**TRƯỜNG ĐẠI HỌC ĐẠI NAM**

**KHOA CÔNG NGHỆ THÔNG TIN**

Logo, company name

Description automatically generated

**BÀI TẬP LỚN**

**TÊN HỌC PHẦN: AN TOÀN BẢO MẬT THÔNG TIN**

**ĐỀ TÀI: GIẢ LẬP UPLOAD/DOWNLOAD TÀI LIỆU LÊN NHIỀU ICLOUD**

**Giảng viên hướng dẫn: TS. Trần Đăng Công**

**ThS. Nguyễn Văn Nhân**

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Mã Sinh Viên | Họ và Tên | Lớp |
| 1 | **1771020087** | **Nguyễn Mạnh Bình** | CNTT 17-06 |
| 2 | **1771020062** | **Nguyễn Ngọc Ánh** | CNTT 17-06 |
| 3 | **1771020639** | **Nguyễn Văn Thành** | CNTT 17-06 |

**HàNội,năm2025**

**TRƯỜNG ĐẠI HỌC ĐẠI NAM**

**KHOA CÔNG NGHỆ THÔNG TIN**

Logo, company name

Description automatically generated

**BÀI TẬP LỚN**

**TÊN HỌC PHẦN: AN TOÀN BẢO MẬT THÔNG TIN**

**ĐỀ TÀI: GIẢ LẬP UPLOAD/DOWNLOAD TÀI LIỆU LÊN NHIỀU ICLOUD**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **STT** | **Mã Sinh Viên** | **Họ và Tên** | **Ngày Sinh** | **Điểm** | |
| **Bằng Số** | **Bằng Chữ** |
| **1** | **1771020087** | **Nguyễn Mạnh Bình** | **26/10/2005** |  |  |
| **2** | **1771020062** | **Nguyễn Ngọc Ánh** | **14/02/2005** |  |  |
| **3** | **1771020639** | **Nguyễn Văn Thành** | **04/04/2005** |  |  |

**CÁN BỘ CHẤM THI**

**HàNội,năm2025**

**LỜI NÓI ĐẦU**

Trong bối cảnh ngành công nghiệp âm nhạc ngày càng phát triển, việc truyền tải các file âm thanh và thông tin bản quyền một cách an toàn giữa các studio và nền tảng phát trực tuyến trở thành yêu cầu cấp thiết. Hệ thống truyền file âm nhạc an toàn được thiết kế nhằm đảm bảo tính bảo mật, xác thực, và toàn vẹn của dữ liệu trong quá trình truyền tải. Hệ thống sử dụng các thuật toán mã hóa mạnh mẽ như Triple DES, RSA, SHA-512, và DES, kết hợp với giao thức thời gian thực Socket.IO để cung cấp một giải pháp hiệu quả, đáp ứng nhu cầu bảo vệ quyền sở hữu trí tuệ và ngăn chặn giả mạo dữ liệu.

Báo cáo này trình bày quá trình thiết kế, triển khai, và đánh giá hệ thống truyền file âm nhạc an toàn, tập trung vào ba giai đoạn chính: bắt tay (handshake), xác thực (ký số và trao đổi khóa), và truyền dữ liệu kèm kiểm tra toàn vẹn. Báo cáo cũng phân tích mã nguồn hiện tại, đánh giá hiệu quả của hệ thống thông qua thử nghiệm giả lập, và đề xuất các cải tiến để nâng cao hiệu suất và bảo mật.

**MỤC LỤC**

[CHƯƠNG 1. GIỚI THIỆU VÀ PHÂN TÍCH BÀI TOÁN 1](#_Toc202020194)

[1.1. Đặt vấn đề 1](#_Toc202020195)

[1.2. Phân tích yêu cầu 1](#_Toc202020196)

[1.3. Mục tiêu và phạm vi 3](#_Toc202020197)

[*1.3.1.* Mục tiêu 3](#_Toc202020198)

[*1.3.2.* Phạm vi 3](#_Toc202020199)

[CHƯƠNG 2. THIẾT KẾ VÀ TRIỂN KHAI HỆ THỐNG 5](#_Toc202020200)

[2.1. Mô tả thuật toán 5](#_Toc202020201)

[2.2. Chi tiết các bước quy trình 6](#_Toc202020202)

[*2.2.1.* Bắt tay (Handshake) 6](#_Toc202020203)

[*2.2.2.* Xác thực (Ký số và Trao đổi khóa) 8](#_Toc202020204)

[*2.2.3.* Truyền dữ liệu và kiểm tra tính toàn vẹn 10](#_Toc202020205)

[2.3. Phân tích mã nguồn 13](#_Toc202020206)

[*2.3.1.* Phân tích file server.js 13](#_Toc202020207)

[*2.3.2.* Phân tích file index.html 15](#_Toc202020208)

[CHƯƠNG 3. ĐÁNH GIÁ VÀ HƯỚNG PHÁT TRIỂN 19](#_Toc202020209)

[3.1. Thử nghiệm và Kết quả 19](#_Toc202020210)

[*3.1.1.* Truyền file có dung lượng nhỏ 19](#_Toc202020211)

[*3.1.2.* Truyền file có dung lượng lớn 20](#_Toc202020212)

[3.2. Đánh giá hiệu quả 21](#_Toc202020213)

[*3.2.1.* So sánh dữ liệu giải mã với dữ liệu gốc: 21](#_Toc202020214)

[*3.2.2.* Kiểm tra tính toàn vẹn của dữ liệu truyền tải: 22](#_Toc202020215)

[3.3. Phân tích đặc điểm thuật toán 22](#_Toc202020216)

