TRƯỜNG ĐẠI HỌC PHENIKAA

TRƯỜNG CÔNG NGHỆ THÔNG TIN PHENIKAA

BÁO CÁO THỰC HIỆN NHÓM CHỨC NĂNG 2

HỌC PHẦN: ĐÁNH GIÁ VÀ KIỂM ĐỊNH CHẤT LƯỢNG PHẦN MỀM

Lớp N01 - Nhóm 8

Phần mềm tính tiền dạy cho giáo viên

Thành viên nhóm

Họ và tên	MSSV	Lớp
Đặng Đức Huy	22010274	K16-CNTT-2
Độ Hội Nam		K16-CNTT-
		K16-CNTT-

GVHD: TS. Mai Thúy Nga

Hà Nội, 6/2025

Lời cảm ơn

Tác giả xin chân thành cảm ơn giảng viên hướng dẫn, các thầy cô bộ môn và bạn bè đã hỗ trợ trong suốt quá trình thực hiện đề tài. Xin gửi lời cảm ơn tới cộng đồng mã nguồn mở đã cung cấp các thư viện và tài liệu tham khảo giúp dự án hoàn thiện.

Hà Nội, ngày … tháng … năm 2025
Sinh viên thực hiện

Lời cam đoan

Tôi xin cam đoan báo cáo này là kết quả nghiên cứu và triển khai của bản thân trong phạm vi môn học. Các nội dung trích dẫn đều được ghi rõ nguồn. Nếu có sai phạm, tôi xin hoàn toàn chịu trách nhiệm.

Tóm tắt

Đề tài xây dựng hệ thống chia sẻ tệp tin bảo mật theo mô hình zero-knowledge: dữ liệu được **mã hóa phía client** bằng AES-256-GCM, máy chủ chỉ lưu ciphertext và metadata. Quyền truy cập được cấp qua **mã chia sẻ (token)** có thời hạn. Người dùng đăng nhập bằng **chữ ký ví Ethereum (MetaMask)** để xác thực mà không cần mật khẩu. Báo cáo trình bày lý do lựa chọn thuật toán, thiết kế hệ thống, phân tích bảo mật, luồng thuật toán chi tiết, hiện thực hoá và đánh giá.

Mục lục

Danh sách hình vẽ

Danh sách bảng

Danh mục từ viết tắt

AES	Advanced Encryption Standard
GCM	Galois/Counter Mode
PBKDF2	Password-Based Key Derivation Function 2
JWT	JSON Web Token
CID	Content Identifier
API	Application Programming Interface
CSRF	Cross-Site Request Forgery
KDF	Key Derivation Function
UI	User Interface

Chương 1

Tổng quan đề tài

1.1 Bối cảnh và động lực

Trong kỷ nguyên dữ liệu, nhu cầu *chia sẻ an toàn* và *kiểm soát quyền riêng tư* ngày càng cấp thiết. Mô hình lưu trữ tập trung có nguy cơ lộ lọt dữ liệu và phụ thuộc nhà cung cấp. Tiếp cận **client-side encryption** đảm bảo khoá bí mật không rời thiết bị người dùng, giảm thiểu rủi ro xâm phạm phía máy chủ.

1.2 Mục tiêu

- Xây dựng hệ thống chia sẻ tệp với **mã hóa client-side AES-256-GCM**.
- Cấp quyền truy cập qua **token** có TTL, có thể thu hồi.
- Xác thực đăng nhập bằng **chữ ký ví Ethereum (MetaMask)**.
- Lưu ciphertext và metadata an toàn; máy chủ **không biết khoá giải mã**.

1.3 Phạm vi

Trong phạm vi

Web app Next.js, mã hóa bằng Web Crypto API, lưu ciphertext local (demo), metadata trong SQLite, quản lý token và phiên đăng nhập bằng JWT cookie.

Ngoài phạm vi

Triển khai IPFS/Filecoin thực tế, hợp đồng thông minh on-chain và ứng dụng di động nằm ngoài phạm vi môn học (để mở hướng phát triển).

Chương 2

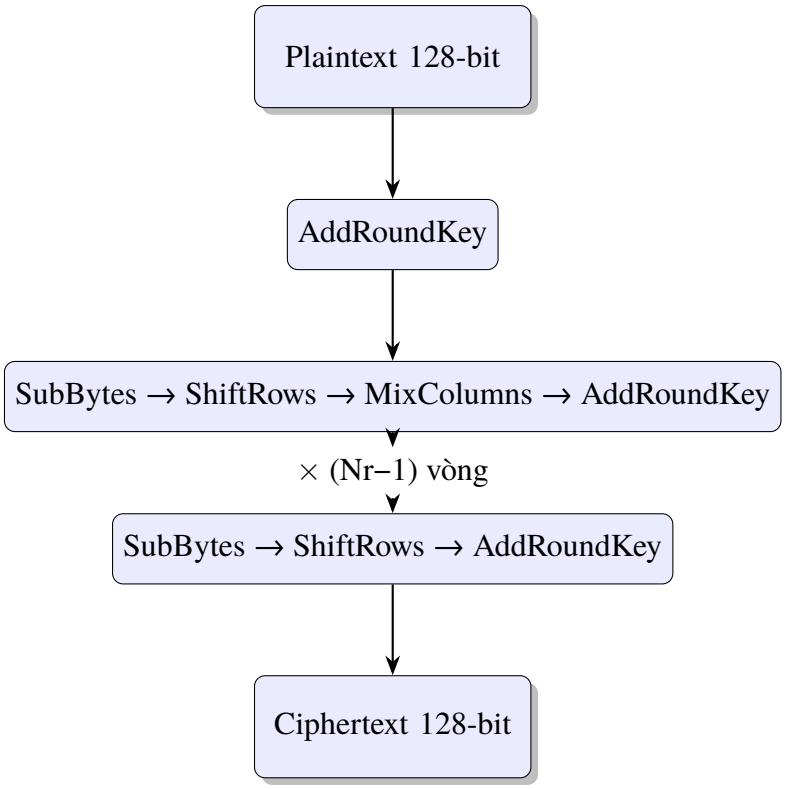
Cơ sở lý thuyết

2.1 Lý do chọn AES-256-GCM

AES là chuẩn mực công nghiệp, được tăng tốc phần cứng và hỗ trợ native bởi Web Crypto API. **GCM** cung cấp cả bảo mật *bí mật* lẫn *toàn vẹn/xác thực* (AEAD). Kích thước IV 96-bit tiêu chuẩn giúp hiệu năng tốt và đơn giản khi triển khai. Với khoá 256-bit, không có tấn công thực tế tốt hơn vét cạn.

