Practices

- Always perform encryption and decryption on the client. Never transmit user plaintext or passwords to the server.
- Use authenticated encryption (AES-GCM) to detect tampering.
- Encourage users to use a strong, randomly generated key (256-bit) and back it up securely; losing the key means losing the ability to decrypt.
- If using passphrases, use a slow KDF (Argon2 preferred) to resist brute-force attacks.
- If storing ciphertext on a server, treat it as public — rotate and secure access to storage but assume ciphertext may be available to attackers.
- Display explicit warnings: "If you lose your password/key, files cannot be recovered."

8 Deployment & Open Source

- Publish the code under an open license (MIT recommended for maximum reuse).
- Host frontend on Vercel for easy CI/CD and fast demos.
- If using a backend for optional features, deploy it on a free tier (Render/Railway) and connect to MongoDB Atlas free tier for metadata.
- Provide a clear README, CONTRIBUTING guide, and issue templates to attract contributors.

9 Repository Structure (suggested)

```
safevault/
 frontend/         # Next.js + TypeScript + Tailwind
    src/
       pages/
       components/
       lib/crypto.ts   # encryption helper (WebCrypto)
 backend/          # optional: Node.js + Express
 README.md
 CONTRIBUTING.md
 LICENSE
```

10 Roadmap (initial)

1. Project initialization, README and basic architecture (complete).
2. Prototype client-side encryption/decryption (AES-GCM) and file download/upload.
3. Build UI (upload, download, QR for keys) and UX polish (Tailwind).
4. Add optional server for encrypted backups (ciphertext only) and metadata.
5. Testing, documentation, and release a demo site.

11 Call for Contributors

SafeVault is fully open source and intended as a collaborative project. If you are interested in frontend design, cryptography, or building privacy-first tools, your contributions are welcome. Every pull request and idea will help make the project stronger and more useful to people in India and around the world.

Contact

- Author: Adeeb Ali
- Email: `aliadee2003@gmail.com`
- Repository (placeholder): `https://github.com/adeeb-ali/safevault`

"Privacy is not a feature — it's a right."
— SafeVault Team