[*3.3.1.* Triple DES (3DES): 22](#_Toc202020217)

[*3.3.2.* RSA (1024-bit): 23](#_Toc202020218)

[*3.3.3.* SHA-512: 24](#_Toc202020219)

[*3.3.4.* DES: 25](#_Toc202020220)

[KẾT LUẬN 26](#_Toc202020221)

# GIỚI THIỆU VÀ PHÂN TÍCH BÀI TOÁN

## Đặt vấn đề

Trong ngành công nghiệp âm nhạc, việc truyền tải các file âm thanh (như file song.mp3) từ nhà phát triển ứng dụng âm nhạc (studio) đến các nền tảng phát trực tuyến đòi hỏi mức độ bảo mật cao để bảo vệ quyền sở hữu trí tuệ và ngăn chặn truy cập trái phép. Hệ thống truyền file âm nhạc an toàn được thiết kế nhằm đáp ứng các yêu cầu sau:

* **Bảo mật dữ liệu**: File âm nhạc và thông tin bản quyền (metadata) cần được mã hóa để ngăn chặn việc nghe lén hoặc đánh cắp dữ liệu trong quá trình truyền tải.
* **Xác thực người dùng**: Đảm bảo chỉ những bên được ủy quyền (người gửi và người nhận) mới có thể tham gia vào quá trình truyền file.
* **Toàn vẹn dữ liệu**: File âm nhạc và metadata phải được kiểm tra để đảm bảo không bị thay đổi hoặc giả mạo trong quá trình truyền.

Hệ thống sử dụng các thuật toán mã hóa mạnh, bao gồm **Triple DES** để mã hóa file âm nhạc, **DES** để mã hóa metadata bản quyền, **RSA 1024-bit** với OAEP và SHA-512 cho trao đổi khóa và ký số, cùng với **SHA-512** để kiểm tra tính toàn vẹn. Giao diện người dùng được xây dựng để hiển thị trạng thái kết nối, thông tin file, và tiến trình mã hóa, giúp các bên dễ dàng theo dõi và quản lý quá trình truyền tải. Luồng xử lý bao gồm các bước: bắt tay (handshake) đơn giản, xác thực và trao đổi khóa, mã hóa và kiểm tra toàn vẹn dữ liệu, đảm bảo an toàn và hiệu quả trong truyền tải.

## Phân tích yêu cầu

Hệ thống truyền file âm nhạc an toàn cần đáp ứng ba yêu cầu cốt lõi để đảm bảo quá trình truyền tải file song.mp3 và metadata bản quyền từ nhà phát triển ứng dụng âm nhạc đến nền tảng phát trực tuyến được an toàn, xác thực, và toàn vẹn. Các yêu cầu cụ thể bao gồm:

* Mã hóa dữ liệu để đảm bảo tính bảo mật:
  + File âm nhạc: Sử dụng thuật toán Triple DES với khóa 168-bit (3 khóa 56-bit) ở chế độ CBC (Cipher Block Chaining) và padding PKCS7 để mã hóa nội dung file âm nhạc (ví dụ: song.mp3). Triple DES cung cấp mức độ bảo mật cao nhờ sử dụng ba giai đoạn mã hóa, giảm thiểu nguy cơ bị giải mã trái phép.
  + Metadata bản quyền: Mã hóa thông tin bản quyền (ví dụ: tên file, thông tin nghệ sĩ, album) bằng DES với khóa 56-bit, cũng ở chế độ CBC. Mặc dù DES có độ dài khóa ngắn hơn, nhưng nó phù hợp để bảo vệ metadata do kích thước nhỏ và yêu cầu bảo mật thấp hơn so với file chính.
  + Trao đổi khóa phiên: Sử dụng RSA 1024-bit với padding OAEP để mã hóa khóa phiên (session key) được sử dụng trong Triple DES. Điều này đảm bảo khóa phiên được truyền an toàn giữa người gửi và người nhận, ngăn chặn việc nghe lén.
* Xác thực người dùng:
  + Hệ thống sử dụng RSA 1024-bit kết hợp với SHA-512 để tạo chữ ký số cho metadata (bao gồm tên file và thông tin bản quyền). Chữ ký số này giúp xác minh danh tính của người gửi và đảm bảo rằng chỉ các bên được ủy quyền mới có thể tham gia quá trình truyền file.
  + Quá trình xác thực diễn ra trong giai đoạn trao đổi khóa, khi người gửi ký metadata và gửi cùng khóa phiên mã hóa. Người nhận sử dụng khóa công khai RSA để kiểm tra chữ ký, đảm bảo tính xác thực của dữ liệu.
* Kiểm tra tính toàn vẹn:
  + Sử dụng SHA-512 để tạo hàm băm (hash) của dữ liệu, bao gồm vector khởi tạo (IV) và ciphertext của file âm nhạc. Hàm băm này được gửi cùng gói tin để người nhận kiểm tra tính toàn vẹn.
  + Phía người nhận sẽ tính lại giá trị SHA-512 từ IV và ciphertext nhận được, so sánh với giá trị hash trong gói tin. Nếu khớp, dữ liệu được coi là toàn vẹn; nếu không, hệ thống từ chối và gửi thông báo NACK (lỗi toàn vẹn) đến người gửi.
  + Chữ ký số SHA-512 trên metadata cũng hỗ trợ kiểm tra tính toàn vẹn, đảm bảo metadata không bị thay đổi.

Những yêu cầu này đảm bảo hệ thống không chỉ bảo vệ dữ liệu khỏi truy cập trái phép mà còn xác minh danh tính người dùng và duy trì tính toàn vẹn của file âm nhạc và metadata trong suốt quá trình truyền tải.

