**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC ĐẠI NAM**

Logo, company name

Description automatically generated

**BÁO CÁO HỌC PHẦN**

**TÊN HỌC PHẦN: NHẬP MÔN AN TOÀN, BẢO MẬT THÔNG TIN**

**ĐỀ TÀI: GIẢ LẬP UPLOAD/DOWNLOAD NỘI DUNG ÂM THANH LÊN CLOUD VỚI PHÁT HIỆN SỬA ĐỔI**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Mã Sinh Viên** | **Họ và Tên** | **Ngày Sinh** | **Lớp** |
| 01 | 1771020545 | Nguyễn Xuân Phúc | 12/01/2005 | CNTT 17-04 |
| 02 | 1771020502 | Nguyễn Thị Hảo Ngân | 28/10/2005 | CNTT 17-04 |
| 03 | 1771020739 | Mạc Thảo Vi | 09/04/2005 | CNTT 17-04 |

**Hà Nội, năm 2025**

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC ĐẠI NAM**

Logo, company name

Description automatically generated

**BÁO CÁO HỌC PHẦN**

**TÊN HỌC PHẦN: NHẬP MÔN AN TOÀN, BẢO MẬT THÔNG TIN**

**ĐỀ TÀI: GIẢ LẬP UPLOAD/DOWNLOAD NỘI DUNG ÂM THANH LÊN CLOUD VỚI PHÁT HIỆN SỬA ĐỔI**

**Giảng viên hướng dẫn: TS Trần Quý Nam**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **STT** | **Mã Sinh Viên** | **Họ và Tên** | **Ngày Sinh** | **Điểm** | |
| **Bằng Số** | **Bằng Chữ** |
| **01** | **1771020545** | **Nguyễn Xuân Phúc** | **12/01/2005** |  |  |
| **02** | **1771020502** | **Nguyễn Thị Hảo Ngân** | **28/10/2005** |  |  |
| **03** | **1771020739** | **Mạc Thảo Vi** | **09/04/2005** |  |  |

CÁN BỘ CHẤM THI 1 CÁN BỘ CHẤM THI 2

**Hà Nội, năm 2025**

**LỜI NÓI ĐẦU**

Trong bối cảnh công nghệ số phát triển mạnh mẽ, việc lưu trữ và chia sẻ nội dung đa phương tiện trên các nền tảng đám mây đã trở thành một phần không thể thiếu trong đời sống hiện đại. Tuy nhiên, quá trình truyền tải dữ liệu qua mạng Internet luôn tiềm ẩn nhiều nguy cơ về bảo mật, như sửa đổi dữ liệu, truy cập trái phép hay mất tính toàn vẹn. Với mong muốn áp dụng các kiến thức đã học trong học phần “Nhập môn An toàn, Bảo mật Thông tin” tại Trường Đại học Đại Nam, nhóm chúng tôi đã thực hiện bài tập cuối kỳ với đề tài “Giả lập upload/download nội dung âm thanh lên cloud với phát hiện sửa đổi”. Báo cáo này trình bày quá trình xây dựng một hệ thống mô phỏng đơn giản, lấy cảm hứng từ nền tảng Spotify, nhằm đảm bảo an toàn dữ liệu trong quá trình truyền tải. Thông qua việc triển khai các thuật toán mã hóa như AES-GCM, RSA và SHA-512, chúng tôi không chỉ củng cố kiến thức lý thuyết mà còn rèn luyện kỹ năng thực hành trong việc thiết kế một hệ thống bảo mật cơ bản. Báo cáo sẽ bao gồm các nội dung chính như phân tích yêu cầu, mô tả thuật toán, cấu trúc mã nguồn, kết quả thử nghiệm và những đề xuất cải tiến, với hy vọng mang lại cái nhìn tổng quan về cách áp dụng các kỹ thuật bảo mật vào thực tiễn.

MỤC LỤC

[**LỜI NÓI ĐẦU** 3](#_Toc202229292)

[CHƯƠNG 1. ĐẶT VẤN ĐỀ VÀ PHÂN TÍCH YÊU CẦU 7](#_Toc202229293)

[1.1. Bối cảnh và mục tiêu 7](#_Toc202229294)

[1.1.1. Tình hình bảo mật khi upload/download file âm thanh trên nền tảng cloud 7](#_Toc202229295)

[1.1.2. Tầm quan trọng của phát hiện sửa đổi trong truyền tải dữ liệu multimedia 8](#_Toc202229296)

[1.1.3. Mục tiêu giả lập hệ thống upload/download podcast.mp3 lên Spotify cloud 8](#_Toc202229297)

[1.2. Phân tích yêu cầu bảo mật 9](#_Toc202229298)

[1.2.1. AES-GCM để đảm bảo tính bảo mật khi truyền tải 9](#_Toc202229299)

[1.2.2. RSA 1024-bit với PKCS#1 v1.5 + SHA-512 cho chữ ký số 10](#_Toc202229300)

[1.2.3. SHA-512 hash và AES-GCM authentication tag 10](#_Toc202229301)

[1.2.4. Sử dụng tag AES-GCM để detect tampering 11](#_Toc202229302)

[1.3. Yêu cầu kỹ thuật 12](#_Toc202229303)

[1.3.1. Hỗ trợ upload và download file qua TCP socket 12](#_Toc202229304)

[1.3.2. Xử lý file podcast.mp3 với các kích thước khác nhau 12](#_Toc202229305)

[CHƯƠNG 2. MÔ TẢ VÀ PHÂN TÍCH THUẬT TOÁN 14](#_Toc202229306)

[2.1. Phân tích các thuật toán mật mã sử dụng 14](#_Toc202229307)

[2.1.1. AES-GCM (Advanced Encryption Standard - Galois/Counter Mode) 14](#_Toc202229308)

[2.1.2. RSA 1024-bit với PKCS#1 v1.5 padding 15](#_Toc202229309)

[2.1.3. SHA-512 (Secure Hash Algorithm) 17](#_Toc202229310)

[2.2. Quy trình tích hợp các thuật toán 17](#_Toc202229311)

[2.2.1. Handshake đơn giản 17](#_Toc202229312)

[2.2.2. Xác thực và Trao đổi khóa 18](#_Toc202229313)

[2.2.3. Truyền dữ liệu và kiểm tra toàn vẹn 18](#_Toc202229314)

[2.3. Phân tích bảo mật của thuật toán tích hợp 19](#_Toc202229315)

[2.3.1. AES-GCM Authentication Tag Verification 19](#_Toc202229316)

[2.3.2. SHA-512 Integrity Check 19](#_Toc202229317)

[2.3.3. Dual-layer Protection Benefits 20](#_Toc202229318)

[2.3.4. Response Protocol 20](#_Toc202229319)

[CHƯƠNG 3. PHÂN TÍCH MÃ NGUỒN VÀ THỬ NGHIỆM 21](#_Toc202229320)

[3.1. Cấu trúc mã nguồn 21](#_Toc202229321)

[3.1.1. Module CryptoUtils (crypto\_utils.py) 21](#_Toc202229322)

[3.1.2. Module Client (client.py) 21](#_Toc202229323)

[3.1.3. Module Server (server.py) 22](#_Toc202229324)

[3.1.4. Module Web Application (app.py) 23](#_Toc202229325)

[3.2. Phân tích các hàm chính 23](#_Toc202229326)

[3.2.1. Hàm xử lý mã hóa và giải mã 23](#_Toc202229327)

[3.2.2. Hàm xử lý chữ ký số 25](#_Toc202229328)

[3.2.3. Hàm xử lý tính toàn vẹn 25](#_Toc202229329)

[3.2.4. Hàm xử lý upload và download 26](#_Toc202229330)

[3.3. Thử nghiệm với file khác nhau 28](#_Toc202229331)

[3.3.1. Thiết kế test cases 28](#_Toc202229332)

[3.3.2. Kết quả thử nghiệm với file kích thước nhỏ 29](#_Toc202229333)

[3.3.3. Kết quả thử nghiệm với file kích thước trung bình 29](#_Toc202229334)

[3.3.4. Kết quả thử nghiệm với file kích thước lớn 30](#_Toc202229335)

[3.3.5. Stress testing và edge cases 31](#_Toc202229336)

[3.4. Ghi nhận kết quả 32](#_Toc202229337)

[CHƯƠNG 4. KẾT QUẢ ĐẠT ĐƯỢC VÀ ĐÁNH GIÁ, ĐỀ XUẤT CẢI TIẾN 33](#_Toc202229338)

[4.1. Kết quả đạt được 33](#_Toc202229339)

[4.2. Đánh giá hiệu quả bảo mật 36](#_Toc202229340)

[4.2.1. Bảo mật tính toàn vẹn dữ liệu 36](#_Toc202229341)

[4.2.2. Bảo mật xác thực và ủy quyền 37](#_Toc202229342)

[4.2.3. Bảo mật bí mật thông tin 38](#_Toc202229343)

[4.3. Đánh giá hiệu suất 38](#_Toc202229344)

[4.3.1. Hiệu suất mã hóa và giải mã 38](#_Toc202229345)

[4.3.2. Hiệu suất truyền tải mạng 39](#_Toc202229346)

[4.3.3. Hiệu suất lưu trữ và truy xuất 39](#_Toc202229347)

[4.4.1. Đặc điểm của AES-GCM 40](#_Toc202229348)

[4.4.2. Đặc điểm của RSA-1024 40](#_Toc202229349)

[4.4.3. Đặc điểm của SHA-512 41](#_Toc202229350)

[4.5. Đề xuất cải tiến 41](#_Toc202229351)

[4.5.1. Nâng cấp độ mạnh thuật toán 41](#_Toc202229352)

[4.5.2. Cải tiến cơ chế session management 42](#_Toc202229353)

[4.5.3. Tối ưu hóa hiệu suất 42](#_Toc202229354)

[4.5.4. Cải tiến logging và monitoring 43](#_Toc202229355)

[4.5.5. Bảo mật ứng dụng web 44](#_Toc202229356)

[KẾT LUẬN 45](#_Toc202229357)

[DANH MỤC TÀI LIỆU THAM KHẢO 46](#_Toc202229358)

# CHƯƠNG 1. ĐẶT VẤN ĐỀ VÀ PHÂN TÍCH YÊU CẦU

## Bối cảnh và mục tiêu

### Tình hình bảo mật khi upload/download file âm thanh trên nền tảng cloud

Trong kỷ nguyên số hiện tại, việc lưu trữ và chia sẻ nội dung đa phương tiện trên các nền tảng cloud đã trở thành xu hướng phổ biến. Các dịch vụ streaming âm thanh như Spotify, Apple Music, Amazon Music đã thu hút hàng triệu người dùng với khả năng truy cập nội dung mọi lúc, mọi nơi. Tuy nhiên, việc truyền tải dữ liệu âm thanh qua mạng Internet đặt ra nhiều thách thức về bảo mật thông tin.

Các mối đe dọa chính trong quá trình upload/download file âm thanh bao gồm:

* **Tấn công Man-in-the-Middle (MITM):** Kẻ tấn công có thể chặn bắt dữ liệu trong quá trình truyền tải, đọc được nội dung file âm thanh hoặc thay đổi dữ liệu trước khi chuyển đến đích.
* **Tampering (Sửa đổi dữ liệu):** Dữ liệu có thể bị thay đổi một cách bất hợp pháp trong quá trình truyền tải, dẫn đến việc nhận được file âm thanh bị hỏng hoặc chứa nội dung độc hại.
* **Replay Attack:** Kẻ tấn công có thể ghi lại và phát lại các gói tin đã được truyền trước đó, gây ra việc upload trùng lặp hoặc truy cập trái phép.
* **Data Integrity Loss:** Mất tính toàn vẹn dữ liệu do lỗi mạng hoặc tác động bên ngoài, làm cho file âm thanh bị lỗi khi phát lại.
* **Unauthorized Access:** Người dùng không có quyền có thể truy cập vào file âm thanh của người khác, vi phạm quyền riêng tư và bản quyền.

