**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN, ĐHQG-HCM**

**KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG**



**BÁO CÁO ĐỒ ÁN MÔN HỌC**

ĐỀ TÀI: MULTIMEDIA PRODUCT

SERVICE PLATFORM

**Môn học:** NT219.P22.ANTT – Mật Mã Học

**Giảng viên hướng dẫn:** PhD. Ngoc-Tu Nguyen

**Thực hiện bởi nhóm 29, bao gồm:**

1. Lê Minh 23520928 Trưởng nhóm
2. Lương Hoàng Long 23520928 Thành viên

**Thời gian thực hiện:** 30/04/2025 - 23/06/2025

# LỜI NÓI ĐẦU

Kính gửi Thầy Nguyễn Ngọc Tự,

Chúng em xin chân thành gửi đến Thầy lời cảm ơn chân thành nhất vì đã hỗ trợ bọn em rất nhiều suốt thời gian qua. Nhờ có sự giúp đỡ của Thầy mà bọn em đã có thể nắm vững lý thuyết và vận dụng được những kiến thức bổ ích trong môn Mật Mã Học vào trong quá trình thực hiện đồ án này. Không chỉ là lý thuyết khô khan, qua những buổi học và quá trình trao đổi cùng Thầy, bọn em cũng đã hiểu rõ hơn được ý nghĩa của công việc mình đang theo đuổi và yêu thích môn học Mật Mã Học nói riêng và ngành An Toàn Thông Tin nói chung. Hơn nữa, phương pháp truyền đạt của Thầy cũng đã giúp bọn em biết thêm được mình cần phải học nhiều hơn, tìm hiểu nhiều hơn nữa, không chỉ qua trường lớp. Và trên hết, bọn em biết được đạo đức là quan trọng nhất trong công việc này.

Trong quá trình học tập và trao đổi cùng Thầy, cũng như trong quá trình thực hiện đồ án này, bọn em cũng đã mắc rất nhiều sai sót và gặp không ít khó khăn. Một lần nữa, bọn em xin bày tỏ lòng biết ơn đến Thầy vì đã luôn kiên trì hướng dẫn, cho bọn em những lời động viên và lời khuyên quý giá.

Chúng em xin hứa sẽ tiếp tục áp dụng những kiến thức đã được học từ Thầy vào học tập, công việc và cuộc sống sau này. Sự tận tụy và hỗ trợ của Thầy sẽ là nguồn động lực và định hướng cho chúng em trong tương tai, trên con đường theo đuổi nghề bảo vệ tài sản số này.

Bọn em chúc Thầy có nhiều sức khỏe và sẽ luôn giữ vững nhiệt huyết đối với những thế hệ sinh viên tiếp theo cũng như trong công việc hàng ngày.

Trân trọng cảm ơn Thầy,

Lê Minh

Lương Hoàng Long

# MỤC LỤC

[**LỜI NÓI ĐẦU 2**](#_is9s1log1klt)

[**MỤC LỤC 3**](#_4efpsxnyuzk8)

[**MỞ ĐẦU 4**](#_ds1w2r8qm1kv)

[**Chương I. TỔNG QUAN 5**](#_jjoek36erpqf)

[1. Ngữ cảnh 5](#_1tvotu5criqa)

[2. Tài sản, các bên liên quan và vai trò 5](#_b53uduscsi3l)

[2.1. Các bên liên quan và vai trò 5](#_rb0rmoen74rz)

[2.2. Tài sản cần bảo vệ 6](#_bc8yz4ptx2io)

[3. Các rủi ro bảo mật và tính năng cần xây dựng 7](#_2t5kdi6b28hv)

[**Chương II. GIẢI PHÁP 9**](#_7oub5ejs97u7)

[1. Kiến trúc giải pháp 9](#_zhplno3gmh4s)

[1.1. Thuật toán và giao thức mã hóa 9](#_k2i5zweekql2)

[1.2. Thành phần hệ thống 10](#_ejqdoaxr5vpg)

[2. Môi trường triển khai 12](#_ceu4rdrul04n)

[3. Phân tích chi tiết 13](#_kpt7fsay9rzs)

[2.1. Nhà cung cấp đăng tải nhạc 13](#_3k9hdxn4yjm7)

[2.2. Lưu trữ key AES-GCM 14](#_rtk3aavmxmvs)

[2.3. Cơ sở dữ liệu 15](#_6i2uj91q9mzt)

[2.4. Lấy nhạc và giải mã 18](#_ps788fv6ju6l)

[2.5. Gắn watermark 18](#)

[2.6. Xử lý file âm thanh trước khi phát: 19](#)

[**Chương III. TRIỂN KHAI 24**](#_ika4dfs9txcc)

[1. Nhà cung cấp đăng tải nhạc 24](#_hqbdpmex98yk)

[2. Người dùng yêu cầu phát nhạc 25](#_57egwaypv55l)

[**THAM KHẢO 26**](#_pjcdh5vg0pep)

# MỞ ĐẦU

Việc bảo mật trong các nền tảng nghe nhạc trực tuyến đang trở thành một vấn đề ngày càng được quan tâm trong kỷ nguyên số hiện nay. Việc chuyển đổi từ các thiết bị thu phát truyền thống sang các nền tảng nghe nhạc trực tuyến như Spotify, ZingMP3, ... mang lại nhiều lợi ích như sự tiện dụng, khả năng truy cập nhanh chóng và phân phối nội dung dễ dàng. Tuy nhiên, cùng với sự phát triển đó là các nguy cơ tiềm ẩn về vi phạm bản quyền, đánh cắp dữ liệu và rò rỉ thông tin người dùng. Vì vậy, bọn em lựa chọn đề tài này nhằm tìm hiểu và ứng dụng các kiến thức đã học trong môn Mật Mã Học để xây dựng một hệ thống nghe nhạc có khả năng bảo vệ dữ liệu, xác thực truy cập và đảm bảo bản quyền cho nhà cung cấp nội dung.

Mục đích của nghiên cứu là tìm hiểu và vận dụng các phương pháp bảo mật hiện đại – bao gồm mã hóa, xác thực và kiểm soát truy cập – vào một nền tảng nghe nhạc trực tuyến. Hệ thống hướng đến việc bảo vệ quyền lợi của cả hai bên: người dùng và nhà cung cấp nhạc, thông qua việc bảo đảm dữ liệu âm thanh không bị chỉnh sửa, sao chép trái phép hay truy cập ngoài ý muốn.

Đối tượng nghiên cứu của đề tài là các file âm thanh được cung cấp bởi các nhà cung cấp như nghệ sĩ, công ty hay tổ chức. Đặc biệt, nghiên cứu tập trung vào việc bảo mật đường truyền, xác thực nhà cung cấp và người nghe, cũng như đảm bảo toàn vẹn dữ liệu.

Phạm vi nghiên cứu bao gồm việc phân tích và áp dụng các thuật toán mật mã như AES, ECDHE, ... vào hệ thống; xây dựng mô hình xác thực và trao đổi khóa an toàn; thực hiện mã hóa luồng (stream cipher) để phát nhạc theo thời gian thực. Nghiên cứu không tập trung vào giao diện người dùng hay khía cạnh thương mại, mà chú trọng vào thiết kế kiến trúc bảo mật và triển khai các thuật toán mật mã để đảm bảo tính an toàn cho hệ thống.

# Chương I. TỔNG QUAN

## 1. Ngữ cảnh

- Sự bùng nổ của các nền tảng truyền thông đa phương tiện (Spotify, Netflix, v.v.) kéo theo yêu cầu ngày càng khắt khe về bảo mật:

* **Bản quyền** – đảm bảo nội dung nhạc không bị sao chép, phát tán trái phép.
* **Bảo vệ dữ liệu cá nhân** – thông tin tài khoản, lịch sử nghe, thống kê doanh thu cho nghệ sĩ.