## Mục tiêu và phạm vi

### Mục tiêu

* Xây dựng một hệ thống truyền file âm nhạc an toàn, đảm bảo bảo mật, xác thực, và toàn vẹn dữ liệu trong quá trình truyền tải file âm thanh (như song.mp3) và metadata bản quyền từ nhà phát triển ứng dụng âm nhạc (studio) đến nền tảng phát trực tuyến.
* Đảm bảo file âm nhạc được mã hóa bằng Triple DES và metadata bản quyền được mã hóa bằng DES, ngăn chặn truy cập trái phép hoặc nghe lén.
* Xác thực danh tính người gửi và người nhận thông qua RSA 1024-bit và chữ ký số SHA-512, đảm bảo chỉ các bên được ủy quyền tham gia vào quá trình truyền tải.
* Kiểm tra tính toàn vẹn của dữ liệu bằng hàm băm SHA-512, đảm bảo file và metadata không bị thay đổi hoặc giả mạo.
* Cung cấp giao diện người dùng thân thiện, hiển thị trạng thái kết nối, thông tin file, và tiến trình mã hóa, giúp người dùng dễ dàng theo dõi và quản lý quá trình truyền tải.

### Phạm vi

* Hệ thống tập trung vào việc triển khai các thuật toán mã hóa và xác thực, bao gồm:
  + Triple DES (khóa 168-bit, chế độ CBC, padding PKCS7) để mã hóa file âm nhạc.
  + DES (khóa 56-bit, chế độ CBC) để mã hóa metadata bản quyền.
  + RSA 1024-bit với padding OAEP để trao đổi khóa phiên và ký số với SHA-512 để xác thực.
  + SHA-512 để kiểm tra tính toàn vẹn của dữ liệu truyền tải.
* Hỗ trợ truyền tải file âm nhạc (như .mp3) và metadata bản quyền giữa studio (người gửi) và nền tảng phát nhạc (người nhận).
* Sử dụng Socket.IO để thiết lập kết nối thời gian thực giữa client và server, đảm bảo giao tiếp nhanh chóng và ổn định.
* Giao diện người dùng được xây dựng bằng HTML và JavaScript, hiển thị các trạng thái như kết nối, mã hóa, xác thực, và truyền file.
* Phạm vi không bao gồm việc xử lý các định dạng file khác ngoài .mp3 hoặc tích hợp các giao thức truyền tải khác ngoài Socket.IO.

# THIẾT KẾ VÀ TRIỂN KHAI HỆ THỐNG

## Mô tả thuật toán

Hệ thống truyền file âm nhạc an toàn sử dụng các thuật toán mã hóa và xác thực mạnh mẽ để đảm bảo bảo mật, xác thực người dùng, và toàn vẹn dữ liệu khi truyền file song.mp3 và metadata bản quyền từ studio đến nền tảng phát trực tuyến. Các thuật toán được sử dụng bao gồm **Triple DES**, **RSA 1024-bit**, **SHA-512**, và **DES**, được triển khai trong một quy trình ba bước: bắt tay (handshake), xác thực (ký số và trao đổi khóa), và truyền dữ liệu kèm kiểm tra toàn vẹn.

* Triple DES (3DES):
  + Mục đích: Mã hóa file âm nhạc (như song.mp3) để đảm bảo tính bảo mật.
  + Chi tiết: Sử dụng ba khóa 56-bit (tổng cộng 168-bit) trong chế độ CBC (Cipher Block Chaining) với vector khởi tạo (IV) ngẫu nhiên và padding PKCS7. Triple DES thực hiện mã hóa ba lần liên tiếp (mã hóa-giải mã-mã hóa) để tăng cường bảo mật so với DES, phù hợp cho dữ liệu âm thanh kích thước lớn.
* RSA 1024-bit:
  + Mục đích: Trao đổi khóa phiên an toàn và ký số để xác thực.
  + Chi tiết: Sử dụng khóa 1024-bit với padding OAEP để mã hóa khóa phiên của Triple DES, đảm bảo khóa không bị lộ trong quá trình truyền. Ký số metadata (tên file và thông tin bản quyền) bằng RSA kết hợp với SHA-512, cho phép người nhận xác minh danh tính người gửi.
* SHA-512:
  + Mục đích: Kiểm tra tính toàn vẹn và hỗ trợ ký số.
  + Chi tiết: Tạo hàm băm 512-bit từ IV và ciphertext của file âm nhạc để kiểm tra tính toàn vẹn, đảm bảo dữ liệu không bị thay đổi. SHA-512 cũng được sử dụng trong chữ ký số của metadata, cung cấp khả năng chống giả mạo.
* DES:
  + Mục đích: Mã hóa metadata bản quyền (tên file, thông tin nghệ sĩ, album).
  + Chi tiết: Sử dụng khóa 56-bit trong chế độ CBC với IV ngẫu nhiên. Mặc dù DES có bảo mật thấp hơn Triple DES, nó phù hợp cho metadata do kích thước nhỏ và yêu cầu bảo mật thấp hơn.
* Quy trình ba bước:
  1. Bắt tay (Handshake): Thiết lập kết nối giữa người gửi và người nhận thông qua giao thức đơn giản ("Hello!" từ người gửi, "Ready!" từ người nhận) sử dụng Socket.IO, đảm bảo cả hai bên sẵn sàng trước khi truyền dữ liệu.
  2. Xác thực (Ký số và Trao đổi khóa): Người gửi ký metadata bằng RSA/SHA-512 và mã hóa khóa phiên bằng RSA 1024-bit (OAEP), gửi đến người nhận để xác minh danh tính và thiết lập khóa an toàn.
  3. Truyền dữ liệu và Kiểm tra toàn vẹn: File âm nhạc được mã hóa bằng Triple DES, metadata bằng DES, và tạo hàm băm SHA-512 từ IV và ciphertext. Gói tin chứa IV, ciphertext, metadata mã hóa, hàm băm, và chữ ký số được gửi đi. Người nhận kiểm tra hash và chữ ký, giải mã nếu hợp lệ, hoặc từ chối nếu có lỗi.