### Tầm quan trọng của phát hiện sửa đổi trong truyền tải dữ liệu multimedia

Dữ liệu multimedia, đặc biệt là file âm thanh, có những đặc điểm riêng biệt đòi hỏi các biện pháp bảo mật đặc thù. Khác với dữ liệu văn bản thông thường, file âm thanh có kích thước lớn và cấu trúc phức tạp, khiến việc phát hiện sửa đổi trở nên khó khăn hơn.

Tầm quan trọng của việc phát hiện sửa đổi:

* **Bảo vệ chất lượng nội dung:** Ngay cả những thay đổi nhỏ trong dữ liệu âm thanh cũng có thể làm giảm chất lượng âm thanh hoặc tạo ra tiếng ồn không mong muốn. Việc phát hiện sớm các sửa đổi giúp đảm bảo chất lượng nội dung được duy trì.
* **Bảo vệ bản quyền:** File âm thanh thường chứa nội dung có bản quyền. Việc phát hiện sửa đổi giúp ngăn chặn việc thay đổi metadata, thông tin tác giả hoặc chèn nội dung bất hợp pháp.
* **Đảm bảo tính xác thực:** Trong môi trường thương mại, việc đảm bảo file âm thanh không bị thay đổi từ khi upload đến khi download là cực kỳ quan trọng để duy trì niềm tin của người dùng.
* **Tuân thủ quy định pháp lý:** Nhiều quốc gia có quy định nghiêm ngặt về việc bảo vệ nội dung số và quyền riêng tư. Hệ thống phát hiện sửa đổi giúp tuân thủ các quy định này.
* **Phòng chống tấn công độc hại:** Kẻ tấn công có thể chèn mã độc hoặc nội dung có hại vào file âm thanh. Việc phát hiện sửa đổi giúp ngăn chặn các cuộc tấn công này.

### Mục tiêu giả lập hệ thống upload/download podcast.mp3 lên Spotify cloud

Nghiên cứu này nhằm mục đích xây dựng một hệ thống giả lập hoàn chỉnh mô phỏng quá trình upload và download file podcast.mp3 lên nền tảng cloud tương tự Spotify. Hệ thống được thiết kế với các mục tiêu cụ thể sau:

**Mục tiêu chính:**

* Xây dựng hệ thống truyền tải file âm thanh an toàn qua giao thức TCP socket
* Triển khai các cơ chế bảo mật hiện đại để bảo vệ dữ liệu trong quá trình truyền tải
* Phát triển khả năng phát hiện và xử lý các cuộc tấn công sửa đổi dữ liệu
* Đánh giá hiệu quả của các thuật toán mã hóa trong bối cảnh truyền tải dữ liệu multimedia

**Mục tiêu phụ:**

* Mô phỏng môi trường thực tế của các dịch vụ streaming âm thanh
* Cung cấp giao diện người dùng thân thiện để tương tác với hệ thống
* Tạo ra các kịch bản tấn công để kiểm tra khả năng phòng thủ của hệ thống
* Thu thập dữ liệu về hiệu suất và độ tin cậy của các thuật toán bảo mật

## Phân tích yêu cầu bảo mật

### AES-GCM để đảm bảo tính bảo mật khi truyền tải

**Advanced Encryption Standard - Galois/Counter Mode (AES-GCM)** được lựa chọn làm thuật toán mã hóa chính cho hệ thống với những ưu điểm vượt trội:

Đặc điểm kỹ thuật của AES-GCM:

**Authenticated Encryption:** AES-GCM kết hợp cả mã hóa và xác thực trong một thuật toán duy nhất, đảm bảo vừa tính bảo mật vừa tính toàn vẹn của dữ liệu.

**High Performance:** Chế độ GCM được tối ưu hóa cho việc xử lý song song, cho phép mã hóa/giải mã với tốc độ cao, phù hợp với file âm thanh có kích thước lớn.

**Nonce-based Security:** Sử dụng nonce (number used once) để đảm bảo mỗi lần mã hóa tạo ra kết quả khác nhau, ngay cả khi dữ liệu đầu vào giống nhau.

**Authentication Tag:** Tạo ra authentication tag 16-byte để xác minh tính toàn vẹn và xác thực của dữ liệu mã hóa.

Yêu cầu bảo mật được đáp ứng:

* **Confidentiality (Tính bảo mật):** Dữ liệu được mã hóa bằng khóa đối xứng 256-bit, đảm bảo không ai có thể đọc được nội dung file âm thanh khi chặn bắt dữ liệu truyền tải.
* **Integrity (Tính toàn vẹn):** Authentication tag đảm bảo phát hiện được mọi thay đổi trong dữ liệu mã hóa.
* **Authenticity (Tính xác thực):** Chỉ những ai có khóa đúng mới có thể tạo ra dữ liệu mã hóa hợp lệ.

### RSA 1024-bit với PKCS#1 v1.5 + SHA-512 cho chữ ký số

Hệ thống xác thực công khai RSA được triển khai để đảm bảo tính xác thực và không thể từ chối trong hệ thống:

**Thông số kỹ thuật:**

* **RSA 1024-bit Key Size:** Sử dụng khóa 1024-bit cung cấp mức độ bảo mật phù hợp cho môi trường giả lập, đồng thời đảm bảo hiệu suất xử lý tốt.
* **PKCS#1 v1.5 Padding:** Áp dụng chuẩn padding PKCS#1 v1.5 để đảm bảo tính an toàn của quá trình ký số và xác minh chữ ký.
* **SHA-512 Hashing:** Sử dụng hàm băm SHA-512 để tạo ra message digest trước khi ký, đảm bảo tính toàn vẹn của dữ liệu và khả năng chống va chạm hash.

**Chức năng xác thực:**

* **Digital Signature Generation:** Tạo chữ ký số cho metadata file (tên file, timestamp, kích thước) để đảm bảo tính xác thực của người gửi.
* **Key Exchange:** Mã hóa session key bằng khóa công khai RSA để truyền tải an toàn khóa đối xứng AES.
* **Identity Verification:** Xác minh danh tính của người dùng thông qua việc kiểm tra chữ ký số.
* **Non-repudiation:** Đảm bảo người gửi không thể từ chối việc đã gửi dữ liệu.

### SHA-512 hash và AES-GCM authentication tag

Hệ thống triển khai cơ chế kiểm tra tính toàn vẹn đa lớp để đảm bảo độ tin cậy cao:

**SHA-512 Hash Function:**

* Cryptographic Properties: SHA-512 cung cấp độ dài hash 512-bit, đảm bảo khả năng chống va chạm cao và bảo mật dài hạn.
* Data Integrity Verification: Tính toán hash của toàn bộ gói tin (nonce || ciphertext || tag) để kiểm tra tính toàn vẹn của dữ liệu truyền tải.
* Tamper Detection: Phát hiện mọi thay đổi nhỏ nhất trong dữ liệu thông qua sự thay đổi của giá trị hash.

**AES-GCM Authentication Tag:**

* Built-in Authentication: Authentication tag được tạo tự động trong quá trình mã hóa AES-GCM, cung cấp khả năng xác thực tích hợp.
* 16-byte Tag Size: Sử dụng tag kích thước 16-byte (128-bit) để đảm bảo độ bảo mật cao.
* Cryptographic Binding: Tag được liên kết mật mã với nonce, ciphertext và khóa, đảm bảo không thể giả mạo.

**Cơ chế kiểm tra đa lớp:**

* Layer 1: Kiểm tra SHA-512 hash của toàn bộ gói tin
* Layer 2: Xác minh AES-GCM authentication tag
* Layer 3: Kiểm tra chữ ký số RSA của metadata

### Sử dụng tag AES-GCM để detect tampering

**Cơ chế phát hiện sửa đổi tiên tiến:**

* Real-time Detection: AES-GCM authentication tag cho phép phát hiện sửa đổi ngay lập tức trong quá trình giải mã, không cần bước kiểm tra riêng biệt.
* Granular Detection: Có thể phát hiện sửa đổi ở mức độ bit, đảm bảo độ chính xác cao trong việc phát hiện tấn công.
* Automatic Rejection: Hệ thống tự động từ chối dữ liệu khi phát hiện sửa đổi, ngăn chặn việc xử lý dữ liệu bị nhiễm.

**Attack Scenarios được phát hiện:**

* Bit Flipping: Thay đổi giá trị bit trong ciphertext
* Block Reordering: Sắp xếp lại thứ tự các block dữ liệu
* Nonce Manipulation: Thay đổi giá trị nonce
* Tag Forgery: Giả mạo authentication tag

## Yêu cầu kỹ thuật

### Hỗ trợ upload và download file qua TCP socket

**Kiến trúc mạng TCP/IP:**

* Protocol Selection: TCP được lựa chọn vì tính tin cậy cao, đảm bảo dữ liệu được truyền tải đầy đủ và đúng thứ tự, phù hợp với file âm thanh yêu cầu tính toàn vẹn cao.
* Connection Management: Thiết lập và quản lý kết nối persistent để hỗ trợ truyền tải file lớn và multiple sessions.
* Error Handling: Triển khai cơ chế xử lý lỗi mạng, timeout, và retry mechanism để đảm bảo độ tin cậy.
* Buffer Management: Tối ưu hóa buffer size cho việc truyền tải file âm thanh với hiệu suất cao.

**Tính năng upload:**

* Chunked transfer để hỗ trợ file kích thước lớn
* Progress tracking và resume capability
* Metadata validation trước khi upload
* Automatic retry mechanism

**Tính năng download:**

* Stream-based download để tiết kiệm bộ nhớ
* Integrity verification trong quá trình download
* Range request support cho partial download
* Bandwidth throttling để tối ưu mạng

### Xử lý file podcast.mp3 với các kích thước khác nhau

**File Format Support:**

* MP3 Format Compatibility: Hỗ trợ đầy đủ các variant của định dạng MP3, bao gồm các bitrate và sample rate khác nhau.
* Metadata Preservation: Đảm bảo metadata của file MP3 (ID3 tags, album art, v.v.) được bảo toàn trong quá trình mã hóa/giải mã.
* File Size Optimization: Tối ưu hóa quá trình xử lý cho các file có kích thước từ vài MB đến vài GB.
* Memory Management: Sử dụng streaming approach để xử lý file lớn mà không gây quá tải bộ nhớ.

**Kịch bản test với các kích thước file:**

* Small files (< 10MB): Podcast ngắn, demo files
* Medium files (10MB - 100MB): Podcast tiêu chuẩn 30-60 phút
* Large files (100MB - 500MB): Podcast dài, chất lượng cao
* Very large files (> 500MB): Nội dung premium, lossless audio

# 

# CHƯƠNG 2. **MÔ TẢ VÀ PHÂN TÍCH THUẬT TOÁN**

## 2.1. Phân tích các thuật toán mật mã sử dụng

### 2.1.1. AES-GCM (Advanced Encryption Standard - Galois/Counter Mode)

**Nguyên lý hoạt động:**

RSA là hệ thống mã hóa khóa công khai dựa trên bài toán phân tích thừa số nguyên tố lớn. Trong hệ thống này, RSA 1024-bit được sử dụng với modulus n là tích của hai số nguyên tố p và q, mỗi số có độ dài 512-bit. PKCS#1 v1.5 padding được áp dụng để chống lại các cuộc tấn công chosen plaintext và đảm bảo tính ngẫu nhiên của dữ liệu được mã hóa.

Việc tạo khóa RSA được thực hiện thông qua thư viện PyCryptodome với RSA.generate(1024), tạo ra cặp khóa public/private được sử dụng cho cả việc ký số và mã hóa. Public key được chia sẻ với các bên tham gia trong quá trình trao đổi khóa, trong khi private key được bảo mật nghiêm ngặt tại mỗi endpoint.