- Đồ án của nhóm nhằm xây dựng một hệ thống streaming nội bộ, minh hoạ cách **củng cố bảo mật** end-to-end bằng các kỹ thuật mật mã hiện đại (ECDHE, ECDSA, AES-GCM) kết hợp **HashiCorp Vault** làm KMS.

## 2. Tài sản, các bên liên quan và vai trò

### 2.1. Các bên liên quan và vai trò

* Các bên liên quan trong Spotify:

| Stakeholder | Role |
| --- | --- |
| Nghệ sĩ/Hãng thu âm | Cung cấp bài hát, album, podcast độc quyền cho Spotify. |
| Người dùng Spotify | Sử dụng dịch vụ streaming và tải xuống nội dung offline. |
| Cơ quan quản lý (ví dụ: IFPI, GDPR) | Giám sát tuân thủ pháp lý về bản quyền và bảo mật dữ liệu. |
| Quản trị viên Spotify | Vận hành hệ thống, quản lý rủi ro bảo mật, cập nhật tính năng. |
| Nhà phát triển Spotify | Tích hợp và tối ưu hóa thuật toán mã hóa, cải thiện trải nghiệm người dùng. |
| Đối tác thanh toán (ví dụ: Visa, PayPal) | Xử lý giao dịch đăng ký Premium. |

* Vai trò và trách nhiệm trong Spoitfy:

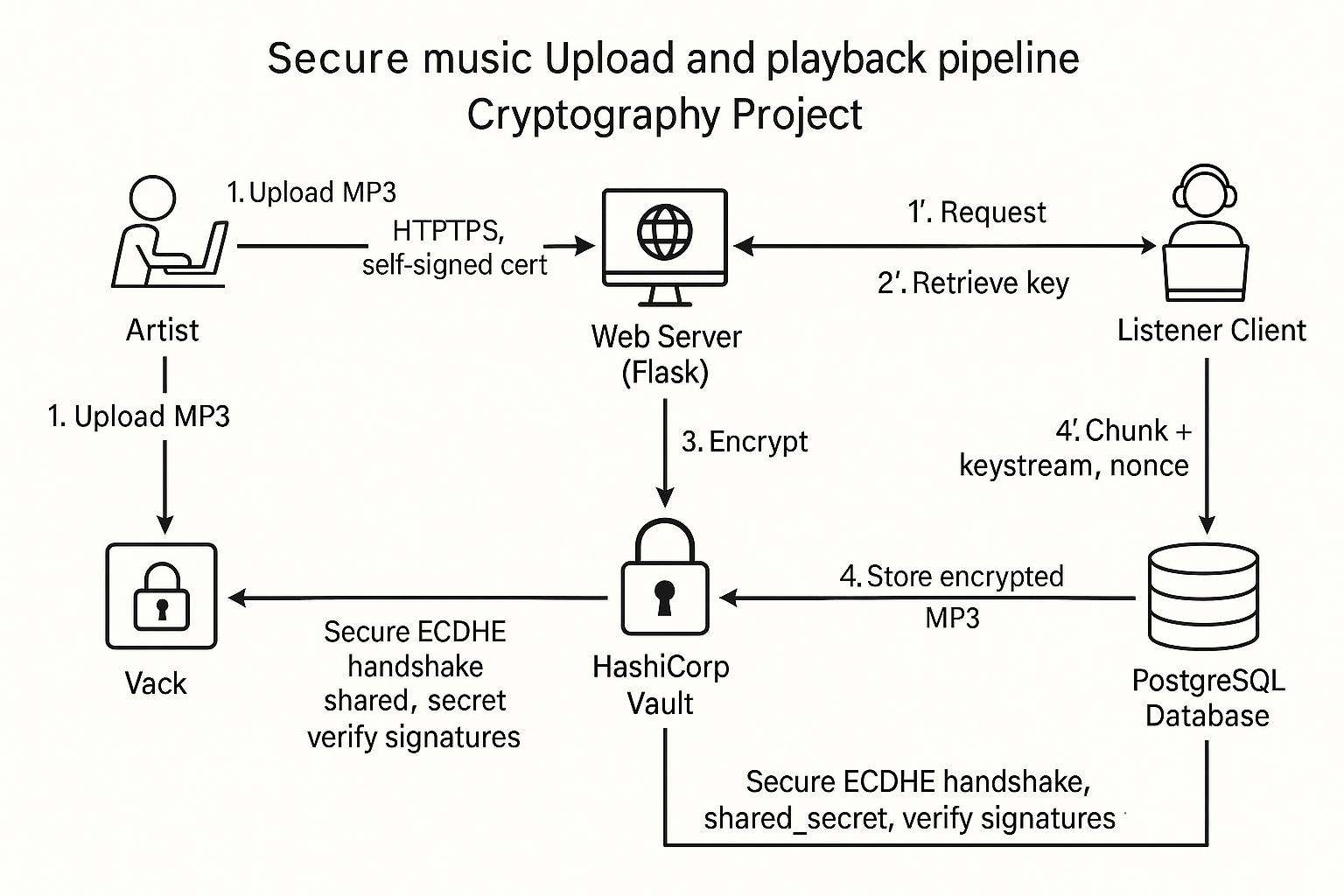
| Role | Trách nhiệm | Quyền hạn |
| --- | --- | --- |
| Quản trị viên bảo mật | - Triển khai DRM và mã hóa AES-256 cho nội dung tải xuống.  - Giám sát tấn công mạng và rò rỉ dữ liệu. | - Truy cập hệ thống mã hóa.  - Điều chỉnh chính sách bảo mật. |
| Người dùng Premium | - Nghe nhạc không gián đoạn, tải xuống bài hát offline.  - Báo cáo lỗi phát hoặc vi phạm. | - Truy cập nội dung cao cấp.  - Lưu trữ offline an toàn. |
| Nhà cung cấp nội dung | - Tải lên bài hát mới, quản lý metadata.  - Theo dõi lượt stream và doanh thu. | - Chỉnh sửa thông tin bài hát.  - Yêu cầu gỡ nội dung vi phạm. |
| Kỹ sư vận hành Spotify | - Duy trì máy chủ streaming, tối ưu CDN.  - Triển khai mã hóa dòng hỗn loạn cho luồng phát trực tiếp. | - Nâng cấp phần cứng/mềm.  - Giám sát hiệu suất hệ thống. |
| Người phụ trách pháp lý | - Đánh giá tuân thủ GDPR và luật bản quyền.  - Kiểm tra logs truy cập nội dung. | - Truy cập báo cáo hoạt động.  - Đề xuất cải tiến chính sách. |

### 2.2. Tài sản cần bảo vệ

* Bảng thống kê các tài sản quan trọng trong Spotify:

| **Loại tài sản** | **Mô tả** |
| --- | --- |
| Dữ liệu người dùng | Thông tin cá nhân (email, mật khẩu), lịch sử thanh toán (thẻ tín dụng), lịch sử nghe nhạc. |
| Nội dung âm nhạc và podcast (quan trọng nhất) | Bài hát, album, playlist, podcast độc quyền |
| Tài sản trí tuệ | Bản quyền âm nhạc, thuật toán mã hóa (AES-256). |
| Khóa mã hóa | Khóa phiên (session keys) cho streaming thời gian thực, khóa lưu trữ trong KMS |
| Hệ thống máy chủ Spotify | Máy chủ streaming, cơ sở dữ liệu lưu trữ metadata và nội dung, CDN (Content Delivery Network). |

* Đường đi của tài sản quan trọng nhất (Tài sản âm thanh của nhà cung cấp):