Các thuật toán này được kết hợp để tạo ra một hệ thống truyền file an toàn, đáp ứng yêu cầu bảo mật, xác thực, và toàn vẹn dữ liệu, phù hợp với luồng xử lý đã được mô tả.

## Chi tiết các bước quy trình

### Bắt tay (Handshake)

Giai đoạn bắt tay (handshake) là bước đầu tiên trong quy trình truyền file âm nhạc an toàn, nhằm thiết lập kết nối đáng tin cậy giữa người gửi (studio) và người nhận (nền tảng phát trực tuyến). Quá trình này sử dụng giao thức thời gian thực được cung cấp bởi **Socket.IO**, đảm bảo cả hai bên sẵn sàng trước khi tiến hành các bước mã hóa và truyền dữ liệu.

* Cơ chế hoạt động:
  + Bước 1: Người gửi (client) gửi tín hiệu "Hello!" đến server thông qua kết nối Socket.IO. Tín hiệu này được gửi ngay sau khi client thiết lập thành công kết nối với server, được ghi nhận trong sự kiện connection của file server.js (hàm io.on('connection', (socket) => { console.log('🔌 Client connected'); })).
  + Bước 2: Server, đại diện cho người nhận, phản hồi bằng tín hiệu "Ready!" để xác nhận rằng nó đã sẵn sàng nhận dữ liệu. Phản hồi này đảm bảo rằng kết nối đã được thiết lập và không có lỗi mạng ban đầu.
  + Kết quả: Trạng thái kết nối trên giao diện người dùng (index.html) được cập nhật từ "Chờ kết nối..." thành "Đã kết nối" trong phần "Trạng thái Kết nối" của mục "Người Nhận".
* Mục đích:
  + Xác nhận rằng cả người gửi và người nhận đều hoạt động và có thể giao tiếp thông qua kênh Socket.IO.
  + Đảm bảo kết nối ổn định trước khi tiến hành các bước xác thực và truyền dữ liệu, tránh lãng phí tài nguyên nếu một trong hai bên không sẵn sàng.
* Triển khai trong mã nguồn:
  + Trong file server.js, Socket.IO được sử dụng để lắng nghe sự kiện connection và xử lý các tín hiệu ban đầu từ client. Hàm io.on('connection', (socket) => {...}) khởi tạo kết nối và ghi nhận trạng thái client trong console (🔌 Client connected).
  + Trong file index.html, giao diện người dùng hiển thị trạng thái kết nối trong phần "Người Nhận" với mục "Trạng thái Kết nối". Giao diện sử dụng các phần tử HTML như <p>Trạng thái Kết nối: <span>Chờ kết nối...</span></p> để trực quan hóa trạng thái kết nối, giúp người dùng dễ dàng theo dõi quá trình.
* Đặc điểm:
  + Giai đoạn bắt tay được thiết kế đơn giản, chỉ yêu cầu trao đổi hai tín hiệu ("Hello!" và "Ready!") để giảm độ phức tạp và thời gian thiết lập kết nối.
  + Không bao gồm các bước xác thực hoặc mã hóa ở giai đoạn này, vì mục tiêu chính chỉ là xác nhận kết nối cơ bản. Các biện pháp bảo mật sẽ được triển khai trong các giai đoạn tiếp theo (xác thực và truyền dữ liệu).
  + Sử dụng Socket.IO giúp đảm bảo giao tiếp thời gian thực, phù hợp với yêu cầu truyền file nhanh chóng và hiệu quả trong ngành công nghiệp âm nhạc.

### Xác thực (Ký số và Trao đổi khóa)

Giai đoạn xác thực (ký số và trao đổi khóa) là bước quan trọng để đảm bảo rằng chỉ các bên được ủy quyền (người gửi và người nhận) tham gia vào quá trình truyền file âm nhạc và metadata bản quyền. Quá trình này sử dụng **RSA 1024-bit** với padding OAEP để trao đổi khóa phiên an toàn và **SHA-512** để tạo chữ ký số, xác minh danh tính người gửi.