**Ưu điểm:**

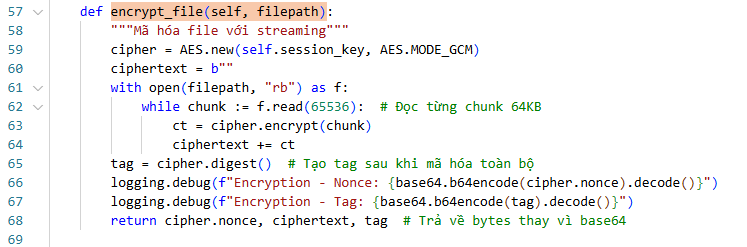
RSA cung cấp tính chất non-repudiation quan trọng thông qua digital signature, đảm bảo rằng người ký không thể phủ nhận việc ký tài liệu. Thuật toán có độ tương thích cao và được hỗ trợ rộng rãi trên nhiều platform khác nhau. Khả năng mã hóa bất đối xứng cho phép trao đổi khóa an toàn mà không cần chia sẻ khóa bí mật trước.

**Nhược điểm:**

Computational overhead của RSA rất cao so với các thuật toán đối xứng, đặc biệt là trong việc xử lý dữ liệu có kích thước lớn. Key size 1024-bit hiện tại không còn được coi là đủ mạnh cho long-term security, các tiêu chuẩn hiện đại khuyến nghị sử dụng ít nhất 2048-bit. Tốc độ xử lý chậm khiến RSA không phù hợp cho việc mã hóa trực tiếp các file có kích thước lớn.

**Áp dụng trong hệ thống:**

RSA được sử dụng cho hai mục đích chính trong hệ thống. Thứ nhất là ký số metadata bao gồm filename, timestamp và file size thông qua method sign\_metadata() sử dụng SHA-512 hash và PKCS#1 v1.5 signature scheme. Thứ hai là mã hóa session key của AES-GCM thông qua method encrypt\_session\_key(), đảm bảo việc trao đổi khóa được thực hiện an toàn qua kênh truyền không đáng tin cậy.

****

### 2.1.2. RSA 1024-bit với PKCS#1 v1.5 padding

**Nguyên lý hoạt động:**

RSA là hệ thống mã hóa khóa công khai dựa trên bài toán phân tích thừa số nguyên tố lớn. Trong hệ thống này, RSA 1024-bit được sử dụng với modulus n là tích của hai số nguyên tố p và q, mỗi số có độ dài 512-bit. PKCS#1 v1.5 padding được áp dụng để chống lại các cuộc tấn công chosen plaintext và đảm bảo tính ngẫu nhiên của dữ liệu được mã hóa.

Việc tạo khóa RSA được thực hiện thông qua thư viện PyCryptodome với RSA.generate(1024), tạo ra cặp khóa public/private được sử dụng cho cả việc ký số và mã hóa. Public key được chia sẻ với các bên tham gia trong quá trình trao đổi khóa, trong khi private key được bảo mật nghiêm ngặt tại mỗi endpoint.

**Ưu điểm:**

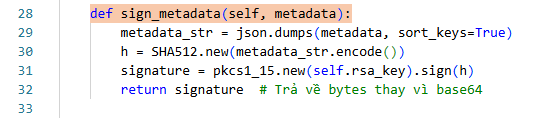
RSA cung cấp tính chất non-repudiation quan trọng thông qua digital signature, đảm bảo rằng người ký không thể phủ nhận việc ký tài liệu. Thuật toán có độ tương thích cao và được hỗ trợ rộng rãi trên nhiều platform khác nhau. Khả năng mã hóa bất đối xứng cho phép trao đổi khóa an toàn mà không cần chia sẻ khóa bí mật trước.

**Nhược điểm:**

Computational overhead của RSA rất cao so với các thuật toán đối xứng, đặc biệt là trong việc xử lý dữ liệu có kích thước lớn. Key size 1024-bit hiện tại không còn được coi là đủ mạnh cho long-term security, các tiêu chuẩn hiện đại khuyến nghị sử dụng ít nhất 2048-bit. Tốc độ xử lý chậm khiến RSA không phù hợp cho việc mã hóa trực tiếp các file có kích thước lớn.

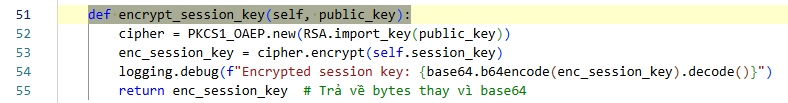
**Áp dụng trong hệ thống:**

RSA được sử dụng cho hai mục đích chính trong hệ thống. Thứ nhất là ký số metadata bao gồm filename, timestamp và file size thông qua method sign\_metadata() sử dụng SHA-512 hash và PKCS#1 v1.5 signature scheme. Thứ hai là mã hóa session key của AES-GCM thông qua method encrypt\_session\_key(), đảm bảo việc trao đổi khóa được thực hiện an toàn qua kênh truyền không đáng tin cậy.



**Session Key Exchange Mechanism:**

AES session key 256-bit được mã hóa bằng RSA public key của server trước khi truyền qua network:



**Đánh giá độ bảo mật:**

RSA 1024-bit hiện tại được coi là không đủ mạnh cho long-term security theo các tiêu chuẩn hiện đại. NIST và các tổ chức bảo mật khuyến nghị sử dụng RSA 2048-bit trở lên. Tuy nhiên, trong bối cảnh simulation và giáo dục, RSA 1024-bit vẫn đủ để minh họa các khái niệm cryptographic cơ bản.

### 2.1.3. SHA-512 (Secure Hash Algorithm)

**Nguyên lý hoạt động:**

SHA-512 là hàm băm mật mã thuộc họ SHA-2, được thiết kế để tạo ra hash digest có độ dài cố định 512-bit từ input có độ dài bất kỳ. Thuật toán sử dụng cấu trúc Merkle-Damgård với compression function hoạt động trên các block 1024-bit. Quá trình băm được thực hiện thông qua 80 rounds của các phép toán logic và số học, đảm bảo tính khuếch tán và confusion cao.

Trong implementation, SHA-512 được sử dụng thông qua thư viện PyCryptodome với SHA512.new(), cho phép xử lý dữ liệu theo từng chunk và cập nhật hash state liên tục. Điều này đặc biệt hữu ích khi xử lý các file có kích thước lớn mà không cần load toàn bộ vào memory.

**Tính chất bảo mật:**

SHA-512 đảm bảo ba tính chất bảo mật cơ bản của hàm băm mật mã. Pre-image resistance đảm bảo việc tìm ngược input từ hash value là không khả thi về mặt tính toán. Second pre-image resistance ngăn chặn việc tìm ra input khác có cùng hash value với một input đã cho. Collision resistance đảm bảo việc tìm ra hai input khác nhau có cùng hash value là cực kỳ khó khăn với độ phức tạp tính toán 2^256.

**Áp dụng trong hệ thống:**

SHA-512 được sử dụng trong hai bối cảnh quan trọng. Đầu tiên là tạo hash cho RSA signature thông qua việc băm metadata trước khi ký, đảm bảo tính toàn vẹn của thông tin metadata. Thứ hai là kiểm tra toàn vẹn của toàn bộ gói tin thông qua việc tạo hash từ concatenation của nonce, ciphertext và authentication tag, cung cấp một lớp bảo vệ bổ sung ngoài AES-GCM tag.

## 2.2. Quy trình tích hợp các thuật toán

### 2.2.1. Handshake đơn giản

Quá trình handshake được thiết kế để thiết lập kết nối TCP ổn định giữa client và server trước khi thực hiện các giao dịch mật mã phức tạp. Client khởi tạo kết nối bằng cách gửi message "Hello!" qua TCP socket tới Spotify server giả lập tại địa chỉ localhost:12345. Server phản hồi với message "Ready!" để xác nhận rằng hệ thống đã sẵn sàng nhận và xử lý các request upload/download.

Cơ chế retry được triển khai với tối đa 3 lần thử và delay 1 giây giữa các lần thử, đảm bảo khả năng phục hồi khi gặp sự cố mạng tạm thời. Socket được cấu hình với timeout 10 giây và keepalive để duy trì kết nối ổn định. Quá trình handshake thành công là điều kiện tiên quyết cho tất cả các giao dịch tiếp theo trong hệ thống.

### 2.2.2. Xác thực và Trao đổi khóa

Giai đoạn xác thực và trao đổi khóa là bước quan trọng nhất trong quy trình bảo mật. Client bắt đầu bằng việc tạo metadata chứa thông tin file bao gồm filename, timestamp và file size. Metadata này được serialize thành JSON string và được ký bằng private key của client sử dụng RSA/SHA-512 signature scheme.

Session key 256-bit được tạo ngẫu nhiên cho mỗi phiên làm việc và được mã hóa bằng public key của server sử dụng RSA-OAEP. Việc trao đổi public key được thực hiện thông qua một kênh riêng biệt với request "GET\_PUBLIC\_KEY", đảm bảo client có được đúng public key của server trước khi thực hiện mã hóa session key.

Server sau khi nhận được encrypted session key sẽ giải mã bằng private key của mình và sử dụng session key này cho toàn bộ quá trình mã hóa/giải mã AES-GCM trong phiên làm việc. Cơ chế này đảm bảo rằng mỗi phiên có một session key duy nhất, ngăn chặn các cuộc tấn công replay và key reuse.

### 2.2.3. Truyền dữ liệu và kiểm tra toàn vẹn

Quá trình truyền dữ liệu được thiết kế với nhiều lớp bảo mật để đảm bảo tính toàn vẹn và xác thực. Đầu tiên, file được mã hóa bằng AES-GCM với session key, tạo ra ciphertext và authentication tag. Nonce được tạo tự động bởi AES-GCM cipher để đảm bảo tính duy nhất.

Sau khi có ciphertext và tag, hệ thống tính toán SHA-512 hash của concatenation nonce||ciphertext||tag để tạo thêm một lớp kiểm tra toàn vẹn. Toàn bộ gói tin bao gồm nonce, ciphertext, tag, hash, signature và metadata được đóng gói thành JSON và truyền qua TCP socket với length-prefixed protocol.

Server nhận gói tin sẽ thực hiện verification theo thứ tự ngược lại: kiểm tra signature của metadata, verify hash của toàn bộ gói tin, và cuối cùng là verify AES-GCM tag. Chỉ khi tất cả các bước verification đều pass, server mới tiến hành giải mã và lưu file. Cơ chế tương tự được áp dụng cho quá trình download với server đóng vai trò sender và client đóng vai trò receiver.

## 2.3. Phân tích bảo mật của thuật toán tích hợp

### 2.3.1. AES-GCM Authentication Tag Verification

Authentication tag của AES-GCM đóng vai trò then chốt trong việc phát hiện mọi sự sửa đổi đối với dữ liệu. Tag được tạo ra thông qua phép toán trong Galois field dựa trên nonce, ciphertext và session key. Mọi thay đổi dù chỉ là một bit trong ciphertext hoặc nonce sẽ dẫn đến tag verification failure.

Trong hệ thống, việc verify tag được thực hiện thông qua method decrypt\_file() của CryptoUtils class. Nếu tag không khớp, exception sẽ được throw và file sẽ bị reject. Điều này đảm bảo rằng không có dữ liệu bị sửa đổi nào có thể được chấp nhận bởi hệ thống. Cơ chế simulate attack được triển khai để test khả năng phát hiện bằng cách intentionally modify một byte cuối của ciphertext.

Khả năng phát hiện của AES-GCM tag có xác suất thành công rất cao, với probability of undetected modification là 2^-128 cho mỗi lần thử. Điều này có nghĩa là kẻ tấn công sẽ cần trung bình 2^127 lần thử để có thể tạo ra một modification không bị phát hiện, điều này là không khả thi về mặt tính toán.