## 3. Các rủi ro bảo mật và tính năng cần xây dựng

## 

| **Nhóm rủi ro (STRIDE)** | **Kịch bản cụ thể** | **Tính năng/biện pháp đáp ứng** |
| --- | --- | --- |
| S – Spoofing | Giả mạo máy chủ để cấp file giả | • Server ký ECDHE-pub bằng ECDSA  • Client kiểm tra chữ ký trước khi tính shared\_secret |
| T – Tampering | Thay đổi chunk trong quá trình truyền | • AES-GCM cung cấp xác thực (tag) cho từng chunk  • TLS/self-signed cert cho kênh HTTPS nội bộ |
| R – Repudiation | Phủ nhận upload/stream | • Log có chữ ký số của server;  • Lưu vết play\_history trong DB (hash integrity) |
| I – Information Disclosure | Lộ file nhạc hoặc khoá AES | • File nhạc luôn ở dạng mã hoá tại rest  • Khoá lưu trong Vault, truy cập qua token gắn hạn |
| D – Denial of Service | Flood request hoặc giữ kết nối treo | • Giới hạn tốc độ, timeout chunk ACK  • Hạn mức parallel stream/địa chỉ IP |
| E – Elevation of Privilege | Người nghe khai thác API upload | • RBAC: chỉ role artist thấy /upload  • CSRF token, session isolation |

## 

# Chương II. GIẢI PHÁP

## 1. Kiến trúc giải pháp

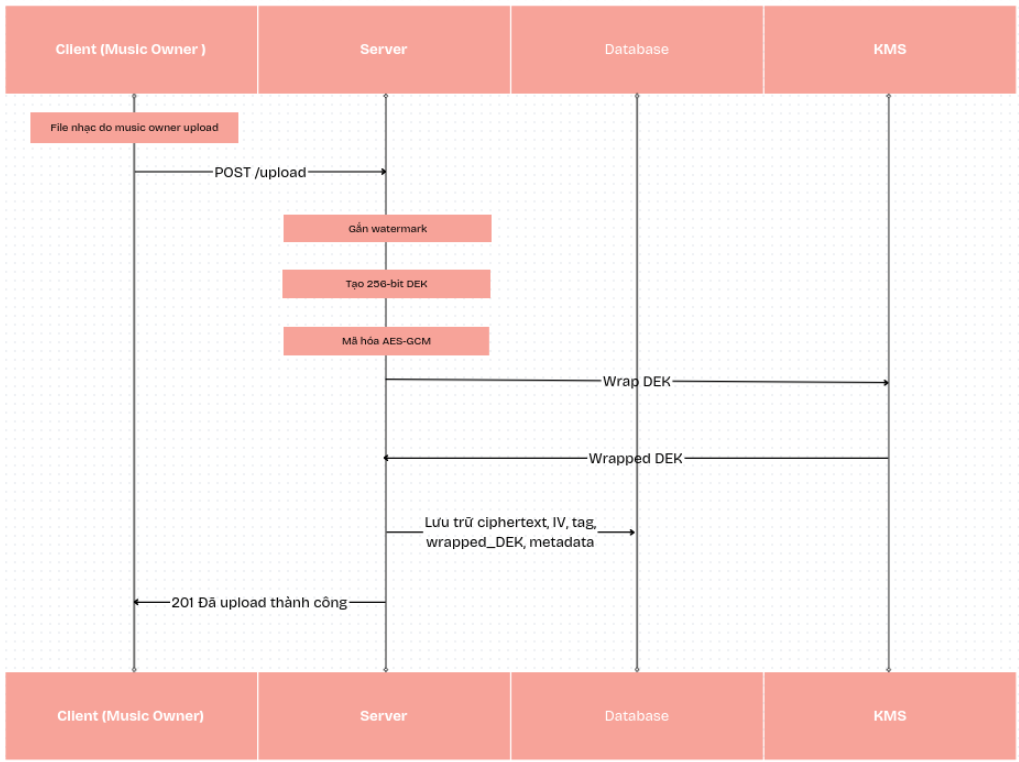
### 1.1. Thuật toán và giao thức mã hóa

Hệ thống sử dụng các thuật toán mã hóa hiện đại và đã được chuẩn hóa:

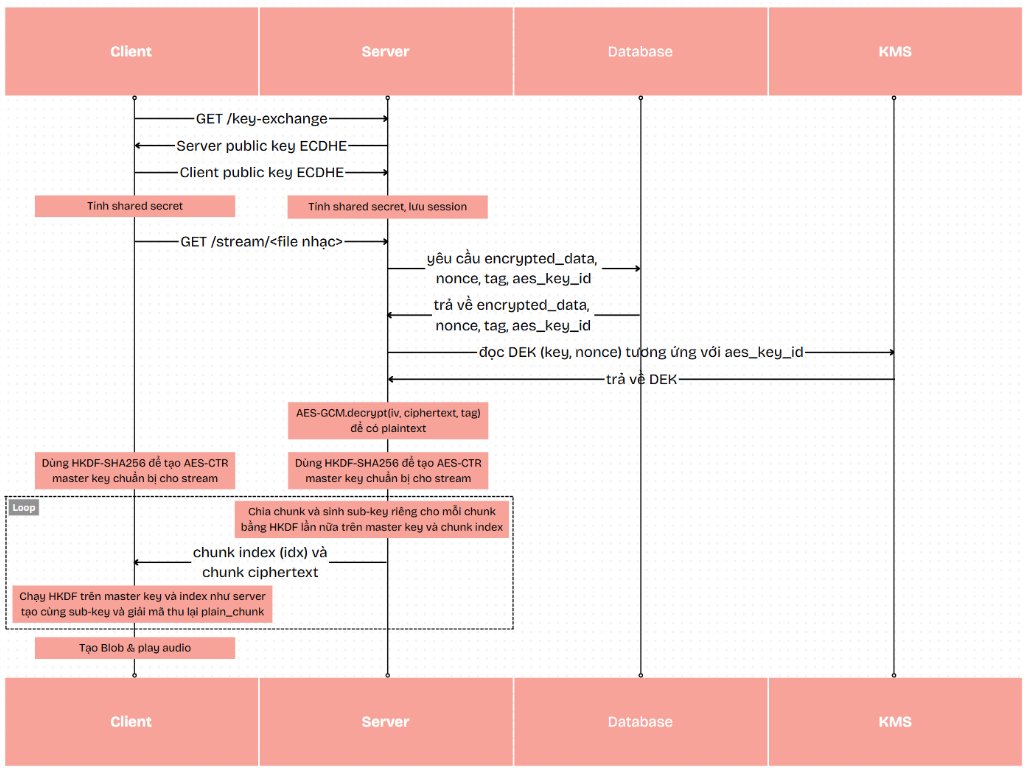
* ECDHE (Elliptic Curve Diffie-Hellman Ephemeral): trao đổi khóa tạm thời tạo shared secret.
* ECDSA: xác thực chữ ký số giữa client và server nhằm tránh MITM.
* AES-GCM (128-bit key, 96-bit nonce): mã hóa at-rest toàn bộ file MP3 trước khi lưu vào database, đồng thời đảm bảo integrity thông qua authentication tag.
* HKDF-SHA256: derive từ shared\_secret của ECDHE ra key 256-bit dùng cho AES-CTR và tiếp tục derive key subkey cho mỗi chunk âm thanh từ shared\_secret.
* AES-CTR (256-bit key): stream cipher cho từng chunk (64 KB) khi truyền tải, hỗ trợ parallelism và random-access.

Các thư viện mã hóa PyCryptodome (Python), và Web Crypto API (JavaScript).

### 1.2. Thành phần hệ thống



*Luồng upload file âm thanh*



*Luồng nghe nhạc*

Client (Browser + JS)

* Xác thực server.
* Tạo khóa ECDHE.
* Tính shared secret, sinh keystream AES-CTR để giải mã các chunk dữ liệu âm thanh.
* Xóa các chunk khỏi RAM sau khi phát.

Server (Flask/Python)

* Xác thực client.
* Sinh khóa ECDHE, tính shared-secret.
* Upload: AES-GCM encrypt + lưu vào Vault KV v2 & PostgreSQL.
* Stream: đọc encrypted\_data từ database, unwrap key từ Vault → AES-GCM.decrypt → plaintext.
* Derive AES-CTR key bằng HKDF → chunk & AES-CTR.encrypt → các stream frame.