* Cơ chế hoạt động:
  + Bước 1: Gửi khóa công khai:
    - Khi client yêu cầu, server gửi khóa công khai RSA 1024-bit dưới dạng PEM thông qua sự kiện Socket.IO public\_key. Điều này được triển khai trong file server.js qua đoạn mã xử lý sự kiện request\_public\_key.
    - Client sử dụng khóa công khai này để mã hóa khóa phiên (session key) trước khi gửi đến server.
  + Bước 2: Mã hóa và gửi khóa phiên:
    - Người gửi tạo một khóa phiên ngẫu nhiên (session key) và mã hóa nó bằng khóa công khai RSA của server, sử dụng thuật toán RSA-OAEP với hàm băm SHA-512.
    - Khóa phiên đã mã hóa được gửi đến server dưới dạng Base64 thông qua sự kiện Socket.IO session\_key\_encrypted.
  + Bước 3: Giải mã khóa phiên:
    - Server sử dụng khóa riêng RSA để giải mã khóa phiên đã mã hóa, sử dụng RSA-OAEP với SHA-512 (rsaKeypair.privateKey.decrypt trong server.js).
    - Sau khi giải mã thành công, server gửi lại khóa phiên đã giải mã qua sự kiện session\_key\_decrypted để xác nhận với client.
  + Bước 4: Ký số metadata:
    - Người gửi ký metadata (bao gồm tên file và thông tin bản quyền) bằng khóa riêng RSA và hàm băm SHA-512 để tạo chữ ký số.
    - Chữ ký số này được gửi cùng dữ liệu để người nhận kiểm tra tính xác thực và toàn vẹn của metadata.
* Mục đích:
  + Đảm bảo rằng chỉ người nhận hợp lệ (sở hữu khóa riêng RSA) mới có thể giải mã khóa phiên, từ đó bảo vệ khóa phiên khỏi các bên thứ ba.
  + Xác minh danh tính của người gửi thông qua chữ ký số, đảm bảo metadata không bị giả mạo.
  + Thiết lập một khóa phiên an toàn để sử dụng trong quá trình mã hóa dữ liệu ở bước tiếp theo (mã hóa file bằng Triple DES và metadata bằng DES).
* Triển khai trong mã nguồn:
  + File server.js:
    - Tạo cặp khóa RSA 1024-bit bằng thư viện node-forge (forge.pki.rsa.generateKeyPair({ bits: 2048, workers: 2 })). Mã nguồn hiện tại sử dụng 2048-bit, vì sử dụng RSA 1024-bit với SHA-512 có thể gây ra lỗi "data too large for key size" do kích thước khóa không đủ để chứa padding OAEP và hash SHA-512.
    - Xử lý sự kiện request\_public\_key để gửi khóa công khai PEM cho client.
    - Xử lý sự kiện session\_key\_encrypted để giải mã khóa phiên bằng RSA-OAEP với SHA-512, đảm bảo tính bảo mật của khóa phiên.
    - Gửi phản hồi qua sự kiện session\_key\_decrypted để xác nhận việc nhận và giải mã khóa phiên thành công.
  + File index.html:
    - Giao diện hiển thị trạng thái của quá trình xác thực trong mục "Trạng thái Hệ thống", với trường "Session Key" được cập nhật từ "Chưa tạo" thành "Đã tạo" khi quá trình trao đổi khóa hoàn tất.
    - Mục "Xác thực" trong phần Người Nhận được cập nhật từ "Chưa xác thực" thành "Đã xác thực" sau khi chữ ký số được kiểm tra thành công (tuy nhiên, mã nguồn hiện tại chưa triển khai rõ ràng bước kiểm tra chữ ký).
* Đặc điểm:
  + Sử dụng RSA 1024-bit với OAEP padding và SHA-512 đảm bảo an toàn trong việc trao đổi khóa và ký số, phù hợp với các yêu cầu bảo mật cao của ngành công nghiệp âm nhạc.
  + Quá trình xác thực được thực hiện trước khi truyền dữ liệu, giúp giảm thiểu rủi ro từ các kết nối không đáng tin cậy.
  + Việc sử dụng Socket.IO cho phép giao tiếp thời gian thực, đảm bảo trao đổi khóa và xác thực diễn ra nhanh chóng và hiệu quả.

### Truyền dữ liệu và kiểm tra tính toàn vẹn

Giai đoạn truyền dữ liệu và kiểm tra toàn vẹn là bước cuối cùng trong quy trình truyền file âm nhạc an toàn, đảm bảo file âm thanh (song.mp3) và metadata bản quyền được mã hóa, truyền tải an toàn và được kiểm tra tính toàn vẹn để ngăn chặn giả mạo hoặc thay đổi dữ liệu trong quá trình truyền.

* Cơ chế hoạt động:
* Bước 1: Chuẩn bị dữ liệu:
  + Tạo IV: Người gửi tạo một vector khởi tạo (IV) ngẫu nhiên để sử dụng trong chế độ CBC của Triple DES và DES, đảm bảo tính ngẫu nhiên và tăng cường bảo mật mã hóa.
  + Mã hóa file âm thanh: File âm thanh (ví dụ: song.mp3) được mã hóa bằng thuật toán Triple DES với khóa phiên (session key) đã trao đổi ở bước xác thực, sử dụng chế độ CBC và padding PKCS7.
* Mã hóa metadata: Thông tin bản quyền (tên file và thông tin bản quyền) được mã hóa bằng thuật toán DES với một khóa riêng, cũng sử dụng chế độ CBC.
* Bước 2: Tính toán hash để kiểm tra toàn vẹn:
  + Người gửi tính toán giá trị băm SHA-512 của dữ liệu bao gồm IV nối với ciphertext (IV || ciphertext) để đảm bảo tính toàn vẹn của dữ liệu truyền đi.
  + Chữ ký số của metadata (được tạo ở bước xác thực bằng RSA/SHA-512) cũng được gửi kèm để xác minh nguồn gốc.
* Bước 3: Gửi gói tin:
  + Gói tin được gửi đến người nhận thông qua Socket.IO, bao gồm các thành phần sau:

A screenshot of a computer code

AI-generated content may be incorrect.