2.2 Tổng quan AES

AES là mã khối kích thước 128-bit, cấu trúc SPN với các bước *SubBytes*, *ShiftRows*, *MixColumns*, *AddRoundKey*. Số vòng phụ thuộc độ dài khoá (14 vòng cho 256-bit).



Hình 2.1: Cấu trúc vòng AES (minh hoạ)

2.3 GCM và tính xác thực

GCM kết hợp CTR để mã hóa và GHASH để tính nhãn xác thực (tag). Đầu vào gồm khoá bí mật, IV (không lặp), dữ liệu kèm theo AAD (tuỳ chọn), bản rõ; đầu ra gồm bản mã và tag xác thực. Sai IV hoặc thay đổi dữ liệu khiến giải mã thất bại.

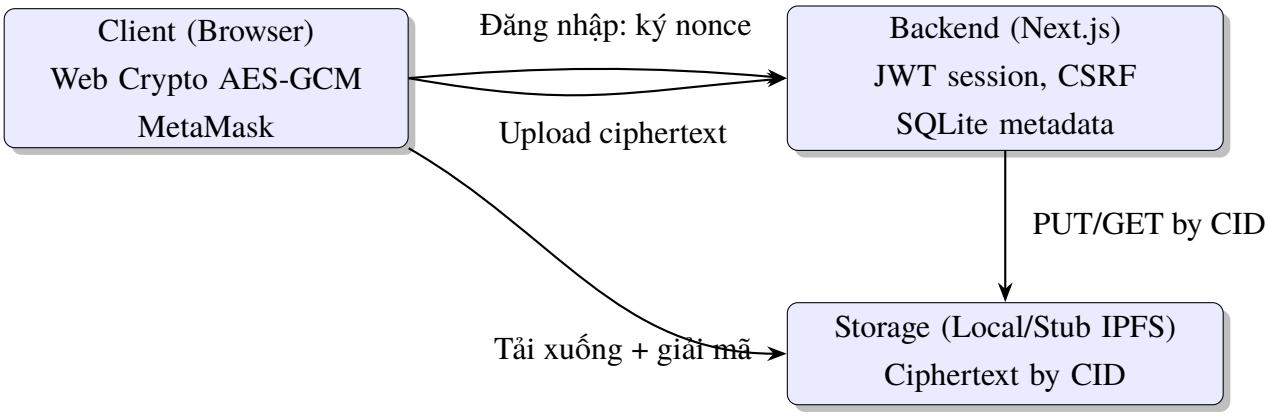
2.4 PBKDF2 và bao bọc khoá

Khi người dùng chọn passphrase, hệ thống dùng PBKDF2-HMAC-SHA-256 (200k vòng) sinh khoá dẫn xuất để **bao bọc khoá AES** bằng AES-GCM (key wrapping). Lưu trữ: salt (16B), ivWrap (12B) và wrappedKey. Nếu ở chế độ demo, có thể lưu khoá dạng base64 (không khuyến nghị sản xuất).