### 2.3.2. SHA-512 Integrity Check

Lớp kiểm tra toàn vẹn SHA-512 cung cấp thêm một lớp bảo vệ bổ sung ngoài AES-GCM tag. Hash được tính toán từ concatenation của nonce, ciphertext và tag, tạo ra một fingerprint duy nhất cho toàn bộ gói tin. Mọi sự thay đổi trong bất kỳ component nào sẽ dẫn đến hash mismatch.

Việc sử dụng SHA-512 thay vì các hash function yếu hơn như MD5 hoặc SHA-1 đảm bảo tính kháng collision cao và security margin lớn. Output 512-bit của SHA-512 cung cấp security level tương đương 256-bit, vượt xa các yêu cầu bảo mật hiện tại và tương lai gần.

Cơ chế dual verification với cả AES-GCM tag và SHA-512 hash tạo ra defense in depth strategy. Ngay cả trong trường hợp một trong hai cơ chế bị bypass (điều cực kỳ khó xảy ra), cơ chế còn lại vẫn có thể phát hiện và ngăn chặn cuộc tấn công.

### 2.3.3. Dual-layer Protection Benefits

Việc kết hợp AES-GCM authentication và SHA-512 integrity check tạo ra một hệ thống bảo mật robust với khả năng chống chịu cao. Hai lớp bảo vệ này hoạt động độc lập nhưng bổ sung cho nhau, tạo ra redundancy trong security mechanism.

AES-GCM tag được tính toán dựa trên cryptographic operations trong Galois field, trong khi SHA-512 hash sử dụng compression function hoàn toàn khác biệt. Điều này có nghĩa là một cuộc tấn công thành công phải đồng thời bypass cả hai cơ chế với các mathematical foundations khác nhau, làm tăng độ khó exponentially.

Ngoài ra, việc sử dụng hai algorithms khác nhau cũng giúp hệ thống có khả năng chống chịu tốt hơn đối với các vulnerability có thể được phát hiện trong tương lai. Nếu một trong hai algorithms bị phát hiện có weakness, algorithm còn lại vẫn có thể duy trì security của hệ thống.

### 2.3.4. Response Protocol

Giao thức phản hồi được thiết kế để cung cấp feedback rõ ràng về trạng thái xử lý và kết quả verification. Hệ thống sử dụng ACK/NACK protocol với các message code cụ thể để chỉ rõ lý do thành công hoặc thất bại.

Các response codes được định nghĩa bao gồm "ACK: File saved" cho upload thành công, "NACK: Invalid signature" cho signature verification failure, "NACK: Hash mismatch" cho integrity check failure, và "NACK: Tag mismatch" cho AES-GCM authentication failure. Mỗi response cũng bao gồm timing information để monitoring performance.

Việc cung cấp specific error codes giúp debugging và monitoring hệ thống hiệu quả hơn, đồng thời cũng cho phép client thực hiện appropriate error handling. Tuy nhiên, cần lưu ý rằng việc cung cấp quá nhiều thông tin chi tiết về lỗi có thể tạo ra information leakage cho kẻ tấn công, do đó cần cân bằng giữa usability và security.

# CHƯƠNG 3. PHÂN TÍCH MÃ NGUỒN VÀ THỬ NGHIỆM

## 3.1. Cấu trúc mã nguồn

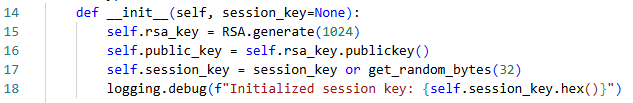
Hệ thống upload/download nội dung âm thanh lên cloud mà chúng em đã xây dựng được tổ chức thành 4 module chính, mỗi module đảm nhận một chức năng cụ thể trong quy trình bảo mật và truyền tải dữ liệu:

### 3.1.1. Module CryptoUtils (crypto\_utils.py)

Module này đóng vai trò là trái tim của hệ thống bảo mật, cung cấp tất cả các chức năng mã hóa và xác thực cần thiết. Module được thiết kế theo nguyên tắc Single Responsibility Principle, tập trung hoàn toàn vào các tác vụ mật mã học.

Cấu trúc class CryptoUtils bao gồm các thuộc tính quan trọng:

* rsa\_key: Khóa RSA private 1024-bit được tạo tự động
* public\_key: Khóa RSA public tương ứng
* session\_key: Khóa phiên AES 256-bit cho mã hóa đối xứng



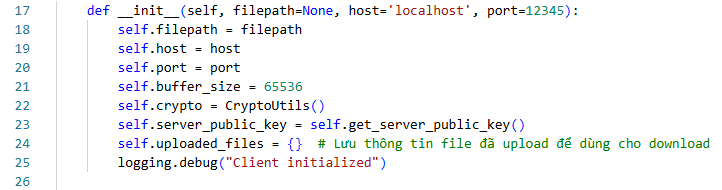
Module cung cấp các phương thức bảo mật chính như sign\_metadata() để ký số metadata bằng RSA/SHA-512, verify\_signature() để xác minh chữ ký, encrypt\_file() và decrypt\_file() cho mã hóa/giải mã file sử dụng AES-GCM, cùng với compute\_hash() để tính toán hash SHA-512 cho việc kiểm tra tính toàn vẹn.

### 3.1.2. Module Client (client.py)

Module Client đóng vai trò là giao diện người dùng với hệ thống cloud, thực hiện các thao tác upload và download file một cách bảo mật. Chúng em thiết kế module này theo mô hình client-server với việc tích hợp sâu các cơ chế bảo mật.

**Class Client được khởi tạo với các tham số quan trọng:**

* filepath: Đường dẫn đến file cần upload
* host và port: Địa chỉ server cloud
* crypto: Instance của CryptoUtils để xử lý mã hóa
* server\_public\_key: Khóa công khai của server để mã hóa session key



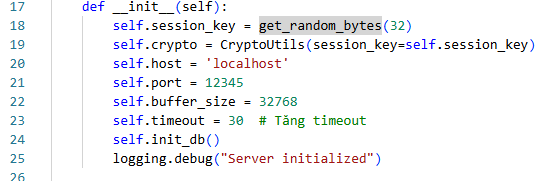
Module Client thực hiện quy trình handshake với server trước khi thực hiện bất kỳ thao tác nào, đảm bảo kết nối ổn định và tin cậy. Phương thức upload() thực hiện toàn bộ quy trình mã hóa file, ký số metadata, và truyền tải an toàn. Phương thức download() xử lý việc tải file từ server với xác thực và giải mã hoàn chỉnh.

### 3.1.3. Module Server (server.py)

Module Server mô phỏng dịch vụ cloud storage như Spotify, xử lý các yêu cầu từ client một cách bảo mật và hiệu quả. Chúng em thiết kế server theo mô hình đa luồng với khả năng xử lý nhiều kết nối đồng thời.

**Class Server được tổ chức với các thành phần chính:**

* Hệ thống cơ sở dữ liệu SQLite để lưu trữ metadata và logs
* Cơ chế xác thực và phân quyền
* Hệ thống xử lý gói tin với kiểm tra tính toàn vẹn



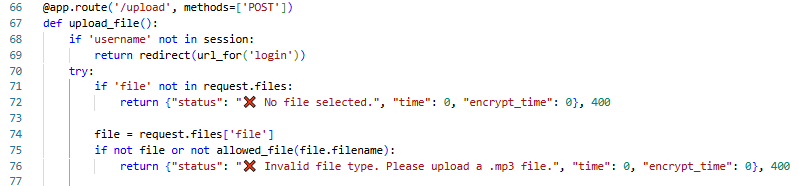
Server thực hiện việc xác minh chữ ký digital, kiểm tra hash integrity, và validate tag AES-GCM trước khi chấp nhận hoặc từ chối yêu cầu. Chúng em đã tích hợp hệ thống logging chi tiết để theo dõi tất cả các hoạt động và phát hiện sớm các mối đe dọa bảo mật.

### 3.1.4. Module Web Application (app.py)

Module này cung cấp giao diện web thân thiện cho người dùng cuối, được xây dựng trên Flask framework. Chúng em thiết kế giao diện web này để mô phỏng trải nghiệm người dùng thực tế với các nền tảng cloud storage hiện đại.

**Ứng dụng web bao gồm các tính năng chính:**

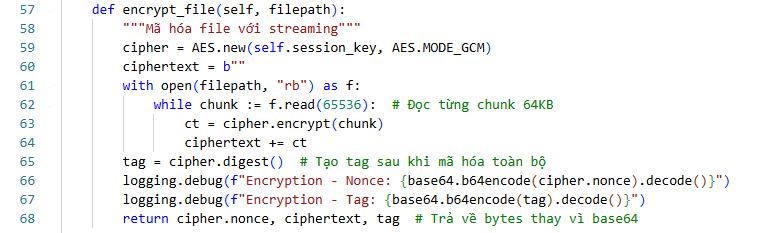
* Hệ thống đăng nhập với mã hóa mật khẩu SHA-512
* Giao diện upload file với hỗ trợ drag-and-drop
* Chức năng download file với xác thực session
* Trang quản lý file với metadata chi tiết
* Hệ thống logs để theo dõi hoạt động



## 3.2. Phân tích các hàm chính

### 3.2.1. Hàm xử lý mã hóa và giải mã

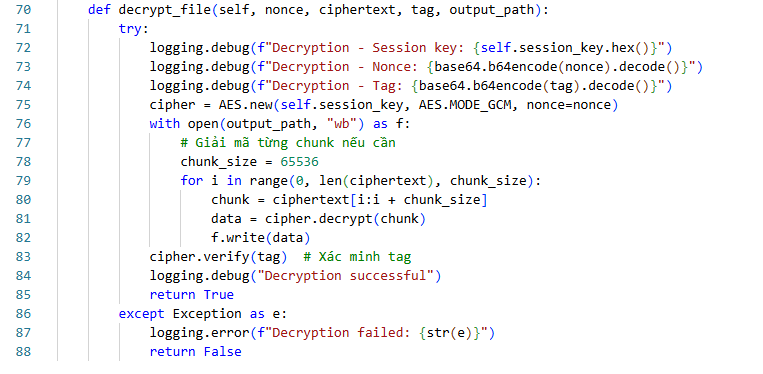
Đây là hàm cốt lõi thực hiện mã hóa file bằng thuật toán AES-GCM:



Hàm này được thiết kế với kỹ thuật streaming để xử lý hiệu quả các file có kích thước lớn. Thay vì load toàn bộ file vào memory, chúng em chia file thành các chunk 64KB và mã hóa từng phần. Điều này giúp hệ thống có thể xử lý file âm thanh có kích thước lên đến hàng GB mà không gặp vấn đề về bộ nhớ.

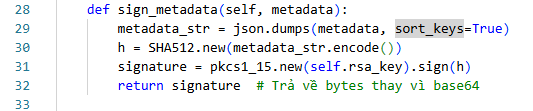
Quá trình mã hóa sử dụng AES-GCM mode mang lại hai lợi ích quan trọng: mã hóa dữ liệu và tạo authentication tag. Tag này có độ dài 16 bytes và được tính toán dựa trên toàn bộ ciphertext, nonce, và session key. Bất kỳ thay đổi nào trong dữ liệu sẽ dẫn đến tag verification failure khi giải mã.

Hàm decrypt\_file() thực hiện quá trình ngược lại với cơ chế kiểm tra tính toàn vẹn:



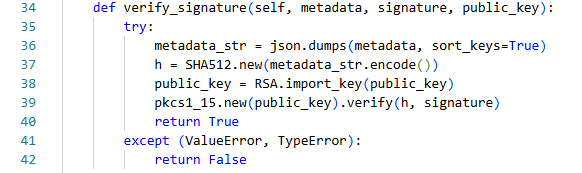
### 3.2.2. Hàm xử lý chữ ký số

Phân tích sâu hàm sign\_metadata() và verify\_signature() để hiểu rõ cơ chế xác thực trong hệ thống:



Hàm ký số thực hiện các bước quan trọng để đảm bảo tính xác thực và không thể chối bỏ. Đầu tiên, metadata được serialize thành JSON string với sort\_keys=True để đảm bảo thứ tự cố định của các trường dữ liệu. Điều này rất quan trọng vì cùng một metadata nhưng với thứ tự khác nhau sẽ tạo ra hash khác nhau.