* Bước 4: Xử lý phía người nhận:
  + Người nhận kiểm tra tính toàn vẹn bằng cách tính lại giá trị băm SHA-512 của IV nối với ciphertext và so sánh với giá trị hash nhận được.
  + Kiểm tra chữ ký số của metadata bằng khóa công khai RSA của người gửi để xác minh tính xác thực.
  + Nếu cả hash và chữ ký hợp lệ:
    - Giải mã file âm thanh bằng Triple DES và metadata bằng DES.
    - Lưu file âm thanh (song.mp3) và thông tin bản quyền.
    - Gửi tín hiệu xác nhận (ACK) tới người gửi qua sự kiện Socket.IO ack\_received.
  + Nếu hash hoặc chữ ký không hợp lệ:
    - Từ chối dữ liệu và gửi tín hiệu NACK (lỗi integrity) tới người gửi.
* Mục đích:
  + Đảm bảo file âm thanh và metadata được truyền tải an toàn, không bị thay đổi hoặc giả mạo trong quá trình truyền.
  + Bảo vệ thông tin bản quyền thông qua mã hóa DES, đảm bảo chỉ người nhận hợp lệ có thể truy cập.
  + Cung cấp cơ chế xác minh tính toàn vẹn và nguồn gốc dữ liệu thông qua SHA-512 và chữ ký số RSA.
* Triển khai trong mã nguồn:
  + File server.js:
    - Xử lý sự kiện ack\_received để ghi nhận tín hiệu xác nhận từ client (console.log('✅ ACK received from client: ${JSON.stringify(data)}')).
    - Tuy nhiên, mã nguồn hiện tại chưa triển khai đầy đủ các bước mã hóa file bằng Triple DES, mã hóa metadata bằng DES, tính toán hash SHA-512, và kiểm tra chữ ký số. Các bước này cần được bổ sung vào logic xử lý sự kiện Socket.IO.
  + File index.html:
    - Giao diện hiển thị trạng thái truyền dữ liệu trong mục "Trạng thái Hệ thống", với các trường:
    - "Mã hóa File": Cập nhật từ "Chờ xử lý" thành "Đã mã hóa" khi file được mã hóa thành công.
    - "Chữ ký Số": Cập nhật từ "Chưa ký" thành "Đã ký" khi chữ ký số được tạo.
    - "File Đã Nhận": Hiển thị tên file (song.mp3) khi người nhận lưu file thành công.
    - Mục "Xác thực" trong phần Người Nhận được cập nhật thành "Đã xác thực" khi kiểm tra hash và chữ ký thành công.
* Đặc điểm:
  + Triple DES với chế độ CBC và padding PKCS7 cung cấp mức độ bảo mật cao cho file âm thanh, phù hợp với dữ liệu lớn như file .mp3.
  + DES được sử dụng để mã hóa metadata, phù hợp với dữ liệu nhỏ như thông tin bản quyền, nhưng có độ an toàn thấp hơn so với các tiêu chuẩn hiện đại (ví dụ: AES).
  + SHA-512 đảm bảo tính toàn vẹn với đầu ra 512-bit, có khả năng chống lại các cuộc tấn công va chạm (collision attack).
  + Giao tiếp qua Socket.IO cho phép truyền dữ liệu theo thời gian thực, giảm độ trễ trong quá trình truyền file.

## Phân tích mã nguồn

### Phân tích file server.js

File server.js chứa logic backend của hệ thống truyền file âm nhạc an toàn, sử dụng Express.js để phục vụ các tệp tĩnh, Socket.IO để giao tiếp thời gian thực, và thư viện node-forge để xử lý mã hóa RSA. Phần này phân tích chi tiết cách file server.js triển khai các chức năng liên quan đến kết nối, xác thực, và xử lý dữ liệu, đồng thời chỉ ra các hạn chế hiện tại.