Chương 3

Phân tích và thiết kế hệ thống

3.1 Kiến trúc tổng thể

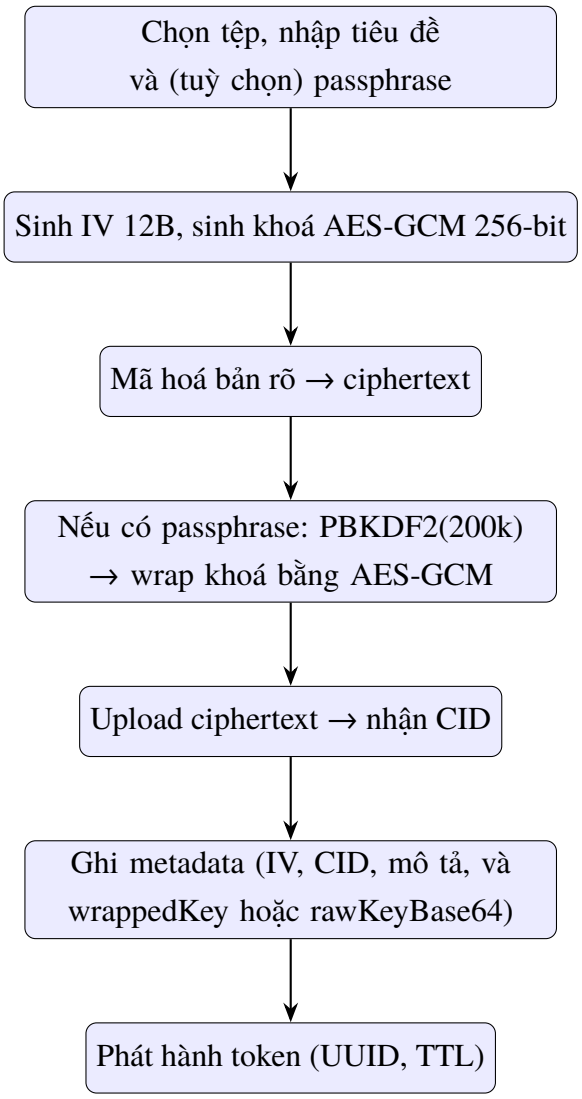


Hình 3.1: Kiến trúc hệ thống và luồng chính

3.2 Thiết kế bảo mật

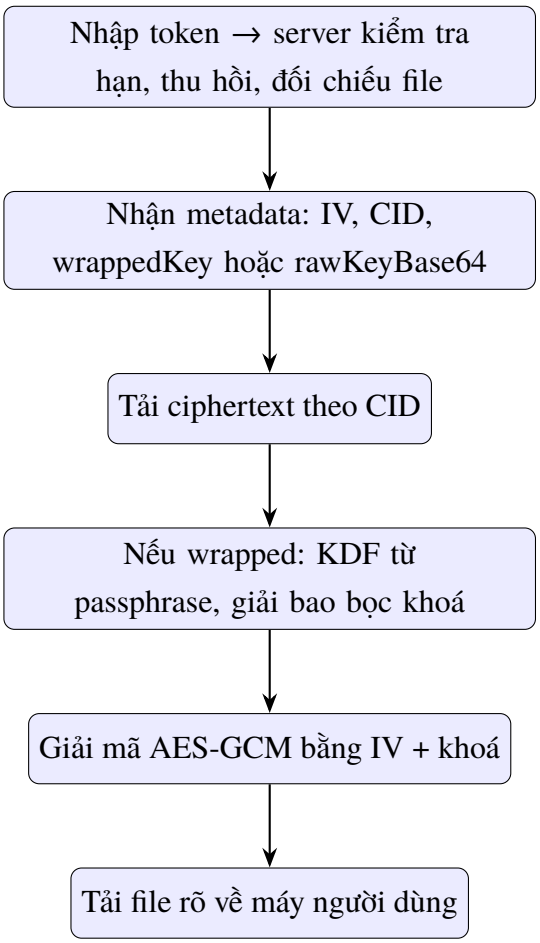
- **Zero-knowledge server**: máy chủ không giữ khoá giải mã.
- **AEAD**: AES-GCM bảo vệ bí mật và toàn vẹn dữ liệu.
- **IV 96-bit ngẫu nhiên**: sinh mới mỗi lần mã hoá.
- **Token TTL + thu hồi**: giảm cửa sổ tấn công.
- **CSRF + SameSite Cookie**: bảo vệ yêu cầu thay đổi trạng thái.
- **Đăng nhập bằng chữ ký**: loại bỏ mật khẩu phía server.

3.3 Sơ đồ luồng Upload



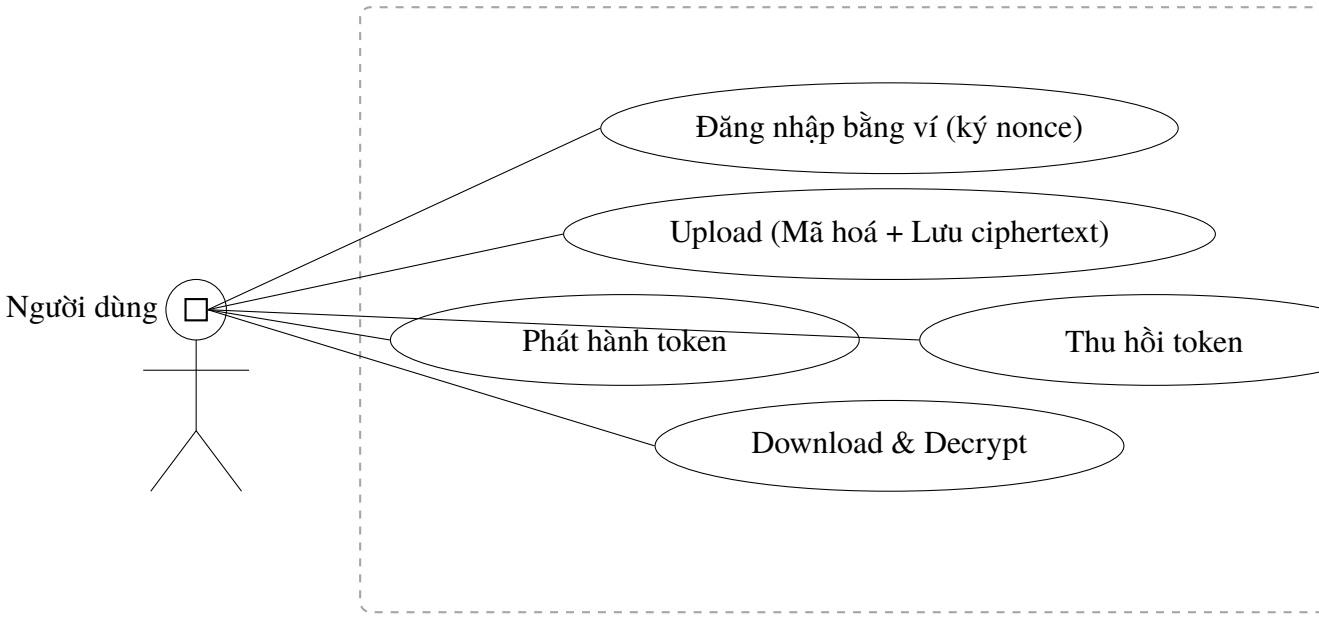
Hình 3.2: Luồng tải lên và phát hành token