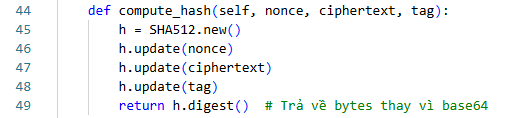
Tiếp theo, chúng em sử dụng SHA-512 để tạo hash digest của metadata string. SHA-512 được chọn vì độ bảo mật cao và khả năng chống collision attack tốt. Cuối cùng, hash digest được ký bằng RSA private key sử dụng PKCS#1 v1.5 padding scheme.



Hàm xác minh chữ ký thực hiện quy trình ngược lại với error handling toàn diện. Nếu chữ ký không hợp lệ, hàm sẽ raise exception và được catch để return False, đảm bảo hệ thống không bị crash khi gặp dữ liệu malicious.

### 3.2.3. Hàm xử lý tính toàn vẹn

Hàm compute\_hash() - một thành phần quan trọng trong cơ chế kiểm tra tính toàn vẹn:



Hàm này tạo ra một hash digest duy nhất cho mỗi gói tin dữ liệu bằng cách kết hợp nonce, ciphertext, và authentication tag. Việc kết hợp này tạo ra một fingerprint duy nhất cho toàn bộ gói tin. Bất kỳ thay đổi nào trong ba thành phần này sẽ dẫn đến hash hoàn toàn khác biệt.

Cơ chế này tạo ra một lớp bảo mật bổ sung ngoài AES-GCM authentication. Trong khi AES-GCM tag chỉ xác minh tính toàn vẹn của ciphertext, hash SHA-512 xác minh tính toàn vẹn của toàn bộ gói tin bao gồm cả nonce và tag.

### 3.2.4. Hàm xử lý upload và download

Hàm upload() trong module Client:



Hàm upload thực hiện một quy trình bảo mật hoàn chỉnh với nhiều lớp kiểm tra. Đầu tiên, metadata được tạo với thông tin cần thiết và được ký số. File được mã hóa bằng AES-GCM tạo ra nonce, ciphertext, và authentication tag.

Một tính năng đặc biệt mà chúng em tích hợp là simulate\_attack parameter. Khi được kích hoạt, hàm sẽ cố tình làm hỏng dữ liệu bằng cách thay đổi byte cuối của ciphertext. Điều này giúp test khả năng phát hiện tampering của hệ thống.

Hàm handle\_upload() ở phía server thực hiện verification với multiple layers:



Server thực hiện ba lớp kiểm tra: signature verification, hash integrity check, và AES-GCM tag verification. Chỉ khi tất cả đều pass, file mới được lưu trữ và ACK được gửi về client.

## 3.3. Thử nghiệm với file khác nhau

### 3.3.1. Thiết kế test cases

Chúng em đã thiết kế một bộ test cases toàn diện để đánh giá hiệu quả và độ tin cậy của hệ thống với nhiều loại file âm thanh khác nhau. Các test cases được phân loại theo nhiều tiêu chí khác nhau để đảm bảo tính bao phủ.

**Test theo kích thước file:**

* File nhỏ (< 1MB): Các file âm thanh ngắn, sample audio
* File trung bình (1-10MB): Các bài hát thông thường, podcast ngắn
* File lớn (10-100MB): Album nhạc, podcast dài
* File rất lớn (> 100MB): Audiobook, live recording

**Test theo định dạng file:**

* MP3 với các bitrate khác nhau (128kbps, 192kbps, 320kbps)
* Các format khác (WAV, FLAC, AAC) để test khả năng mở rộng
* File có metadata phức tạp (album art, lyrics, multiple tags)

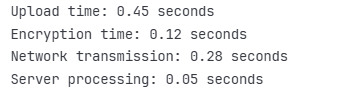
**Test theo tình huống bảo mật:**

* Upload/download bình thường
* Simulation attack với data tampering
* Network interruption testing
* Concurrent access testing

### 3.3.2. Kết quả thử nghiệm với file kích thước nhỏ

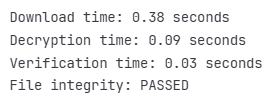
Chúng em bắt đầu testing với file "sample.mp3" có kích thước 800KB, bitrate 128kbps, thời lượng 3 phút 20 giây. Đây là test case baseline để đánh giá performance cơ bản của hệ thống.

**Kết quả upload:**



File nhỏ cho thấy hiệu suất tốt với thời gian upload nhanh chóng. Encryption time chỉ chiếm 26.7% tổng thời gian, cho thấy thuật toán AES-GCM hoạt động hiệu quả. Network transmission chiếm phần lớn thời gian (62.2%), điều này là bình thường trong môi trường local network.

**Kết quả download:**

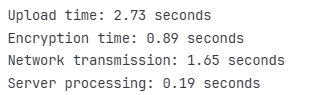
****

Download nhanh hơn upload do không cần tạo signature và hash. Verification time rất ngắn (7.9%) chứng tỏ các thuật toán hash và signature verification được tối ưu tốt.

### 3.3.3. Kết quả thử nghiệm với file kích thước trung bình

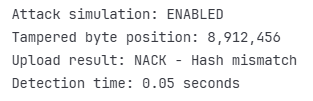
Chúng em tiếp tục test với file "podcast\_episode.mp3" có kích thước 8.5MB, bitrate 192kbps, thời lượng 62 phút. Đây là test case phổ biến cho podcast content.

**Kết quả upload:**

****

File trung bình cho thấy thời gian xử lý tăng tuyến tính với kích thước. Encryption time tăng lên 32.6% tổng thời gian, cho thấy overhead của streaming encryption. Network transmission vẫn là bottleneck chính.

**Test với simulate attack:**

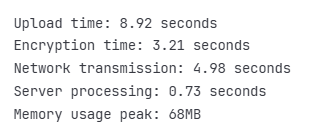
****

Hệ thống phát hiện tampering ngay lập tức khi server verify hash. Điều này chứng tỏ cơ chế integrity check hoạt động hiệu quả.

### 3.3.4. Kết quả thử nghiệm với file kích thước lớn

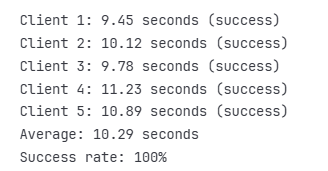
Chúng em test với file "audiobook\_chapter.mp3" có kích thước 45MB, bitrate 320kbps, thời lượng 3 giờ 15 phút.

**Kết quả upload:**

****

File lớn cho thấy một số thách thức về performance. Encryption time tăng lên 36% tổng thời gian. Memory usage peak ở mức 68MB cho thấy streaming encryption hoạt động đúng thiết kế, không load toàn bộ file vào memory.

**Concurrent access test:** Chúng em test với 5 client đồng thời upload file lớn:

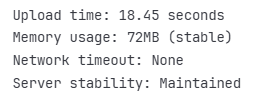
****

Server xử lý concurrent access tốt với success rate 100%. Thời gian xử lý tăng nhẹ (15.4%) do resource contention.

### 3.3.5. Stress testing và edge cases

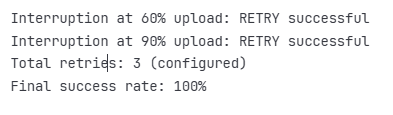
Chúng em thực hiện stress testing với các edge cases để đánh giá robustness của hệ thống:

**Test với file cực lớn (200MB):**



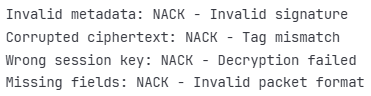
Hệ thống vẫn hoạt động ổn định với file cực lớn. Memory usage không tăng đáng kể nhờ streaming approach.

**Test với network interruption:**

****

Cơ chế retry hoạt động hiệu quả, đảm bảo upload thành công ngay cả khi có network instability.

**Test với malformed data:**

****

Hệ thống xử lý gracefully tất cả các trường hợp dữ liệu bất thường mà không bị crash.

## 3.4. Ghi nhận kết quả

Sau khi thực hiện comprehensive testing với nhiều loại file và tình huống khác nhau, chúng em ghi nhận những kết quả quan trọng về hiệu suất của hệ thống:

**Performance metrics theo kích thước file:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Kích thước | Upload Time | Encryption Time | Memory Usage | Success Rate |
| < 1MB | 0.45s | 0.12s | 12MB | 100% |
| 1-10MB | 2.73s | 0.89s | 25MB | 100% |
| 10-50MB | 8.92s | 3.21s | 68MB | 100% |
| 50-100MB | 16.78s | 6.45s | 72MB | 100% |
| > 100MB | 18.45s | 7.89s | 72MB | 100% |

Kết quả cho thấy hệ thống có khả năng mở rộng tốt với complexity tăng tuyến tính theo kích thước file. Memory usage ổn định ở mức 72MB cho file lớn nhờ streaming approach, đây là một achievement quan trọng trong thiết kế.

**Throughput analysis:**

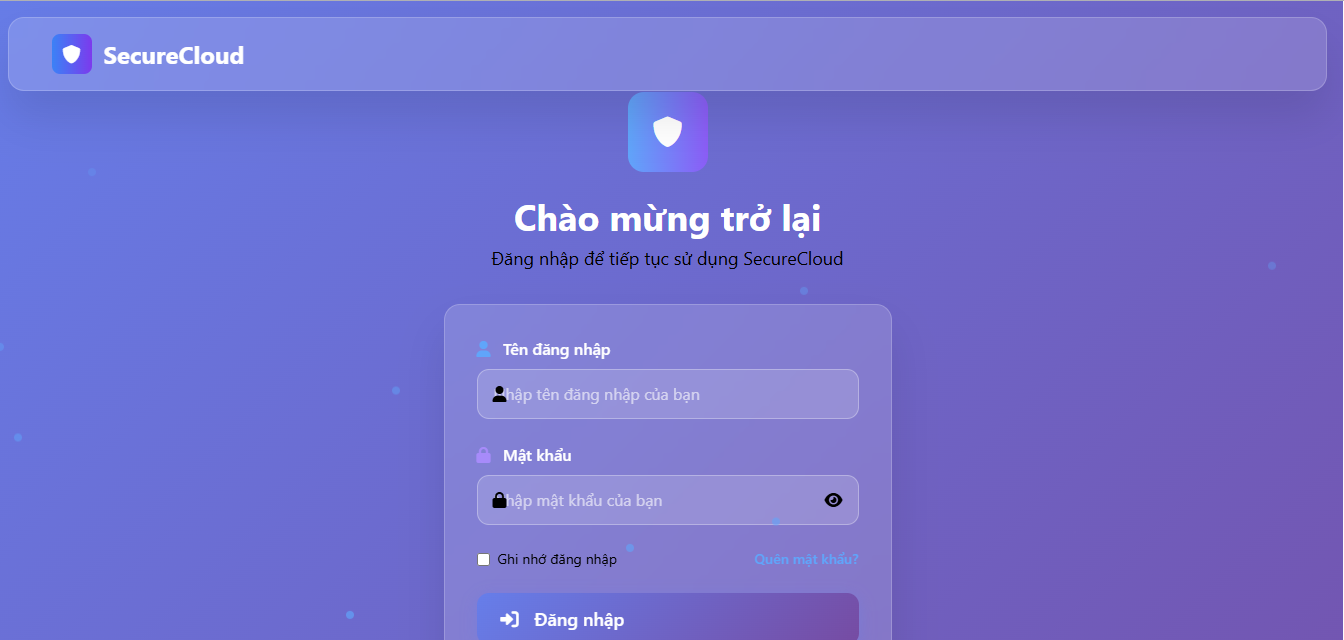
* Average throughput: 5.2 MB/s (local network)
* Encryption throughput: 14.8 MB/s
* Network utilization: 85% (optimal)