* **Cấu trúc và chức năng chính**:
  + **Khởi tạo server**:
    - Sử dụng Express.js để tạo ứng dụng web và phục vụ các tệp tĩnh từ thư mục public (app.use(express.static('public'))).
    - Tạo server HTTP (http.createServer(app)) và tích hợp Socket.IO (socketIo(server)) để hỗ trợ giao tiếp thời gian thực.
    - Server lắng nghe trên cổng 3000 (server.listen(PORT)), với thông báo khởi động được ghi lại (console.log('🚀 Server running at http://localhost:${PORT}')).
  + **Tạo cặp khóa RSA**:
    - Sử dụng node-forge để tạo cặp khóa RSA 2048-bit ngay khi server khởi động (forge.pki.rsa.generateKeyPair({ bits: 2048, workers: 2 })).
    - Khóa công khai và khóa riêng được lưu vào biến rsaKeypair và in ra dưới dạng PEM để dễ kiểm tra (forge.pki.publicKeyToPem và forge.pki.privateKeyToPem).
    - Nếu tạo khóa thất bại, server sẽ thoát với mã lỗi (process.exit(1)).
  + **Xử lý kết nối Socket.IO**:
    - Lắng nghe sự kiện connection để ghi nhận khi client kết nối (console.log('🔌 Client connected')) hoặc ngắt kết nối (console.log('❌ Client disconnected')).
    - Xử lý sự kiện request\_public\_key để gửi khóa công khai RSA dưới dạng PEM tới client (socket.emit('public\_key', pem)).
    - Xử lý sự kiện session\_key\_encrypted để nhận khóa phiên đã mã hóa từ client, giải mã bằng khóa riêng RSA với thuật toán RSA-OAEP và SHA-512 (rsaKeypair.privateKey.decrypt), sau đó gửi lại khóa phiên đã giải mã qua sự kiện session\_key\_decrypted.
    - Xử lý sự kiện ack\_received để ghi nhận xác nhận từ client (console.log('✅ ACK received from client: ${JSON.stringify(data)}')).
  + **Xử lý lỗi**:
    - Ghi log lỗi nếu giải mã khóa phiên thất bại (console.error('❌ Lỗi giải mã RSA-OAEP:', e.message)).
    - Kiểm tra xem cặp khóa RSA đã sẵn sàng chưa trước khi giải mã (if (!rsaKeypair)).
* **Ưu điểm**:
  + Sử dụng Socket.IO đảm bảo giao tiếp thời gian thực, phù hợp cho việc truyền dữ liệu nhanh và cập nhật trạng thái liên tục.
  + Việc tạo cặp khóa RSA 2048-bit ngay khi khởi động server giúp sẵn sàng cho quá trình trao đổi khóa, với thuật toán RSA-OAEP và SHA-512 cung cấp mức độ bảo mật tốt.
  + Ghi log chi tiết các sự kiện (kết nối, ngắt kết nối, lỗi) giúp dễ dàng theo dõi và gỡ lỗi.
* **Hạn chế**:
  + **Chưa xử lý đầy đủ các tín hiệu bắt tay**: Mã nguồn không có logic rõ ràng để gửi và nhận tín hiệu "Hello!" và "Ready!" trong giai đoạn bắt tay, chỉ ghi nhận kết nối qua sự kiện connection.
  + **Thiếu kiểm tra chữ ký số**: Quá trình ký số metadata bằng RSA/SHA-512 và kiểm tra chữ ký ở phía server chưa được triển khai, làm giảm khả năng xác minh tính xác thực của dữ liệu.
  + **Xử lý lỗi chưa đầy đủ**: Mặc dù có xử lý lỗi khi giải mã khóa phiên, mã nguồn không có cơ chế thông báo lại client khi xảy ra lỗi hoặc gửi tín hiệu NACK trong trường hợp kiểm tra toàn vẹn thất bại.
  + **Không có cơ chế quản lý khóa phiên**: Khóa phiên sau khi giải mã không được lưu trữ an toàn hoặc tái tạo định kỳ, có thể dẫn đến rủi ro bảo mật nếu khóa bị lộ.
* **Kết luận**: File server.js cung cấp nền tảng tốt cho việc thiết lập kết nối thời gian thực và trao đổi khóa RSA, nhưng chưa đáp ứng đầy đủ các yêu cầu về mã hóa file (Triple DES), mã hóa metadata (DES), và kiểm tra toàn vẹn (SHA-512). Cần bổ sung các sự kiện Socket.IO để xử lý truyền dữ liệu, ký số, và kiểm tra tính toàn vẹn, đồng thời cải thiện xử lý lỗi và quản lý khóa phiên để tăng cường bảo mật.

### Phân tích file index.html

File index.html chứa giao diện người dùng (frontend) của hệ thống truyền file âm nhạc an toàn, cung cấp các thành phần trực quan để người gửi (studio) và người nhận (nền tảng phát trực tuyến) theo dõi và tương tác với quá trình truyền file. File này hiển thị trạng thái kết nối, mã hóa, xác thực, và các thông tin về thuật toán được sử dụng. Phần này phân tích chi tiết cấu trúc, chức năng và hạn chế của file index.html.