3.4 Sơ đồ luồng Download



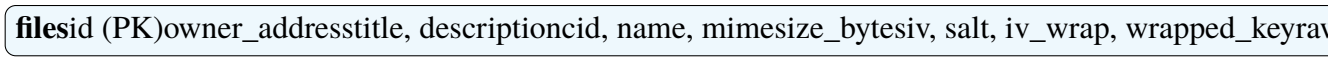
Hình 3.3: Luồng tải xuống và giải mã

3.5 Sơ đồ Use Case



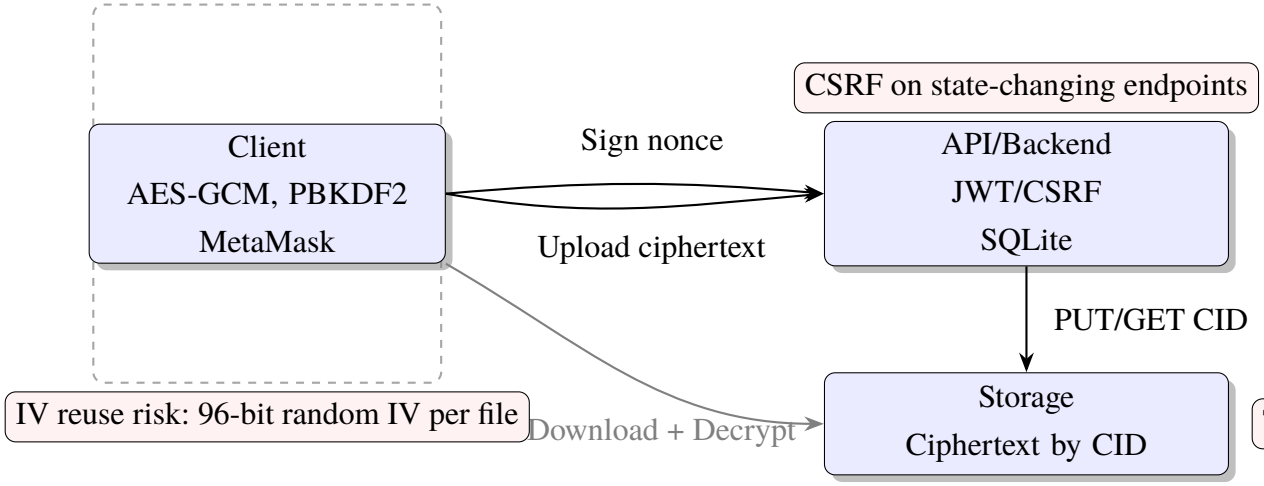
Hình 3.4: Sơ đồ ca sử dụng (Use Case)

3.6 Sơ đồ ERD (rút gọn)



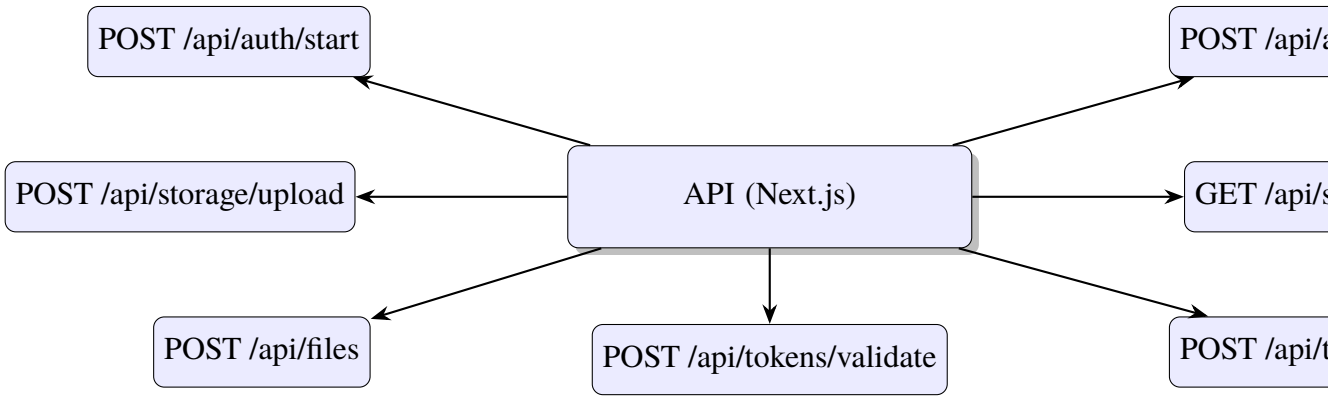
Hình 3.5: Sơ đồ thực thể–quan hệ giữa files và tokens