Database (PostgreSQL)

* Bảng users (lưu thông tin nghệ sĩ): username, email, password\_hash, role.
* Bảng songs (lưu thông tin bài hát): filename, title, artist, length, aes\_key\_id, nonce, tag, encrypted\_data.
* Bảng song\_stats (lưu thông tin để hỗ trợ chi trả cho nghệ sĩ): play\_count (số lượt nghe) và revenue (doanh thu).

KMS (HashiCorp Vault)

* KV v2 mount secret/music/ lưu { key, nonce }.
* Chỉ unwrap key khi server có token hợp lệ; không bao giờ lưu key plaintext trong cơ sở dữ liệu.

## 2. Môi trường triển khai

Client:

* Viết bằng JavaScript, chạy trong trình duyệt (browser).
* Sử dụng Web Crypto API để tạo khóa ECDHE, xác minh chữ ký ECDSA, tính shared\_secret, giải mã và phát từng chunk audio.

Web server:

* Apache Web Server + mod\_ssl để phục vụ HTTPS.
* Backend viết bằng Flask.

KMS:

* Dùng HashiCorp Vault chạy độc lập để lưu key AES-GCM và xác thực truy cập.

Database:

* PostgreSQL.
* Lưu dữ liệu người dùng và nhạc đã mã hóa bằng AES-GCM.

## 3. Phân tích chi tiết

### 2.1. Nhà cung cấp đăng tải nhạc

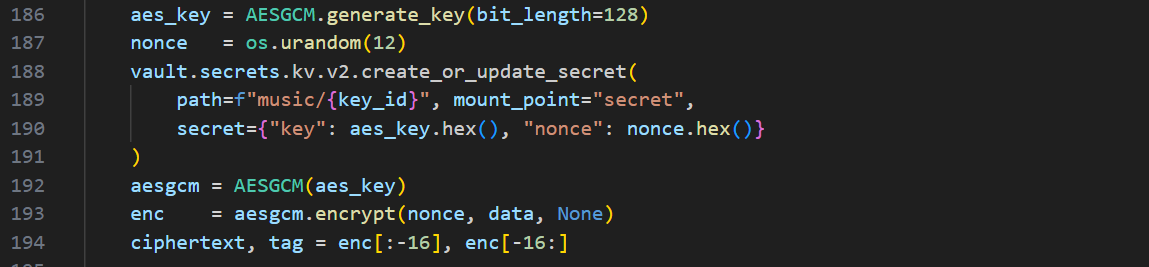
Giao diện upload: File templates/upload.html định nghĩa form gửi lên endpoint /upload với phương thức POST



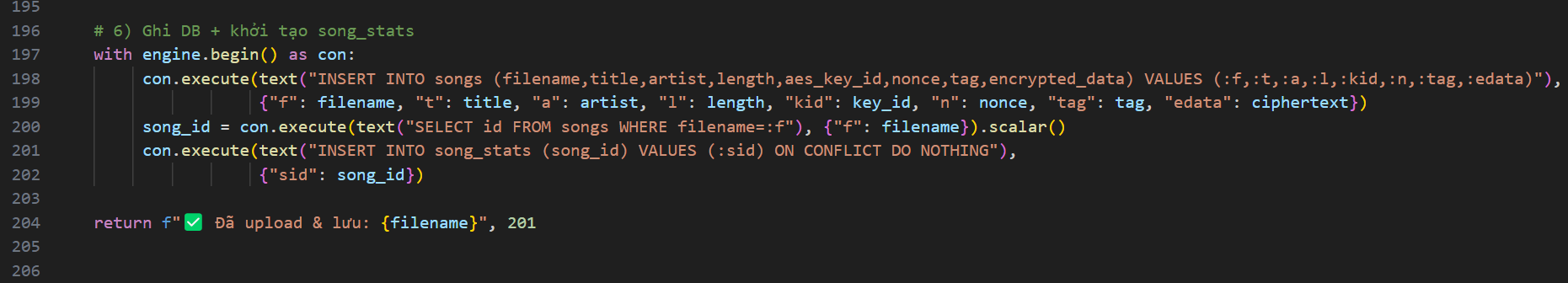
Route xử lý upload: Được khai báo trong app.py, với các logic trên web cơ bản sau:

* Xác thực và kiểm tra (chỉ artist được upload)
* GET → Trả về trang upload: render\_template("upload.html")
* POST → Nhận file: request.files.get["file"]
* Kiểm tra phần mở rộng, chỉ cho phép (.mp3), ngăn path traversal và ký tự bất hợp lệ.
* Đọc nội dung file và lấy data: data = file.read()

Mã hoá AES-GCM và lưu key vào Vault: Sinh key, nonce. Sau đó lưu vào HashiCorp Vault rồi thực hiện encrypt.



Lưu dữ liệu vào database và trả về kết quả thành công:

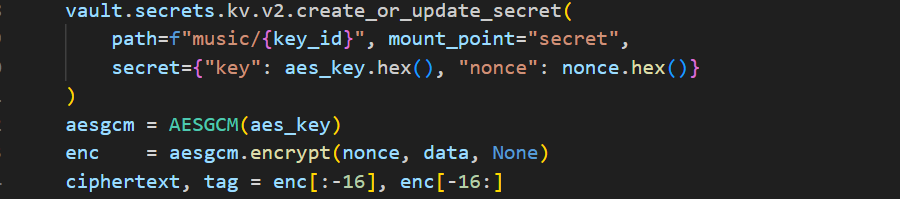


### 2.2. Lưu trữ key AES-GCM

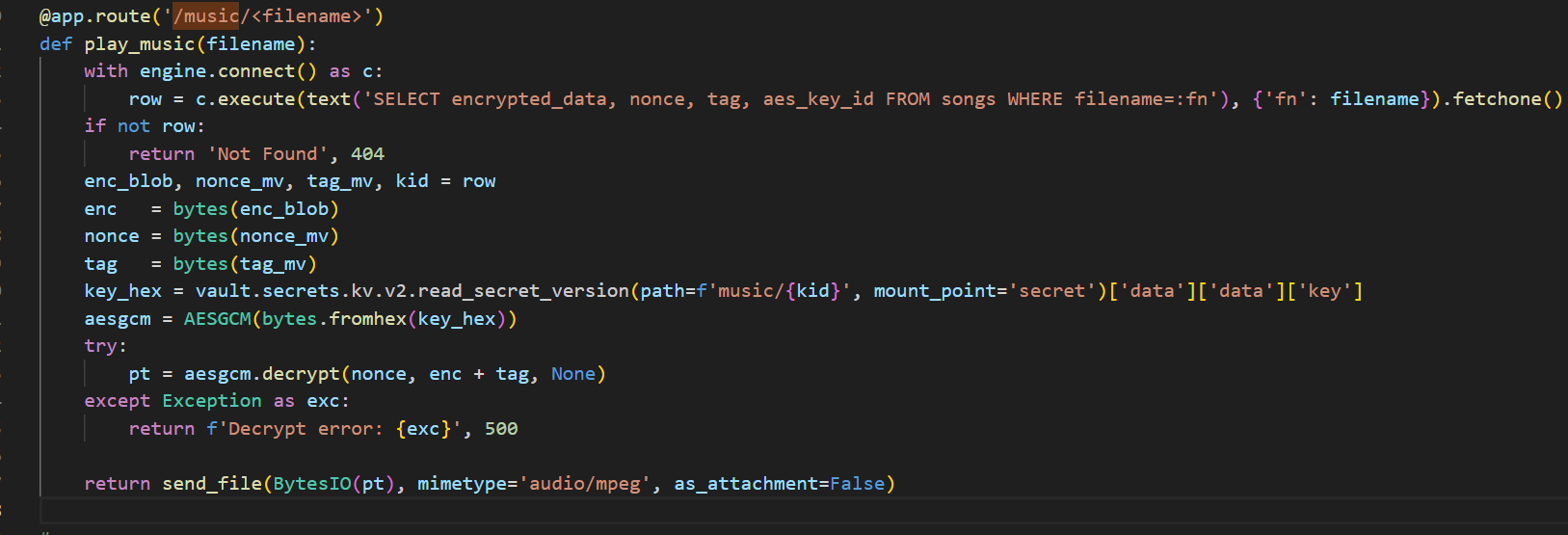
Trong route /upload của app.py, mỗi file MP3 sẽ có AES-GCM key 16 byte và nonce 12 byte để đảm bảo tính duy nhất cho mỗi lần encrypt.