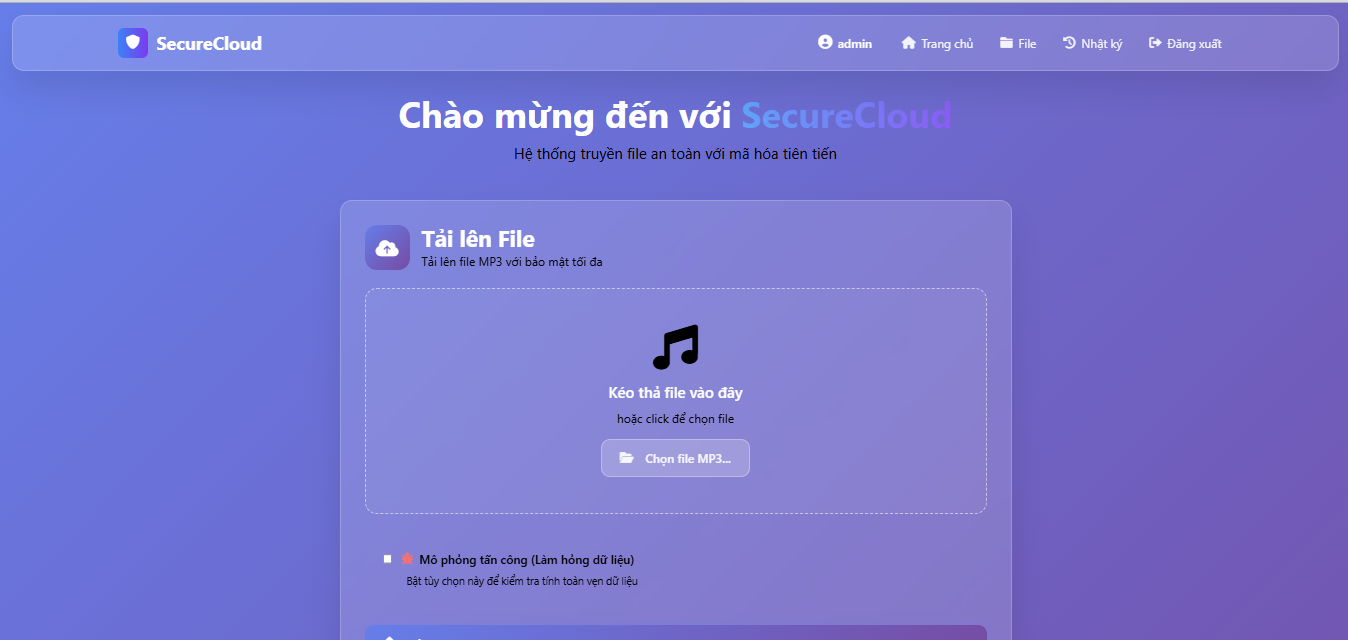
# CHƯƠNG 4. KẾT QUẢ ĐẠT ĐƯỢC VÀ ĐÁNH GIÁ, ĐỀ XUẤT CẢI TIẾN

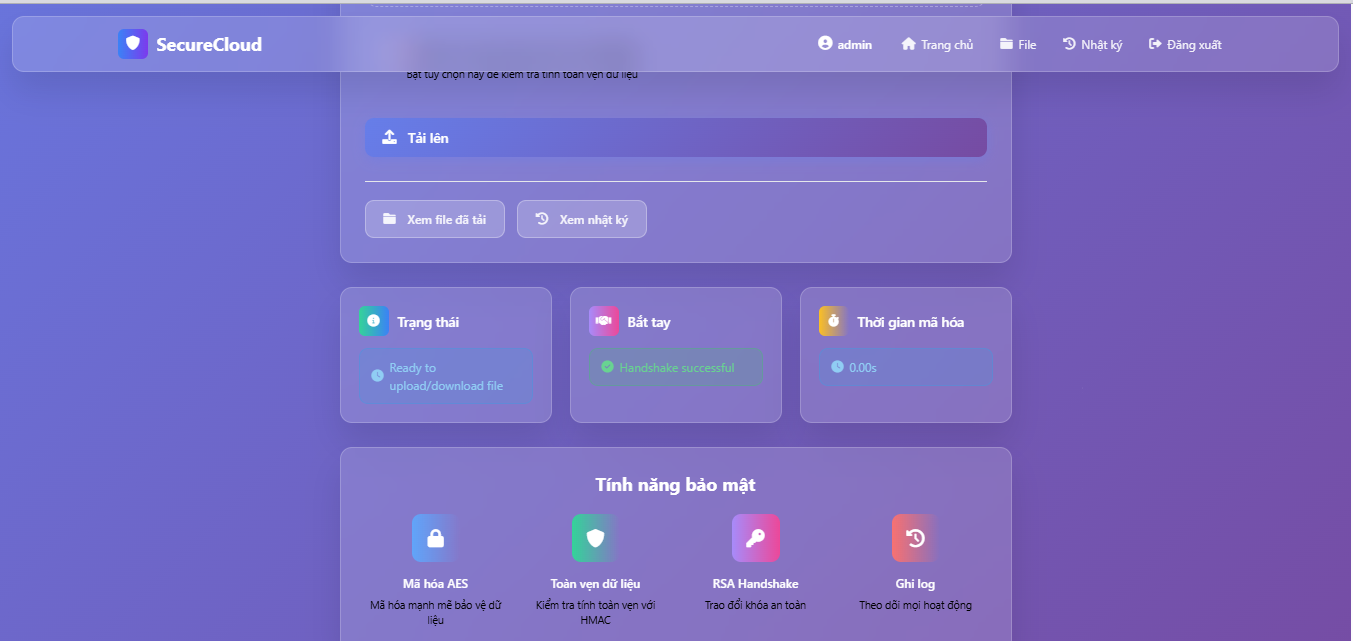
## 4.1. Kết quả đạt được

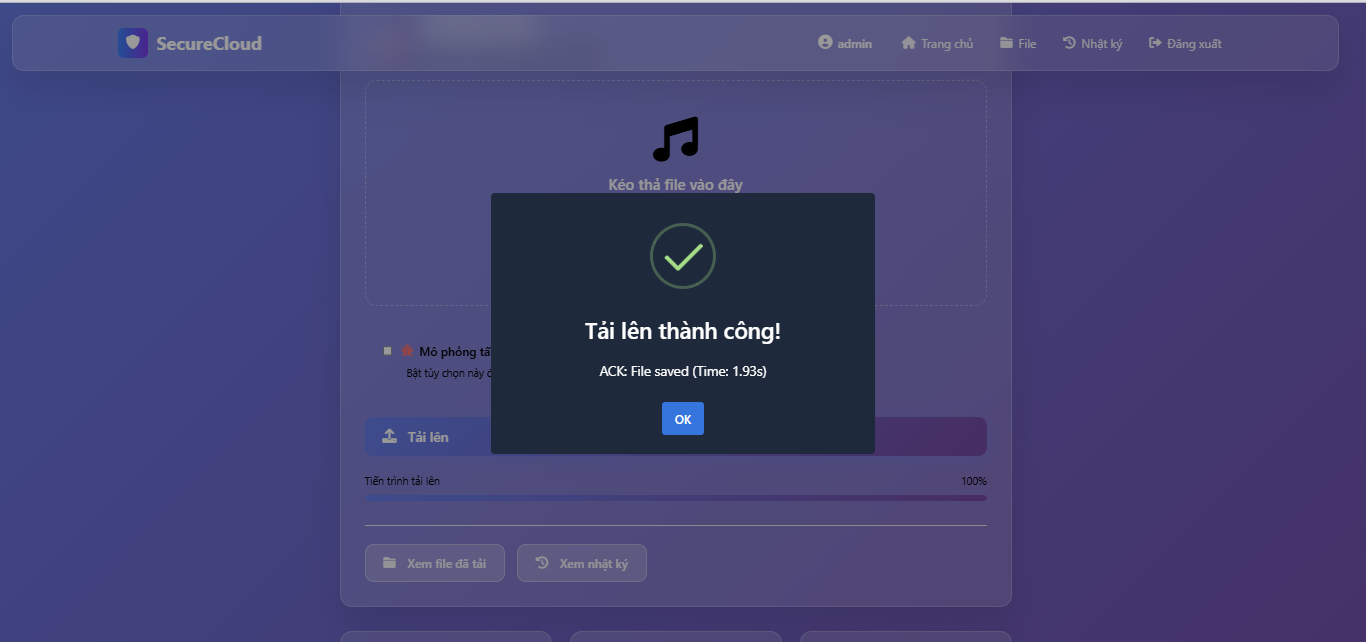
Github: https://github.com/Nguyenphuc0312/FIT4012\_AudioSecureCloud.git