3.7 DFD / Threat Model tổng quát



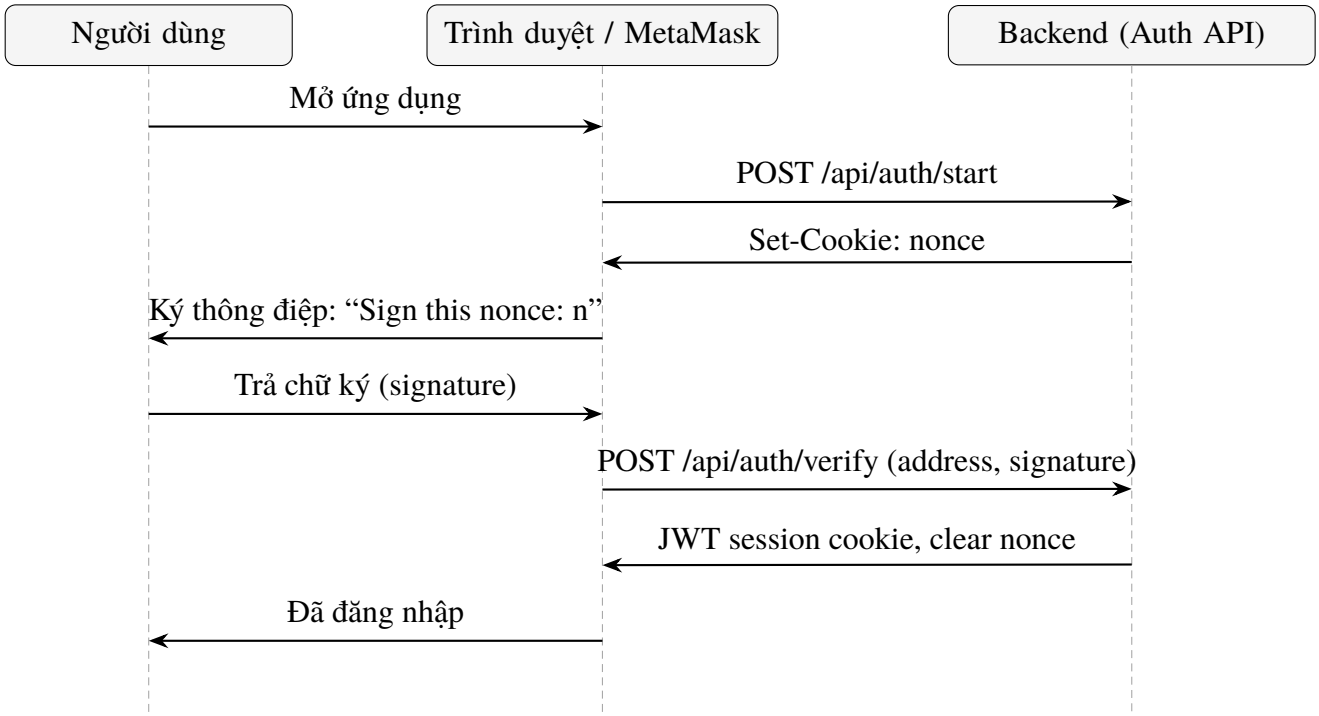
Hình 3.6: DFD và ghi chú đe doạ (CSRF, token leak, IV reuse)