Ngay sau khi sinh, key và nonce được chuyển về dạng hex và lưu vào HashiCorp Vault (KV Engine) tại đường dẫn secret/music/{key\_id}



Khi client yêu cầu phát nhạc, server đọc lại key từ Vault. Sau đó, server chuyển từ kiểu dữ liệu hex sang kiểu dữ liệu bytes rồi dùng AES-GCM để giải mã.



Các thông tin truy xuất key sẽ được đề cập ở phần tiếp theo (Cơ sở dữ liệu).

### 2.3. Cơ sở dữ liệu

Cơ sở dữ liệu bọn em thiết kế gồm 3 bảng: users (Quản lý tài khoản), songs (Lưu bài hát và các thông tin liên quan), song\_stats (Theo dõi số lượt phát và doanh thu). Em sẽ giải thích sơ qua về các cột của mỗi bảng:

Bảng users:

* username: tên tài khoản
* Email: email đăng ký
* password\_hash: password được hash để không lưu plaintext.
* Role: vai trò (user hoặc artist)

Bảng songs:

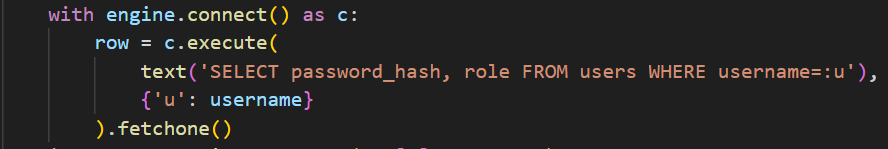
* id: id bài hát
* title: tiêu đề
* artist: nghệ sĩ đăng tải
* length: thời lượng (giây)
* aes\_key\_id: khóa tham chiếu trong Vault, dùng để lấy được key giải mã.
* nonce: 12 byte nonce cho AES-GCM
* Tag: 16 byte AES-GCM tag
* encrypted\_data: Nội dung file MP3 đã mã hóa

Bảng songs\_stats:

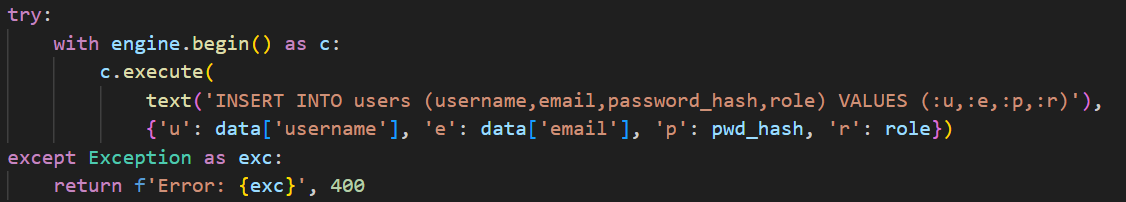
* song\_id: id bài hát cần theo dõi
* play\_count: số lần được phát
* revenue: Doanh thu

Các đoạn logic truy xuất, sử dụng cơ sở dữ liệu:

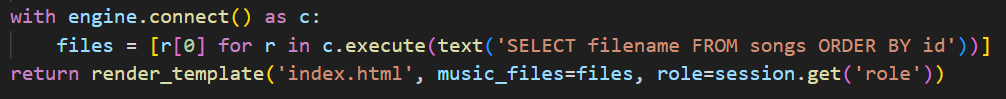
Đăng nhập: tạo đối tượng SQL text, cho phép binding parameter :u an toàn (tránh SQL injection). Thực thi truy vấn 'SELECT password\_hash, role FROM users WHERE username=:u' và lấy dòng kết quả đầu tiên (hoặc none nếu không có).



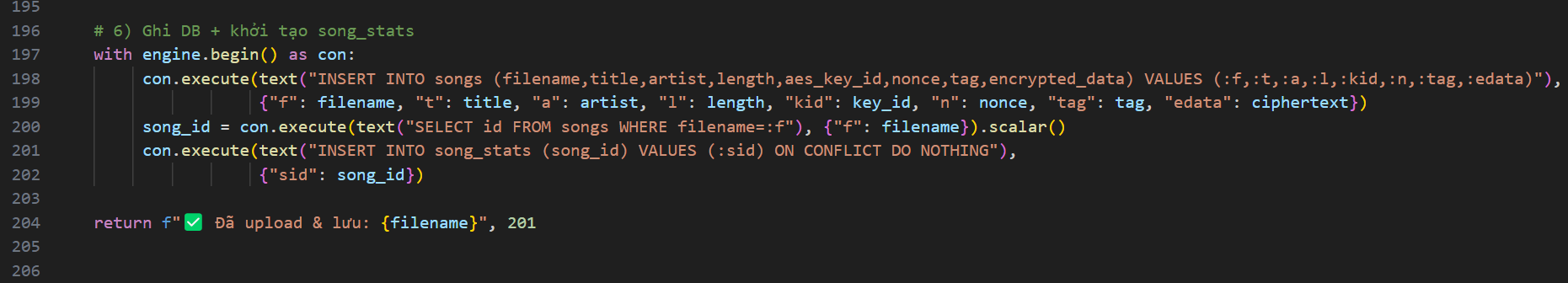
Đăng ký: mở một transaction; khi khối with kết thúc mà không có lỗi, SQLAlchemy tự động commit; nếu có exception, tự động rollback. Dòng INSERT chèn bản ghi người dùng mới vào bảng users. Dùng binding parameters (:u, :e, :p, :r) để tăng tính an toàn.



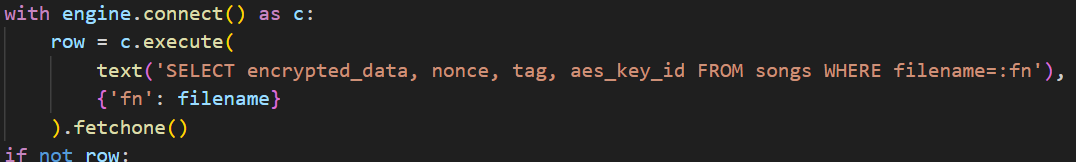
Lấy danh sách nhạc:



Upload nhạc: Đã miêu tả ở trên



Giải mã toàn bộ nhạc: Truy vấn 4 trường cần thiết để giải mã là encrypted\_data, nonce, tag, aes\_key\_id. Chi tiết phần giải mã sẽ miêu tả ở dưới.