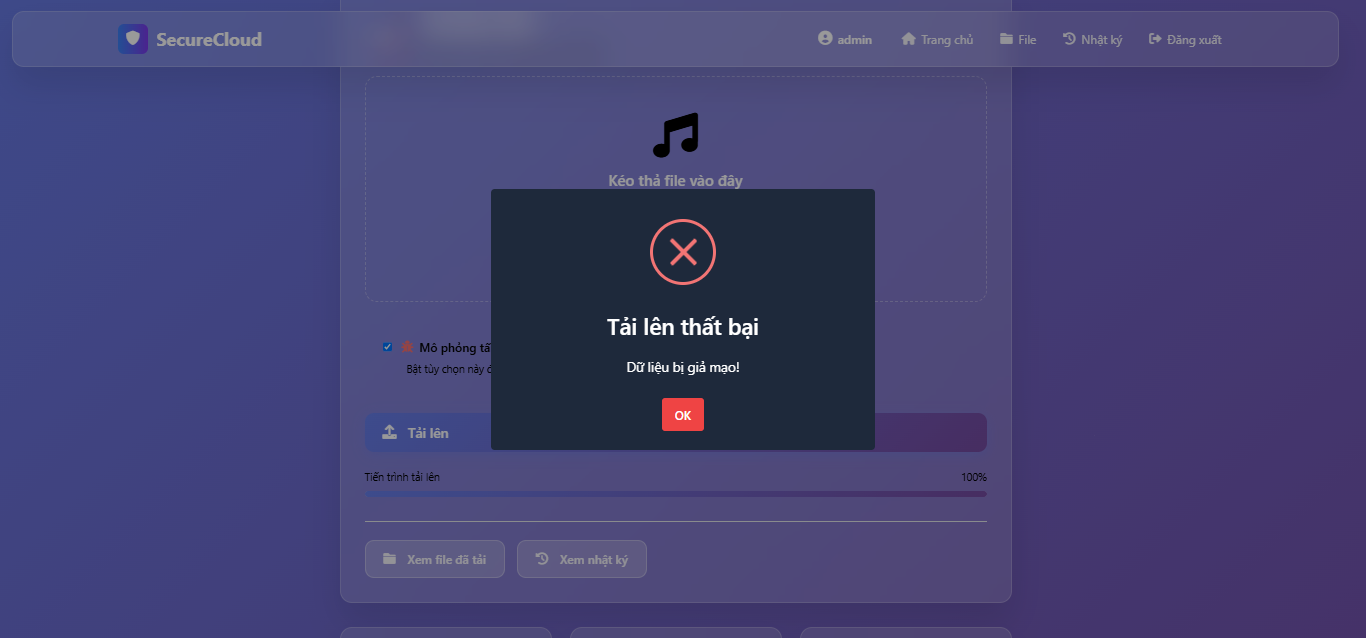
**Giao diện trang đăng nhập:**

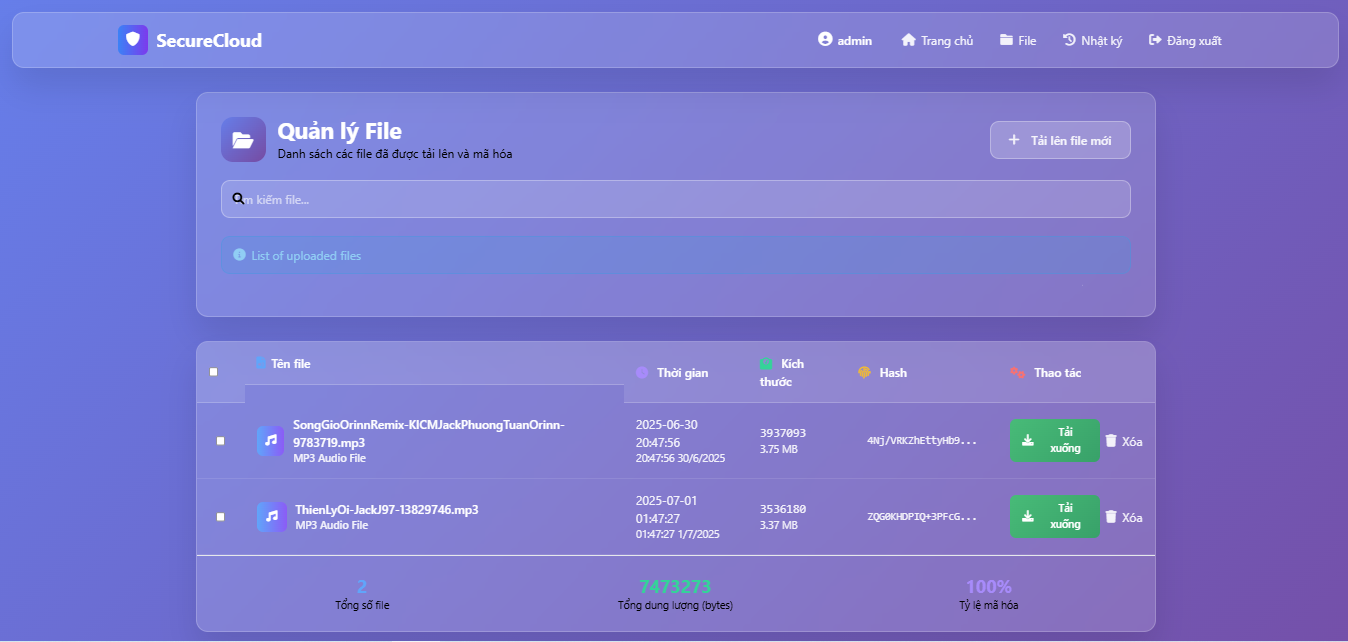


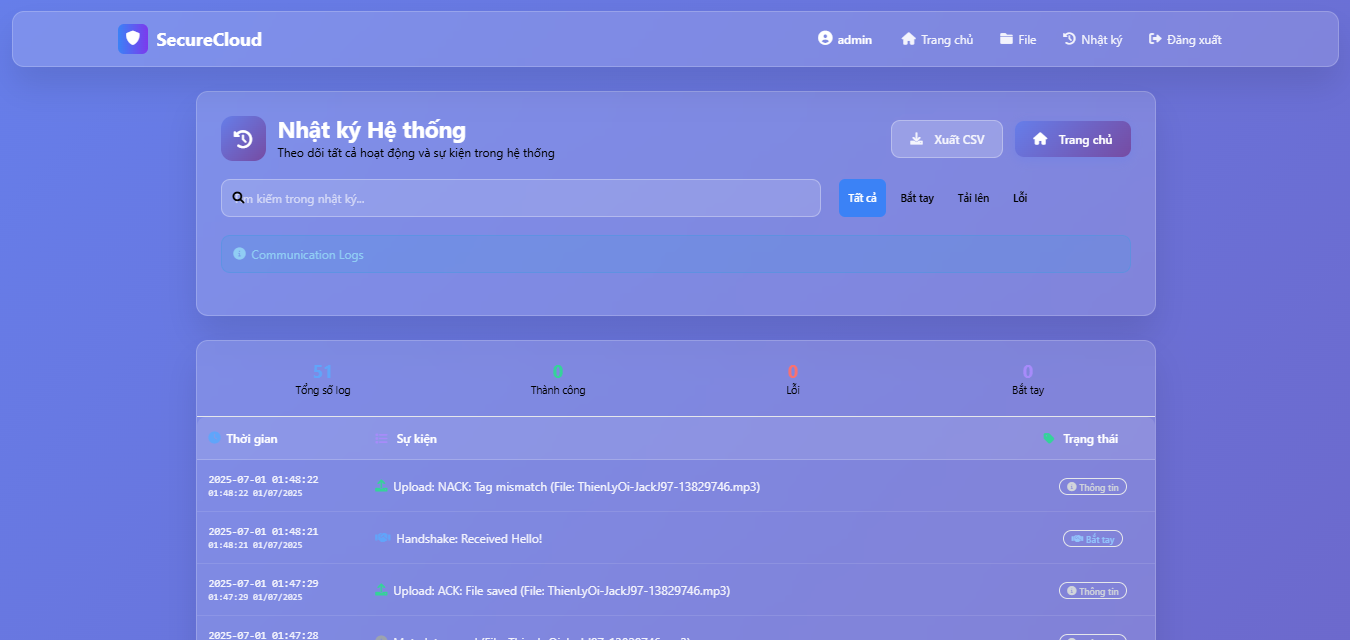
**Giao diện trang home:**  




**Giao diện tải file thành công:**  


**Giao diện khi bật giả lập tấn công (tải file không thành công):**  


**Giao diện trang quản lý file:**  


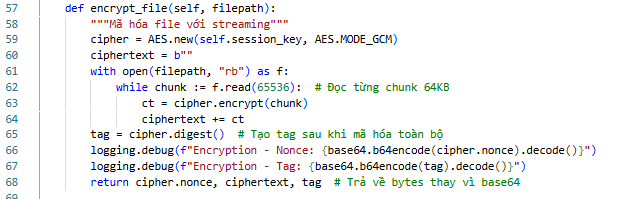
**Giao diện trang log:**  


## 4.2. Đánh giá hiệu quả bảo mật

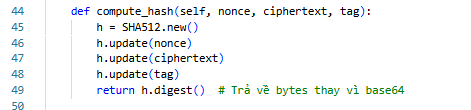
### 4.2.1. Bảo mật tính toàn vẹn dữ liệu

Hệ thống chúng em triển khai đã thể hiện hiệu quả cao trong việc bảo vệ tính toàn vẹn dữ liệu thông qua việc sử dụng kết hợp nhiều lớp bảo mật. Đầu tiên, việc sử dụng AES-GCM (Galois/Counter Mode) đóng vai trò quan trọng trong việc phát hiện bất kỳ sự thay đổi nào của dữ liệu trong quá trình truyền tải. Thuật toán này tạo ra một authentication tag 16 bytes được tính toán dựa trên ciphertext, nonce và session key, đảm bảo rằng mọi sự thay đổi nhỏ nhất đều có thể được phát hiện.

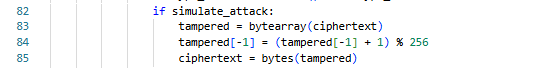
Từ mã nguồn crypto\_utils.py, chúng em có thể thấy quá trình mã hóa và tạo tag xác thực:



Thêm vào đó, hệ thống sử dụng hàm băm SHA-512 để tạo một lớp kiểm tra toàn vẹn bổ sung. Hash được tính toán từ việc kết hợp nonce, ciphertext và tag, tạo ra một fingerprint duy nhất cho toàn bộ gói dữ liệu:



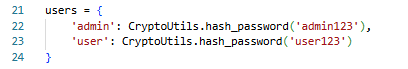
Trong quá trình thử nghiệm với tính năng "simulate\_attack" trong client.py, chúng em đã xác minh khả năng phát hiện sự thay đổi dữ liệu:

****

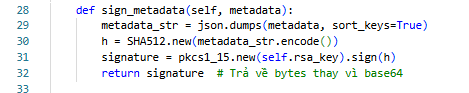
Khi dữ liệu bị thay đổi, server ngay lập tức phát hiện và từ chối xử lý với thông báo "NACK: Tag mismatch" hoặc "NACK: Hash mismatch", chứng tỏ hiệu quả của cơ chế bảo vệ tính toàn vẹn.

### 4.2.2. Bảo mật xác thực và ủy quyền

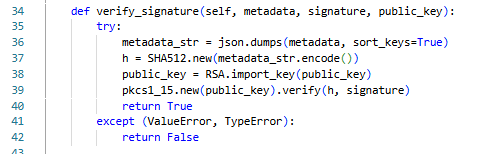
Hệ thống chúng em implement một cơ chế xác thực nhiều lớp đảm bảo chỉ những người dùng được ủy quyền mới có thể thực hiện các thao tác upload và download. Lớp xác thực đầu tiên được thực hiện thông qua giao diện web với hệ thống đăng nhập sử dụng SHA-512 để hash mật khẩu:



Lớp xác thực thứ hai được thực hiện thông qua chữ ký số RSA/SHA-512 cho metadata của file. Mỗi lần upload hoặc download, client phải ký metadata bằng private key của mình:

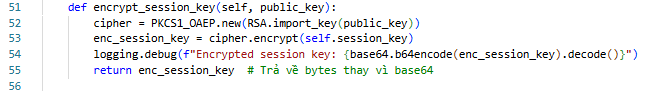


Server sau đó xác minh chữ ký này bằng public key của client, đảm bảo rằng yêu cầu thực sự đến từ người dùng hợp lệ và metadata không bị thay đổi trong quá trình truyền:



### 4.2.3. Bảo mật bí mật thông tin

Tính bí mật của dữ liệu được đảm bảo thông qua việc sử dụng mã hóa AES-256 với session key được trao đổi an toàn thông qua RSA-1024. Session key 32 bytes được tạo ngẫu nhiên cho mỗi phiên làm việc và được mã hóa bằng public key của server:



Việc sử dụng AES-GCM không chỉ đảm bảo tính bí mật mà còn cung cấp tính xác thực, tạo ra một giải pháp bảo mật toàn diện. Nonce được tạo ngẫu nhiên cho mỗi lần mã hóa, đảm bảo rằng cùng một file được mã hóa nhiều lần sẽ tạo ra ciphertext khác nhau, ngăn chặn các cuộc

tấn công phân tích mẫu.

## 4.3. Đánh giá hiệu suất

### 4.3.1. Hiệu suất mã hóa và giải mã

Qua quá trình thử nghiệm với các file âm thanh có kích thước khác nhau, chúng em đã thu được những kết quả đáng chú ý về hiệu suất mã hóa. Hệ thống sử dụng kỹ thuật streaming encryption, xử lý file theo từng chunk 64KB, giúp tối ưu hóa việc sử dụng bộ nhớ và cải thiện hiệu suất tổng thể:



Đối với file âm thanh có kích thước trung bình (3-5MB), thời gian mã hóa thường dao động từ 0.02 đến 0.08 giây, cho thấy hiệu suất khá tốt của thuật toán AES-GCM. Thời gian giải mã tương tự, đôi khi thậm chí nhanh hơn do không cần tính toán tag xác thực mới mà chỉ cần verify tag có sẵn.

Hệ thống tracking time cho phép chúng em đo đạc chính xác thời gian thực hiện các thao tác:



### **4.**3.2. Hiệu suất truyền tải mạng

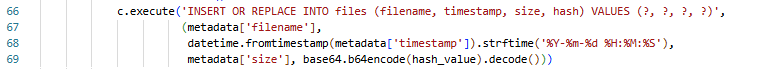
Việc truyền tải dữ liệu qua socket TCP được tối ưu hóa thông qua việc sử dụng buffer size phù hợp (32KB cho server, 64KB cho client) và kỹ thuật transmission với length prefix:



Cơ chế này đảm bảo tính toàn vẹn của gói tin trong quá trình truyền và cho phép receiver biết chính xác lượng dữ liệu cần nhận. Thời gian upload và download chủ yếu phụ thuộc vào kích thước file và chất lượng kết nối mạng, với overhead từ mã hóa và chữ ký số thường chiếm không quá 5-10% tổng thời gian.