3.8 Bản đồ API



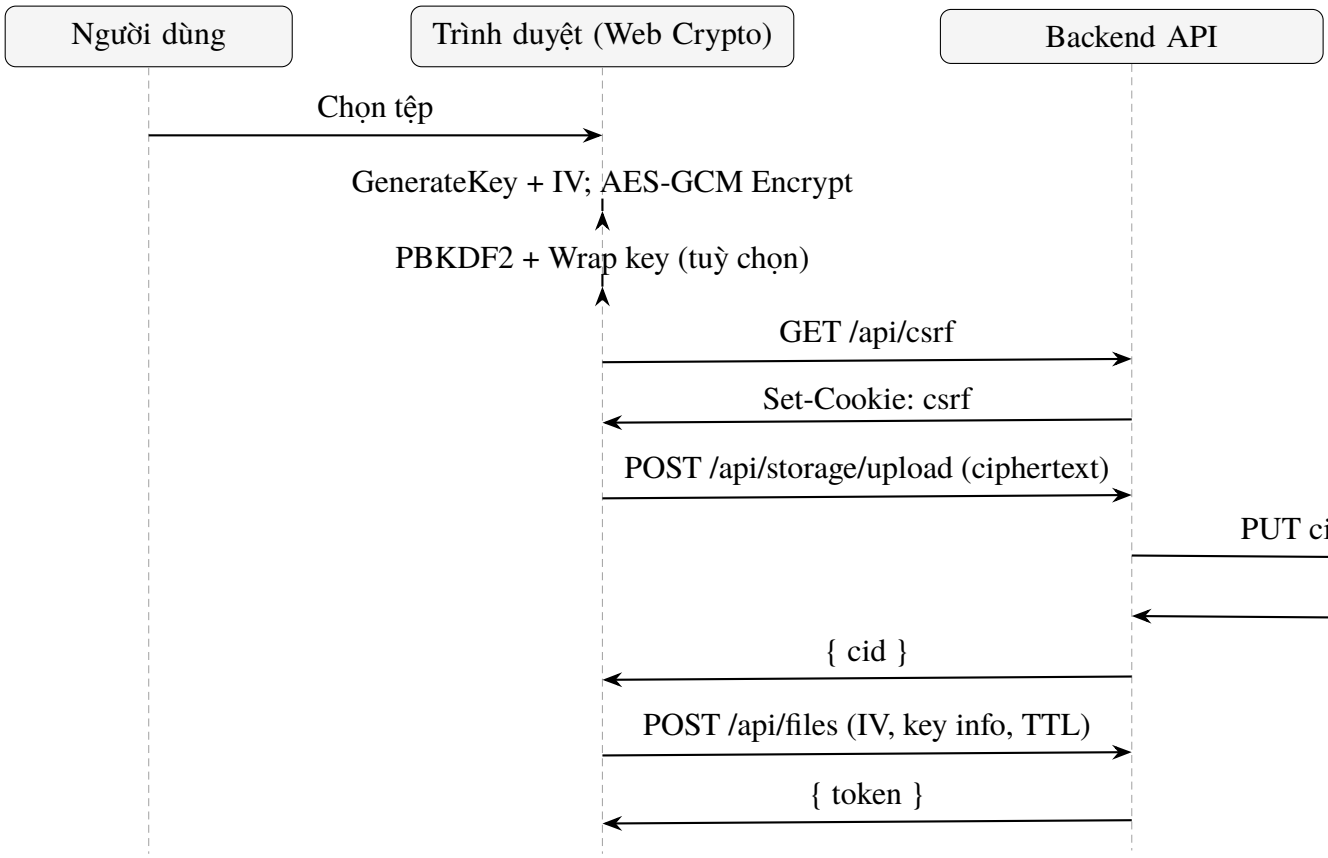
Hình 3.7: Các route API chính

3.9 Sequence: Đăng nhập (ký nonce)



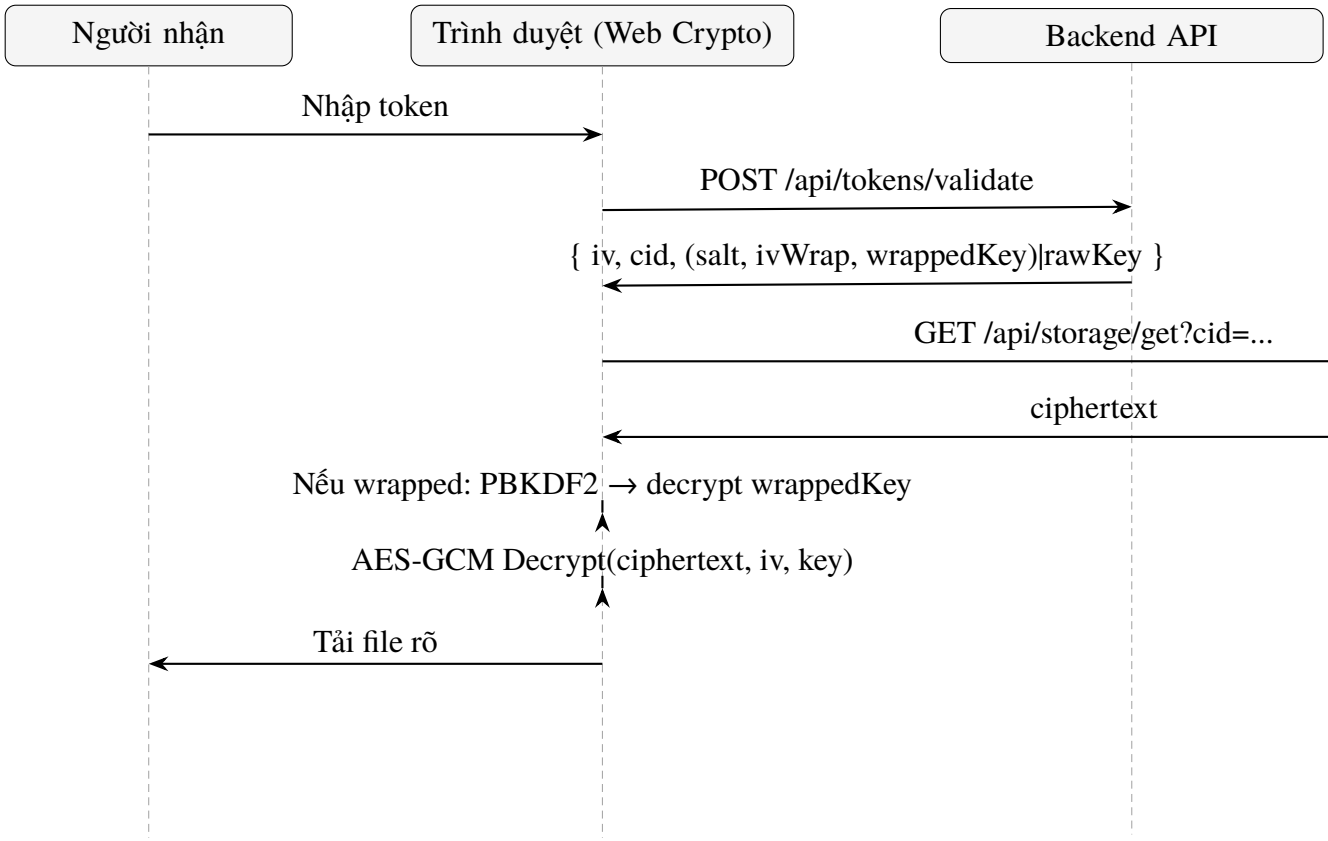
Hình 3.8: Sequence đăng nhập: ký nonce bằng MetaMask

3.10 Sequence: Upload (mã hoá phía client)



Hình 3.9: Sequence upload: mã hoá client-side, lưu ciphertext, phát token

3.11 Sequence: Download (giải mã phía client)



Hình 3.10: Sequence download: xác thực token, tải ciphertext, giải mã trên client

Chương 4

Thuật toán và giải thuật chi tiết

4.1 Mã hoá AES-256-GCM phía client

Algorithm 1 EncryptClient(plain)

1: $iv \leftarrow$ RandomBytes(12)
2: $key \leftarrow$ GenerateKey(AES-GCM, 256)
3: $cipher \leftarrow$ AESGCM.Encrypt($key, iv, plain, AAD = \varnothing$)
4: $rawKey \leftarrow$ ExportKeyRaw(key)
5: **return** $(cipher, iv, rawKey)$

4.2 Bao bọc khoá bằng passphrase

Algorithm 2 WrapKeyPBKDF2(rawKey, pass)

1: $salt \leftarrow$ RandomBytes(16)
2: $base \leftarrow$ PBKDF2(pass, salt, iter=200000, hash=SHA-256)
3: $iv_w \leftarrow$ RandomBytes(12)
4: $wrapped \leftarrow$ AESGCM.Encrypt($base, iv_w, rawKey$)
5: **return** $(salt, iv_w, wrapped)$

4.3 Xác thực bằng chữ ký ví Ethereum

Algorithm 3 LoginMetaMask()

1: Server phát nonce, set cookie `nonce`
2: Client ký thông điệp ``Sign this nonce to login: <nonce>''
3: Server `verifyMessage` → địa chỉ ví; so khớp với yêu cầu
4: Nếu hợp lệ: phát JWT phiên (cookie `session`)

4.4 Cấp và xác thực token

Algorithm 4 IssueAndValidateToken

1: Issue: tạo UUID, `expiresAt`, lưu kèm `fileId`
2: Validate: kiểm tra thu hồi/hết hạn; trả về metadata (IV, CID, key info)

Chương 5

Cài đặt và triển khai

5.1 Công nghệ sử dụng

- **Frontend**: Next.js (App Router), React, Tailwind CSS.
- **Crypto client**: Web Crypto API (AES-GCM, PBKDF2), Ethers.js (ký thông điệp).
- **Backend**: Next.js API routes, JWT (HS256), CSRF double-submit cookie.
- **Lưu trữ**: Ciphertext local theo CID (SHA-256 demo), metadata trong SQLite.