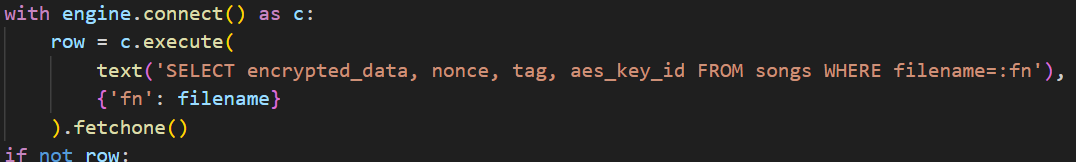


Cập nhật thông tin bài hát khi streaming (số lượt nghe, doanh thu): Mỗi khi client đã nhận đủ chunk hoặc hoàn thành stream, server gọi đoạn này để tăng play\_count lên 1 và tăng revenue lên 0.005

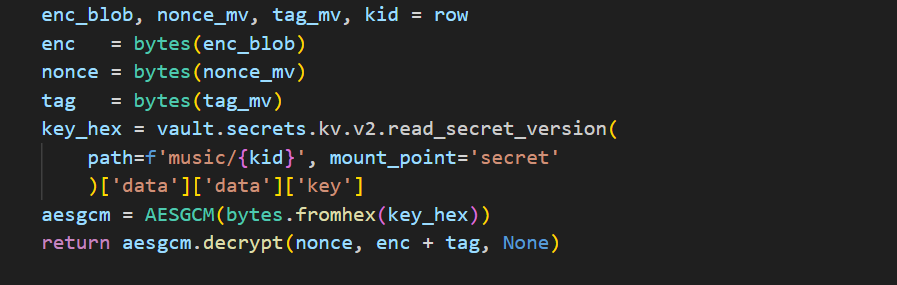


### 2.4. Lấy nhạc và giải mã

Client gửi GET đến /stream/<filename> để yêu cầu phát file. Truy xuất dữ liệu mã hóa từ cơ sở dữ liệu:

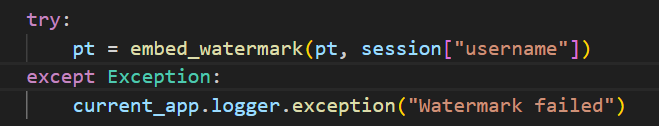


Rút AES-GCM key từ Vault. Kết hợp ciphertext và tag để giải mã, đồng thời xác thực integrity bằng GCM tag.

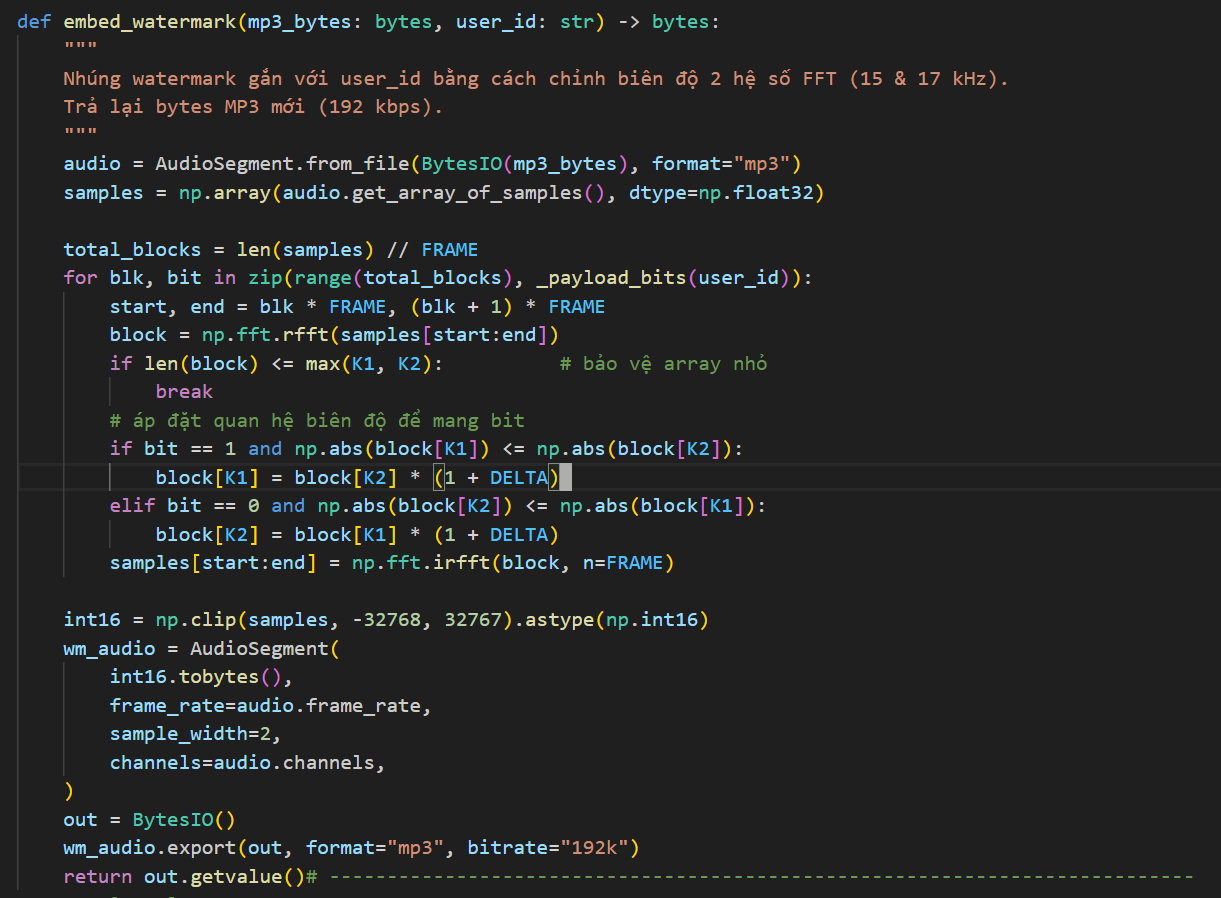


### 2.5. Gắn watermark

Sau khi giải mã AES-GCM, nhúng watermark vào file âm thanh vừa giải mã bằng FFT 15 kHz/17 kHz:



Đây là hàm embed\_watermark, sử dụng nguyên lý chính là điều chỉnh tương quan biên độ giữa 2 hệ số FFT nằm ngoài vùng nghe được (khoảng 15 kHz & 17 kHz) trên từng khối tín hiệu ngắn, gần như không làm giảm chất lượng âm thanh mà vẫn tồn tại sau khi dữ liệu được mã hóa truyền qua mạng. Đây là cơ chế nhẹ nhưng hiệu quả để điều tra pháp chứng số nếu file âm thanh bị sao chép và phổ biến bên ngoài.



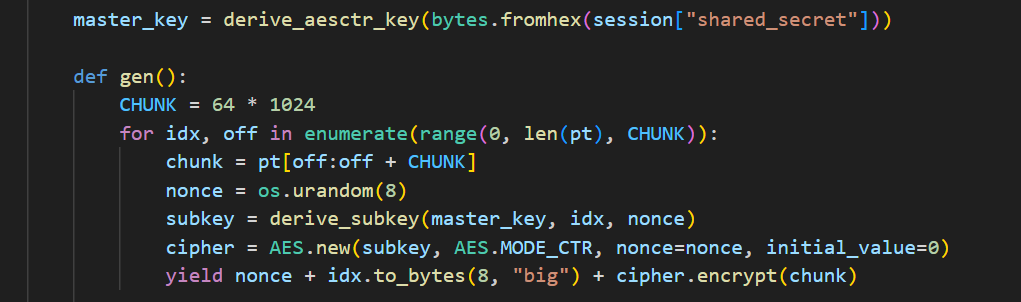
### 2.6. Xử lý file âm thanh trước khi phát:

Đầu tiên, sinh master key AES-CTR bằng HKDF-SHA256 trên shared secret làm khóa chính để sinh ra các khóa con cho mỗi chunk sau này.

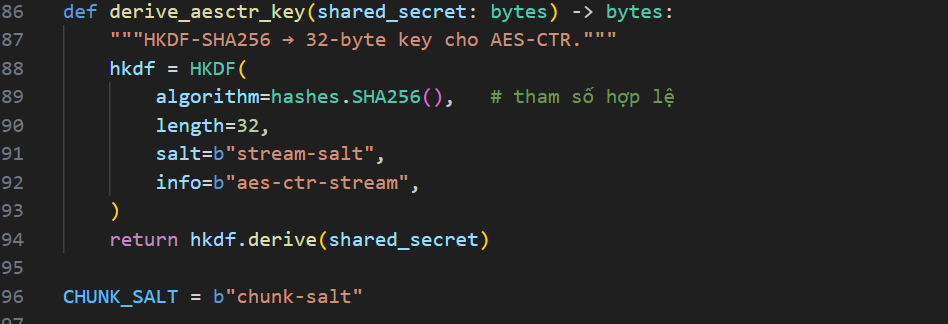
Sau đó, thực hiện stream theo generator, mỗi chunk sẽ có: nonce ngẫu nhiên cho mỗi chunk, sinh bằng hàm random; index là số thứ tự cho mỗi chunk; ciphertext là đoạn âm thanh của chunk được mã hóa.