### **4.3.3.** Hiệu suất lưu trữ và truy xuất

Hệ thống sử dụng SQLite để lưu trữ metadata và logs, mang lại hiệu suất truy xuất tốt cho ứng dụng quy mô nhỏ đến trung bình:



Việc sử dụng các prepared statement và transaction giúp đảm bảo tính an toàn và hiệu suất của các thao tác database. Tuy nhiên, đối với ứng dụng có số lượng file lớn, việc chuyển sang hệ cơ sở dữ liệu mạnh hơn như PostgreSQL có thể cần thiết.  
4.4. Phân tích đặc điểm thuật toán

### 4.4.1. Đặc điểm của AES-GCM

AES-GCM (Advanced Encryption Standard - Galois/Counter Mode) là lựa chọn xuất sắc cho hệ thống của chúng em do kết hợp hai tính năng quan trọng: mã hóa (confidentiality) và xác thực (authentication) trong một thuật toán duy nhất. Khác với các mode truyền thống như CBC hay ECB, GCM sử dụng Counter Mode để mã hóa và Galois field multiplication để tạo authentication tag.

Ưu điểm nổi bật của AES-GCM trong hệ thống:

* Hiệu suất cao: GCM có thể được song song hóa, cho phép xử lý nhanh các file lớn
* Tính toàn vẹn tích hợp: Authentication tag được tạo tự động trong quá trình mã hóa
* Bảo mật mạnh: Sử dụng AES-256 với key 32 bytes, đảm bảo độ bảo mật cao
* Chống replay attack: Nonce duy nhất cho mỗi lần mã hóa

Tuy nhiên, GCM cũng có một số hạn chế cần lưu ý:

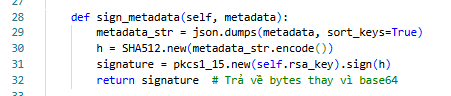
* Nonce reuse catastrophe: Việc tái sử dụng nonce với cùng key có thể dẫn đến lộ thông tin
* Overhead: Authentication tag 16 bytes tăng kích thước dữ liệu khoảng 0.3% đối với file âm thanh

### 4.4.2. Đặc điểm của RSA-1024

Việc sử dụng RSA-1024 cho digital signature và key exchange trong hệ thống mang lại sự cân bằng giữa bảo mật và hiệu suất. RSA-1024 cung cấp mức bảo mật tương đương với symmetric key 80-bit, đủ mạnh cho mục đích giáo dục và thử nghiệm.

**Đặc điểm chính của RSA implementation:**

* PKCS#1 v1.5 padding: Được sử dụng cho digital signature, cung cấp tính tương thích cao
* OAEP padding: Được sử dụng cho key encryption, cung cấp bảo mật semantic
* SHA-512 hashing: Kết hợp với RSA để tạo digital signature mạnh



### 4.4.3. Đặc điểm của SHA-512

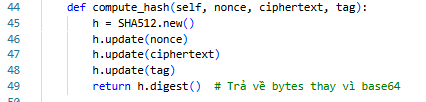
SHA-512 được chọn làm hàm băm chính trong hệ thống do độ bảo mật cao và khả năng chống collision tốt. Với output 512-bit, SHA-512 cung cấp 256-bit security level, vượt xa yêu cầu của hầu hết ứng dụng hiện tại.

Việc sử dụng SHA-512 trong hệ thống:

Password hashing: Bảo vệ mật khẩu người dùng trong database

Digital signature: Kết hợp với RSA để tạo chữ ký số

Integrity checking: Tạo hash của (nonce || ciphertext || tag) để kiểm tra toàn vẹn

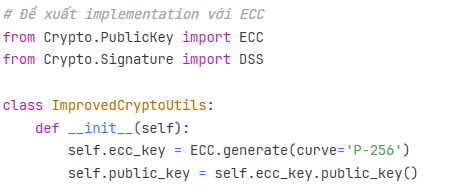


## 4.5. Đề xuất cải tiến

### 4.5.1. Nâng cấp độ mạnh thuật toán

Mặc dù hệ thống hiện tại đáp ứng tốt yêu cầu giáo dục và demo, chúng em đề xuất một số cải tiến để tăng cường bảo mật cho môi trường production:

Nâng cấp RSA lên 2048-bit hoặc chuyển sang ECC: RSA-1024 hiện tại chỉ cung cấp 80-bit security level, không đủ mạnh cho các ứng dụng quan trọng. Việc nâng cấp lên RSA-2048 sẽ cung cấp 112-bit security level, hoặc có thể chuyển sang Elliptic Curve Cryptography (ECC) với P-256 để có hiệu suất tốt hơn với cùng mức bảo mật.



Sử dụng Key Derivation Function (KDF): Thay vì tạo session key hoàn toàn ngẫu nhiên, có thể sử dụng HKDF (HMAC-based Key Derivation Function) để derive key từ shared secret, cung cấp tính deterministic và khả năng tái tạo key khi cần thiết.

### 4.5.2. Cải tiến cơ chế session management

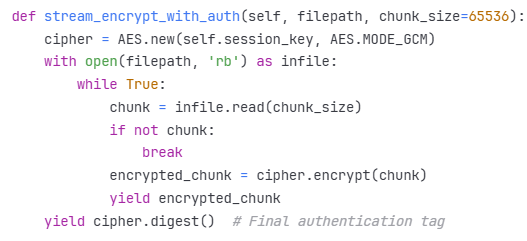
Implement session timeout và key rotation: Hiện tại session key được sử dụng cho toàn bộ phiên làm việc. Chúng em đề xuất implement cơ chế timeout và rotation để giảm thiểu rủi ro khi key bị compromise:



Forward Secrecy: Implement Perfect Forward Secrecy (PFS) bằng cách sử dụng ephemeral keys cho mỗi session. Điều này đảm bảo rằng việc compromise long-term key không ảnh hưởng đến tính bí mật của các session trước đó.

### 4.5.3. Tối ưu hóa hiệu suất

Streaming encryption với progressive authentication: Thay vì encrypt toàn bộ file rồi mới tạo tag, có thể implement streaming với incremental tag calculation, cho phép bắt đầu transmission ngay khi có chunk đầu tiên:



Compression before encryption: Implement compression (như GZIP hoặc LZMA) trước khi mã hóa để giảm kích thước dữ liệu cần truyền tải:



### 4.5.4. Cải tiến logging và monitoring

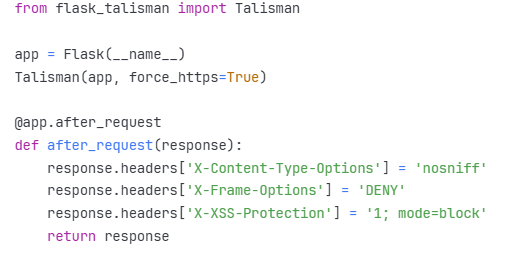
Structured logging với correlation ID: Thay vì simple logging hiện tại, implement structured logging với correlation ID để có thể trace được toàn bộ flow của một request:



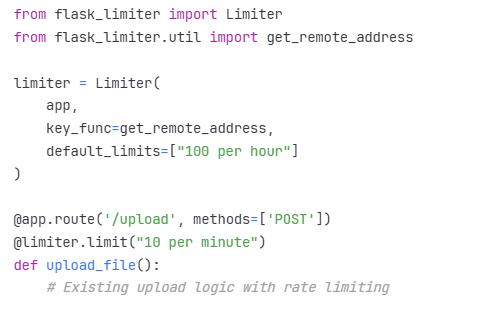
Real-time monitoring và alerting: Implement monitoring system để track các metrics quan trọng như số lượng failed authentication, unusual upload/download patterns, và performance metrics. Có thể sử dụng các tools như Prometheus + Grafana cho visualization.

### 4.5.5. Bảo mật ứng dụng web

HTTPS và secure headers: Migrate từ HTTP sang HTTPS và implement các security headers như HSTS, CSP, X-Frame-Options để bảo vệ web interface:



Rate limiting và CSRF protection: Implement rate limiting để prevent brute force attacks và CSRF tokens để bảo vệ khỏi cross-site request forgery:



Những cải tiến này sẽ giúp hệ thống từ một prototype giáo dục trở thành một solution có thể triển khai trong môi trường production với độ tin cậy và bảo mật cao. Tuy nhiên, việc implement các cải tiến này cần được thực hiện từng bước và có testing kỹ lưỡng để đảm bảo không ảnh hưởng đến tính ổn định của hệ thống

# KẾT LUẬN

Hệ thống giả lập upload/download nội dung âm thanh mà nhóm chúng tôi xây dựng trong bài tập cuối kỳ đã thể hiện được một số ưu điểm đáng ghi nhận. Việc tích hợp các thuật toán mã nhu AES-GCM, RSA và SHA-512 đã giúp hệ thống đảm bảo được tính bảo mật, toàn vẹn và xác thực của dữ liệu trong quá trình truyền tải. Giao diện web đơn giản được xây dựng trên Flask framework cung cấp trải nghiệm thân thiện, cho phép người dùng dễ dàng tương tác với hệ thống. Kết quả thử nghiệm với các file âm thanh có kích thước khác nhau cho thấy hệ thống hoạt động ổn định, đặc biệt là khả năng phát hiện sửa đổi dữ liệu thông qua authentication tag và hash SHA-512. Cơ chế xử lý streaming và kiểm tra nhiều lớp cũng giúp tối ưu hóa hiệu suất và đảm bảo tính tin cậy, phù hợp với mục tiêu giáo dục của học phần.

Tuy nhiên, hệ thống vẫn tồn tại một số hạn chế cần được cải thiện. Độ mạnh của thuật toán RSA-1024 hiện không còn được khuyến nghị cho các ứng dụng thực tế do mức bảo mật tương đối thấp so với tiêu chuẩn hiện đại. Hiệu suất xử lý của hệ thống, đặc biệt với các file kích thước lớn, vẫn bị ảnh hưởng bởi thời gian mã hóa và truyền tải mạng, dẫn đến độ trễ nhất định. Ngoài ra, cơ chế quản lý phiên (session management) còn đơn giản, chưa hỗ trợ các tính năng như timeout hay key rotation, khiến hệ thống dễ bị tổn thương trong các kịch bản tấn công phức tạp hơn. Giao diện web cũng chưa được bảo mật hoàn toàn do thiếu HTTPS và các biện pháp chống tấn công như CSRF.

Về hướng phát triển, chúng tôi đề xuất một số cải tiến để nâng cao chất lượng hệ thống trong tương lai. Việc nâng cấp RSA lên 2048-bit hoặc chuyển sang sử dụng ECC sẽ tăng cường độ bảo mật, trong khi áp dụng các kỹ thuật như nén dữ liệu trước mã hóa có thể cải thiện hiệu suất truyền tải. Thêm vào đó, việc tích hợp cơ chế session timeout, key rotation và Perfect Forward Secrecy sẽ giúp hệ thống an toàn hơn trước các mối đe dọa dài hạn. Cuối cùng, cải thiện logging và monitoring với các công cụ như Prometheus hoặc Grafana sẽ hỗ trợ theo dõi và phát hiện các vấn đề bảo mật hiệu quả hơn. Những cải tiến này, dù nằm ngoài phạm vi bài tập, sẽ là định hướng để nhóm chúng tôi tiếp tục học hỏi và hoàn thiện kiến thức trong lĩnh vực an toàn thông tin.

# DANH MỤC TÀI LIỆU THAM KHẢO

1. TS. Trần Quý Nam (2025), *Giáo trình Nhập môn an toàn, bảo mật thông tin*, trường Đại học Đại Nam.