5.2 Cấu trúc dữ liệu chính

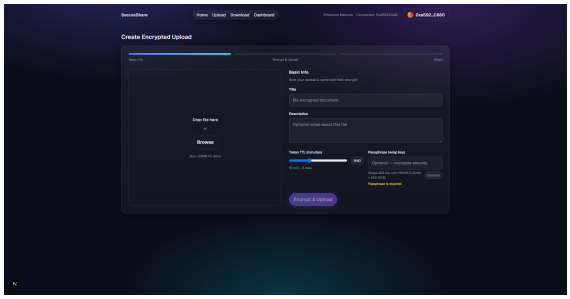
- `files`: id, owner_address, title, description, cid, name, mime, size_bytes, iv, (salt, iv_wrap, wrapped_key | raw_key_base64), created_at.
- `tokens`: token, file_id, issued_to_address, expires_at, revoked, created_at.

5.3 Các API tiêu biểu

- `POST /api/auth/start`: phát nonce; `POST /api/auth/verify`: xác minh chữ ký, set JWT.
- `POST /api/storage/upload`: nhận ciphertext (octet-stream), trả CID.
- `POST /api/files`: lưu metadata (IV, CID, key info), phát token TTL.
- `POST /api/tokens/validate`: trả metadata theo token; `POST /api/tokens/revoke`: thu hồi.

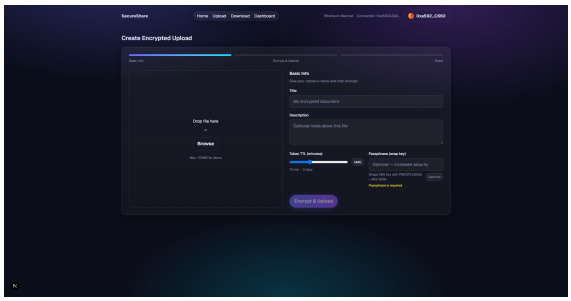
5.4 UI Screenshots

(a) Home Page with Hero Section

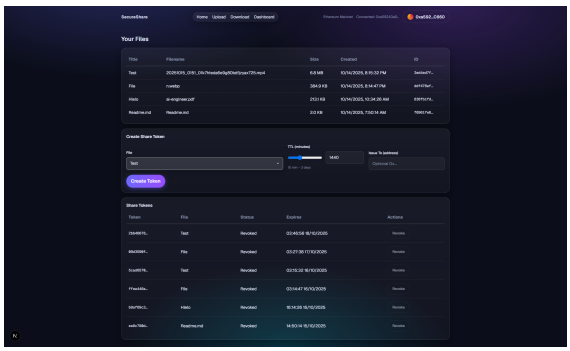


(b) Upload Page with Encryption Wizard

Hình 5.1: Application Interface - Part 1



(a) Download & Decrypt Page



(b) Dashboard with File Management

Hình 5.2: Application Interface - Part 2

5.5 Installation & Setup

Prerequisites

- Node.js v18 or higher
- npm v9 or higher
- MetaMask browser extension (for wallet integration)

Installation Steps

1. Clone or extract the project repository
2. Navigate to the project root directory
3. Run `npm install` to install dependencies
4. Create `.env.local` file with required environment variables
5. Run `npm run dev` to start the development server
6. Open `http://localhost:3000` in your browser

5.6 User Guide

1. Connecting Your Wallet

To use the application, connect your Ethereum wallet (MetaMask):

1. Click the **"Connect Wallet"** button in the top-right corner
2. MetaMask will open; select your account and approve the connection
3. Sign the authentication message (nonce) to verify ownership
4. Your wallet address will appear in the header

2. Uploading and Encrypting a File

To upload a file with encryption:

1. Navigate to the **Upload** page
2. Drag and drop a file or click **Browse** to select one
3. Enter a **Title** and optional **Description**
4. (Optional) Enter a **Passphrase** to wrap the AES key (recommended for production)
5. Adjust **Token TTL** (time-to-live) from 10 minutes to 3 days
6. Click **Encrypt & Upload**
7. The file is encrypted client-side using AES-256-GCM; only ciphertext is sent to the server
8. A share token is generated and displayed

3. Sharing the File

After uploading:

1. Copy the **Share Token** or **Download Link**
2. Send the token/link to the recipient via any communication channel
3. The recipient can use the token to download and decrypt the file
4. Tokens can be revoked at any time from the Dashboard

4. Downloading and Decrypting a File

To decrypt and download a shared file:

1. Navigate to the **Download** page (or use the share link directly)
2. Paste the **Token** into the input field
3. Click **Validate** to check token validity and retrieve file metadata
4. If the file was wrapped with a passphrase, enter it
5. Click **Download & Decrypt** to download the decrypted file
6. The decryption happens entirely in your browser; the server never sees the decrypted content

5. Managing Files and Tokens

On the **Dashboard**:

- View all files you have uploaded (owner only)
- See file details: name, size, upload date
- Issue new tokens for any of your files
- Revoke existing tokens to immediately disable access
- Manage token TTL and optionally restrict tokens to specific addresses

Chương 6

Kiểm thử và đánh giá

6.1 Tiêu chí đánh giá

- **Đúng chức năng**: mã hoá/giải mã chính xác; token hoạt động, TTL/thu hồi hợp lệ.
- **Bảo mật**: không rò rỉ khoá; kiểm tra CSRF; từ chối token sai/hết hạn.
- **Hiệu năng**: thời gian mã hoá/giải mã và tải tệp trong ngưỡng chấp nhận được.