Tiếp theo, một lần nữa sinh subkey bằng HKDF-SHA256 cho mỗi chunk dựa vào master-key, index và nonce.

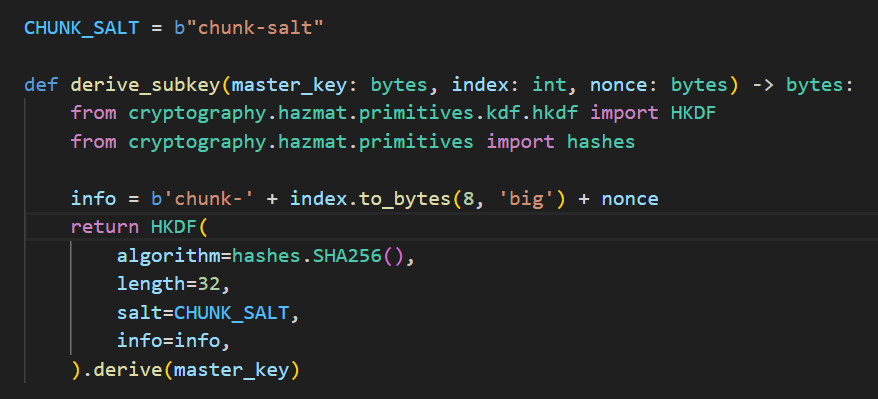
Đoạn code sinh master key và stream theo generator:



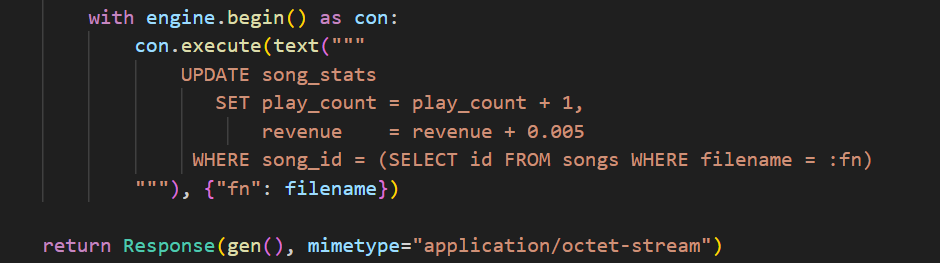
Hàm sinh master key



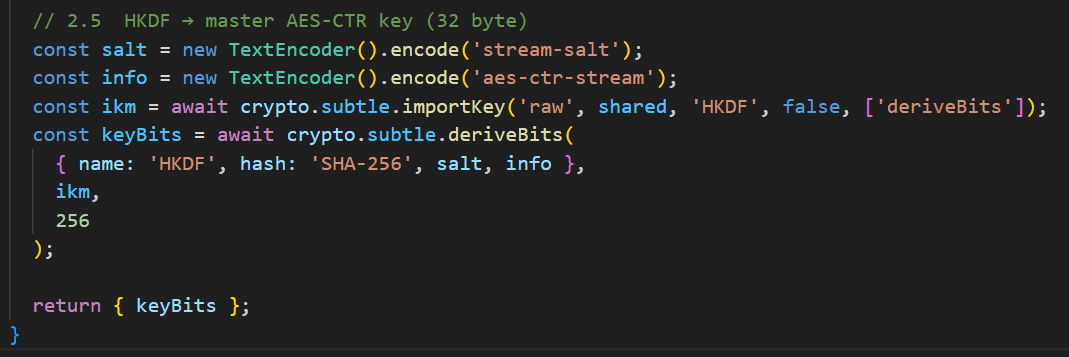
Hàm sinh subkey cho mỗi chunk



Sau khi gửi xong: cập nhật lại play\_count & revenue để thực hiện theo dõi doanh thu cho nhà cung cấp.

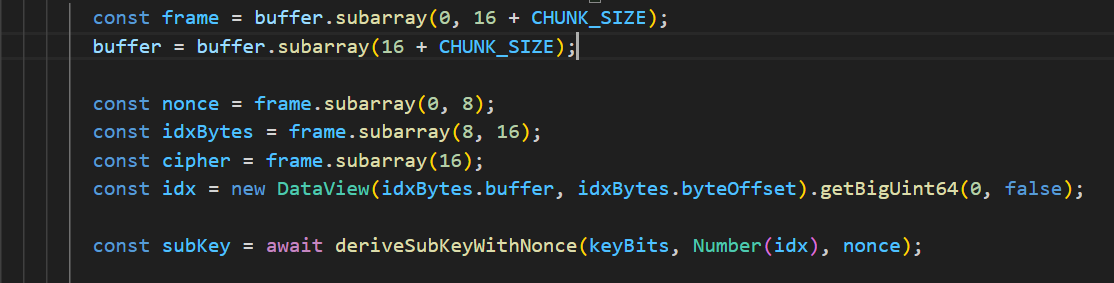


Tương tự server, phía client cũng sinh master key để chuẩn bị giải mã:

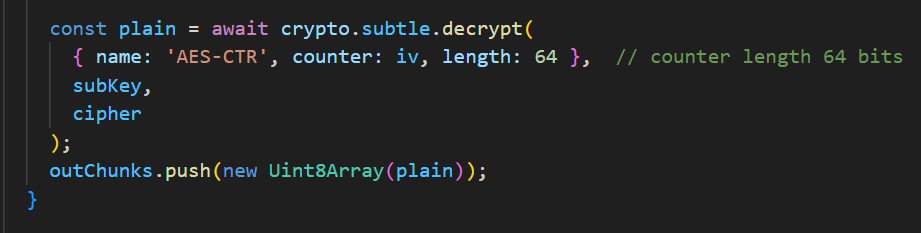


Sau đó, client nhận và giải mã luồng nhạc từ server gửi về với các bước sau:

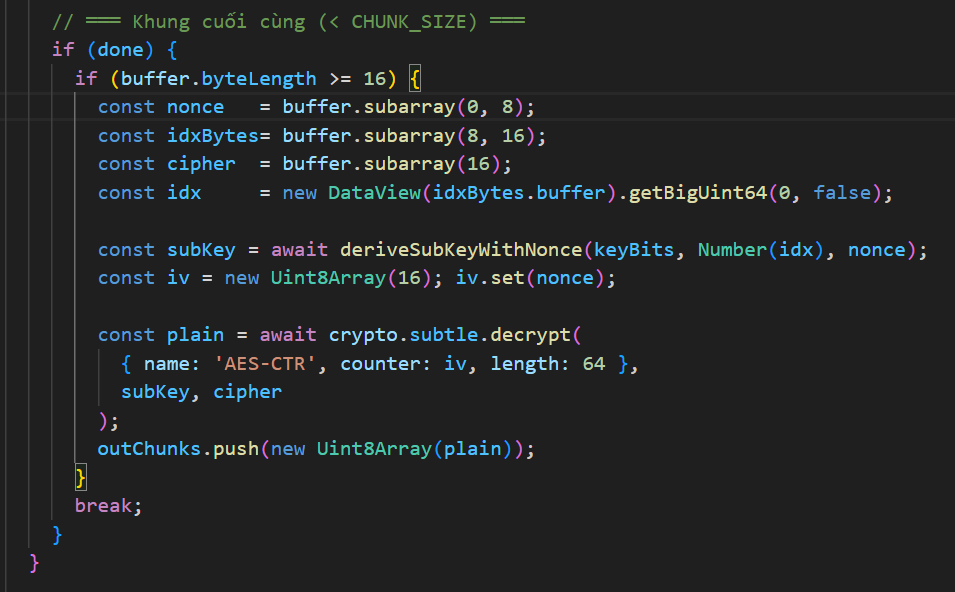
Tạo subkey cho chunk tương tự như server, giúp mỗi chunk có khoá riêng, tái sử dụng keystream được loại bỏ.