6.2 Phân tích rủi ro và giảm thiểu

- **IV trùng lặp**: sinh ngẫu nhiên 96-bit; tuyệt đối không tái sử dụng IV cùng khoá.
- **Passphrase yếu**: gợi ý độ mạnh, khuyến nghị tối thiểu 12 ký tự, PBKDF2 200k vòng.
- **Rò rỉ khoá (chế độ demo)**: tắt `ALLOW_RAW_KEYS` trong sản xuất; dùng wrappedKey bắt buộc.
- **Tấn công CSRF/XSRF**: header `x-csrf` + cookie cùng giá trị; kiểm tra phía server.

Chương 7

Kết luận và hướng phát triển

7.1 Kết luận

Đề tài đã chứng minh tính khả thi của mô hình chia sẻ tệp bảo mật với mã hoá phía client (client-side encryption). Bằng cách lưu trữ ciphertext trên máy chủ và chỉ giữ metadata, hệ thống đạt được mục tiêu **zero-knowledge**: máy chủ không bao giờ biết được nội dung hoặc khóa giải mã của tệp.

Những thành tựu chính

- **Mã hoá an toàn**: Triển khai thành công AES-256-GCM với Web Crypto API; tất cả khóa được tạo và quản lý phía client.
- **Xác thực không mật khẩu**: Sử dụng chữ ký Ethereum (MetaMask) để xác thực, loại bỏ nhu cầu lưu trữ mật khẩu phía server.
- **Kiểm soát quyền truy cập**: Token-based access control với TTL (time-to-live) và khả năng thu hồi ngay lập tức.
- **Bảo vệ CSRF**: Triển khai double-submit cookie pattern và SameSite cookie để bảo vệ các endpoint thay đổi trạng thái.
- **Giao diện thân thiện**: Ứng dụng web hiện đại với flow trực quan cho upload, tạo token, và download.

Những thách thức gặp phải

- **Tương thích trình duyệt**: Web Crypto API có hỗ trợ tốt nhưng cần kiểm tra polyfill cho các trình duyệt cũ hơn.
- **Hiệu năng**: Mã hoá/giải mã các tệp lớn phía client có thể mất nhiều thời gian; cần tối ưu hóa UI để hiển thị tiến độ.
- **Trải nghiệm người dùng**: Cân bằng giữa bảo mật (yêu cầu passphrase) và dễ sử dụng (tùy chọn passphrase).

- **Lưu trữ**: Triển khai hiện tại sử dụng lưu trữ cục bộ; scaling lên IPFS/Filecoin đòi hỏi nghiên cứu thêm.

Bài học rút ra

- Lựa chọn thuật toán đúng (AES-GCM) quan trọng hơn tối ưu hóa tùy ý; tiêu chuẩn công nghiệp cung cấp độ tin cậy tốt.
- Quản lý khoá là phần khó nhất: cần cân nhắc kỹ cách lưu trữ, sao lưu, và khôi phục.
- Bảo mật phía server (CSRF, input validation, rate limiting) cũng quan trọng như bảo mật phía client.
- Testing bảo mật cần phải toàn diện: unit tests, integration tests, và manual penetration testing.

7.2 Hướng phát triển

Để nâng cấp dự án lên mức độ sản xuất và mở rộng các tính năng, các công việc sau được đề xuất:

Ngắn hạn (1-2 tháng)

1. **Bắt buộc passphrase**: Loại bỏ chế độ demo ALLOW_RAW_KEYS; yêu cầu PBKDF2-wrapped key bắt buộc.
2. **Performance optimization**: Implement Web Worker để mã hoá/giải mã không chặn UI.
3. **Mobile responsiveness**: Tối ưu hoá giao diện cho các thiết bị di động.
4. **Comprehensive testing**: Thêm unit tests, integration tests, và E2E tests với Playwright.

Trung hạn (3-6 tháng)

1. **IPFS integration**: Thay thế lưu trữ cục bộ bằng IPFS hoặc Filecoin để phân tán hóa lưu trữ.
2. **Attribute-Based Access Control (ABAC)**: Hỗ trợ chia sẻ dựa trên thuộc tính (ví dụ: địa chỉ ví, role).
3. **Audit logging**: Ghi lại tất cả các hoạt động (upload, token issuance, revocation) cho mục đích kiểm toán.
4. **Multi-signature support**: Cho phép yêu cầu từ nhiều ví để phê duyệt chia sẻ.

Dài hạn (6+ tháng)

1. **Smart contract integration**: Triển khai hợp đồng thông minh on-chain (Ethereum, Polygon, etc.) để quản lý tokens và quyền truy cập.
2. **Mobile app**: Phát triển ứng dụng di động native cho iOS/Android với xác thực sinh trắc học.
3. **Decentralized naming**: Hỗ trợ ENS (Ethereum Name Service) để thay thế địa chỉ ví dài.
4. **Cross-chain compatibility**: Hỗ trợ xác thực từ các blockchain khác nhau.

Chất lượng & Bảo mật

1. **Security audit**: Nhờ audit bên thứ ba cho code và cryptography.
2. **Bug bounty program**: Mở chương trình bounty để khuyến khích researcher tìm lỗ hổng.
3. **Documentation**: Cải thiện tài liệu API, architecture docs, và security guidelines.
4. **Compliance**: Đảm bảo tuân thủ GDPR, CCPA, và các quy định bảo vệ dữ liệu khác.

Tài liệu tham khảo

[1] NIST FIPS 197. Advanced Encryption Standard (AES), 2001.

[2] NIST SP 800â€‘38D. Galois/Counter Mode (GCM) and GMAC, 2007.

[3] IETF RFC 2898. PKCS #5: PBKDF2, 2000.

[4] IETF RFC 7519. JSON Web Token (JWT), 2015.

[5] W3C. Web Cryptography API, Recommendation.