Sau khi có subkey, tiến hành giải mã AES-CTR cho từng chunk, riêng chunk cuối được xử lý riêng vì có kích thước nhỏ hơn CHUNK\_SIZE (16 bytes header + 64 KB mã hóa).

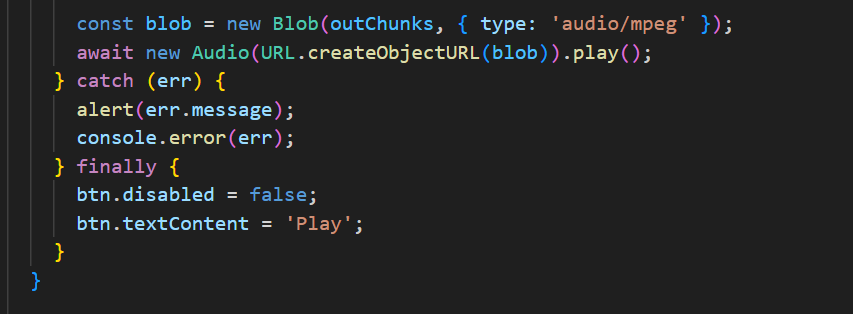


*Đoạn giải mã AES-CTR mỗi chunk*



*Đoạn giải mã chunk cuối (vì có độ dài khác các chunk trước)*

Cuối cùng, client tiến hành ghép và phát âm thanh:



# Chương III. TRIỂN KHAI

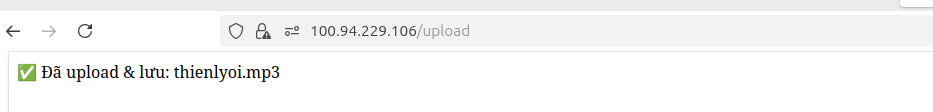
## 1. Nhà cung cấp đăng tải nhạc

- Đăng nhập vào trang web với tài khoản được cấp role "artist":

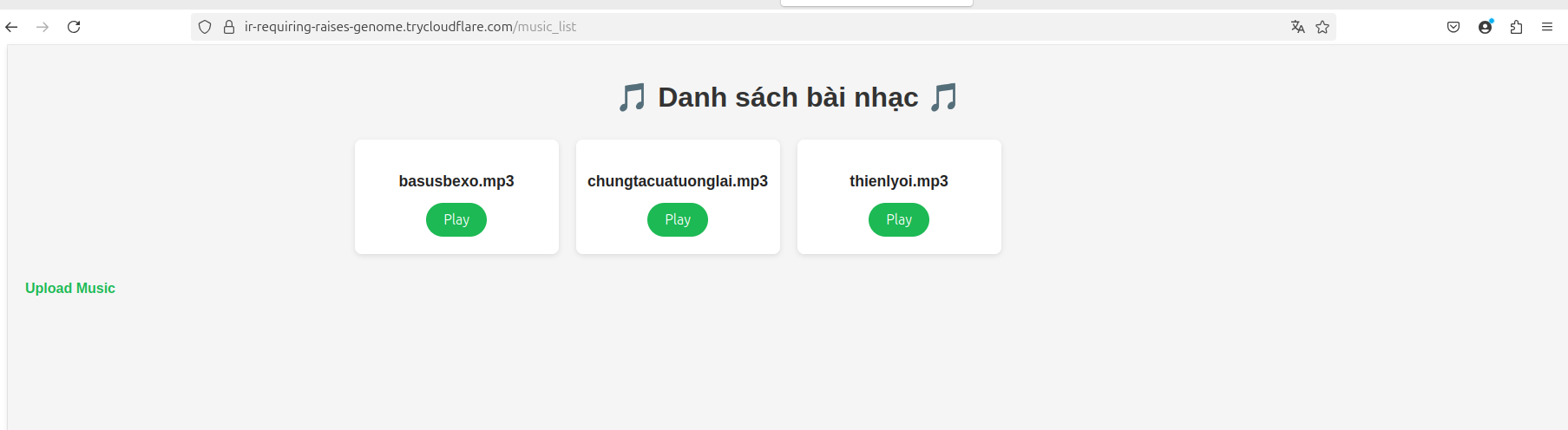
- Với role "artist" thì người cung cấp tài sản âm thanh có thể upload (đã có các bước sàng lọc và xác thực):



- Sau khi upload thành công sẽ có thông báo trả về.

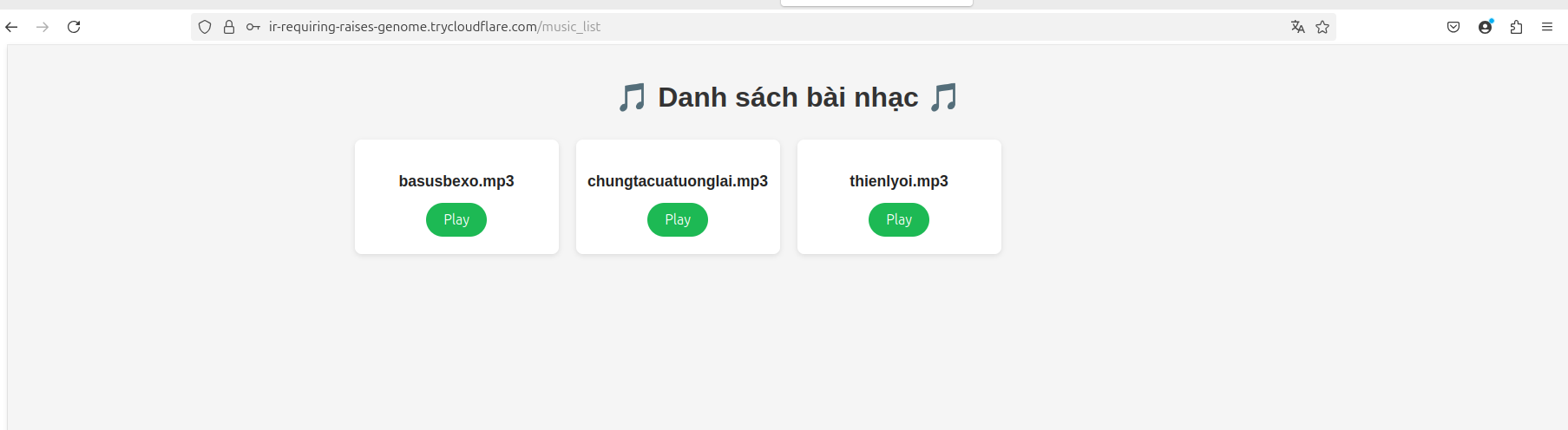


- Update danh sách nhạc:



## 2. Người dùng yêu cầu phát nhạc

- Khi người dùng đăng nhập tài khoản với role "user" thì sẽ có giao diện sau:



- Khi người dùng bấm play thì server xử lí logic, tương tác với database và KMS như đã đề cập ở phần **3. Phân tích chi tiết** sau đó người dùng sẽ được nghe nhạc.

# THAM KHẢO

1. V. Murari, R. K. C., and M. E. Raghu, Selective encryption of the audio extracted from the video streamed over the content delivery network, Proc. 2021 IEEE Mysore Sub Section International Conference (MysuruCon), Hassan, India, pp. 1–5, Oct. 2021.
2. M. Charfeddine, E. Mezghani, S. Masmoudi, C. B. Amar, and H. Alhumyani, Audio watermarking for security and non-security applications, IEEE Access, vol. 9, pp. 163157–163169, 2021