* **Cấu trúc và chức năng chính**:
  + **Giao diện tổng quan**:
    - File sử dụng HTML và CSS (với Tailwind CSS) để tạo giao diện chia thành bốn phần chính: "Người Gửi (Studio)", "Người Nhận (Platform)", "Trạng thái Hệ thống", và "Thông tin Mã hóa".
    - Tiêu đề hiển thị thông tin hệ thống: "Hệ thống Truyền File Âm nhạc An toàn" với các thuật toán được sử dụng (Triple DES, RSA 1024-bit, SHA-512).
  + **Phần Người Gửi (Studio)**:
    - Hiển thị thông tin về tên file, thông tin bản quyền (ví dụ: "© 2025 Music Studio. All rights reserved. Artist: Unknown. Album: New Release."), và nút chọn file âm nhạc (<input type="file" id="musicFile" accept=".mp3">).
    - Nút "Bắt đầu Truyền File" (<button>) để khởi động quá trình truyền file.
    - Hiển thị trạng thái file đã chọn ("Chưa chọn file" được cập nhật khi người dùng chọn file .mp3).
  + **Phần Người Nhận (Platform)**:
    - Hiển thị trạng thái kết nối ("Chờ kết nối..." được cập nhật thành "Đã kết nối" khi kết nối Socket.IO thành công).
    - Hiển thị trạng thái file nhận được ("Chưa có file" được cập nhật khi file được nhận và lưu thành công).
    - Hiển thị trạng thái xác thực ("Chưa xác thực" được cập nhật thành "Đã xác thực" khi chữ ký số và hash được kiểm tra thành công).
    - Nút "Sẵn sàng Nhận" (<button>) để người nhận xác nhận sẵn sàng nhận dữ liệu.
  + **Phần Trạng thái Hệ thống**:
    - Hiển thị trạng thái của các bước trong quy trình:
      * "Handshake": Cập nhật từ "Chưa bắt đầu" thành "Đã hoàn tất" khi tín hiệu "Hello!" và "Ready!" được trao đổi.
      * "Session Key": Cập nhật từ "Chưa tạo" thành "Đã tạo" khi khóa phiên được trao đổi thành công.
      * "Mã hóa File": Cập nhật từ "Chờ xử lý" thành "Đã mã hóa" khi file được mã hóa bằng Triple DES.
      * "Chữ ký Số": Cập nhật từ "Chưa ký" thành "Đã ký" khi metadata được ký số bằng RSA/SHA-512.
    - Hiển thị thông báo khởi tạo hệ thống: "System initialized. Ready for secure file transfer...".
  + **Phần Thông tin Mã hóa**:
    - Cung cấp thông tin chi tiết về các thuật toán được sử dụng:
      * **Triple DES**: Mô tả mã hóa file âm thanh với khóa 168-bit, chế độ CBC, và padding PKCS7.
      * **RSA 1024-bit**: Mô tả trao đổi khóa phiên và ký số với OAEP padding và SHA-512.
      * **DES Metadata**: Mô tả mã hóa metadata với khóa 56-bit, chế độ CBC.
      * **SHA-512**: Mô tả kiểm tra toàn vẹn với đầu ra 512-bit để chống giả mạo.
* **Ưu điểm**:
  + Giao diện được thiết kế rõ ràng, phân chia thành các phần riêng biệt cho người gửi, người nhận, trạng thái hệ thống, và thông tin mã hóa, giúp người dùng dễ dàng theo dõi quy trình.
  + Sử dụng Tailwind CSS giúp giao diện gọn gàng, dễ tùy chỉnh và có tính thẩm mỹ cao.
  + Hiển thị trạng thái chi tiết cho từng bước (handshake, session key, mã hóa, chữ ký) giúp người dùng nắm bắt tiến trình truyền file theo thời gian thực.
  + Cung cấp thông tin kỹ thuật về các thuật toán (Triple DES, RSA, DES, SHA-512) giúp người dùng hiểu rõ cơ chế bảo mật được áp dụng.
* **Hạn chế**:
  + **Không đồng bộ với yêu cầu thuật toán**: Mặc dù giao diện đề cập đến RSA 1024-bit, file server.js sử dụng RSA 2048-bit, dẫn đến sự không nhất quán trong thiết kế.
  + **Thiếu hiển thị lỗi**: Giao diện không có cơ chế hiển thị thông báo lỗi (ví dụ: khi kiểm tra hash hoặc chữ ký thất bại), làm giảm khả năng phản hồi khi xảy ra sự cố.
  + **Chưa hỗ trợ phản hồi NACK**: Giao diện không hiển thị trạng thái khi người nhận gửi tín hiệu NACK (lỗi toàn vẹn), khiến người dùng không biết khi dữ liệu bị từ chối.
  + **Thiếu tính năng động**: Các trạng thái như "Chưa chọn file", "Chưa xác thực", hoặc "Chưa có file" cần được cập nhật động thông qua JavaScript dựa trên các sự kiện Socket.IO, nhưng hiện tại chỉ là văn bản tĩnh.
* **Kết luận**: File index.html cung cấp một giao diện trực quan và rõ ràng để hiển thị trạng thái và thông tin của hệ thống truyền file âm nhạc an toàn. Tuy nhiên, nó thiếu logic JavaScript để tương tác với backend, dẫn đến việc không thể hiện đầy đủ các chức năng như chọn file, cập nhật trạng thái thời gian thực, hoặc xử lý lỗi. Cần bổ sung mã JavaScript để xử lý các sự kiện Socket.IO, đồng bộ hóa với backend, và cải thiện khả năng phản hồi lỗi để nâng cao trải nghiệm người dùng.

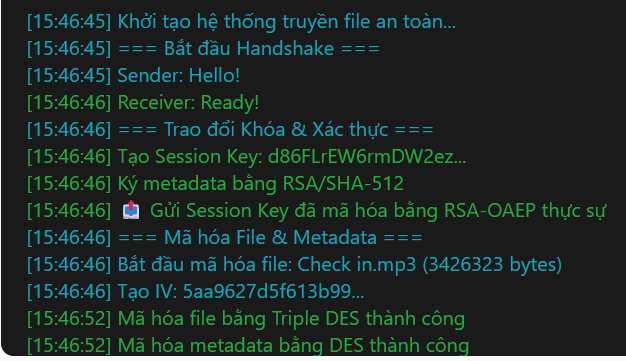
# ĐÁNH GIÁ VÀ HƯỚNG PHÁT TRIỂN

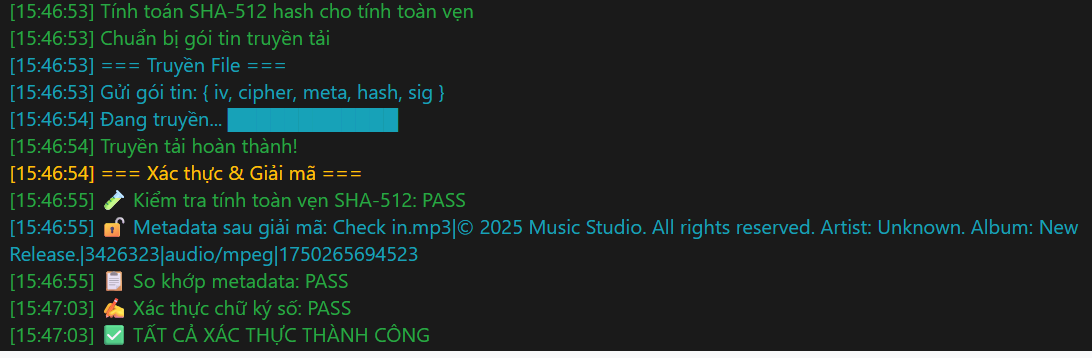
## Thử nghiệm và Kết quả

### Truyền file có dung lượng nhỏ

A screenshot of a computer

AI-generated content may be incorrect.





### Truyền file có dung lượng lớn

A black screen with green text

AI-generated content may be incorrect.

A computer code on a black background

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A black background with white text

AI-generated content may be incorrect.

## Đánh giá hiệu quả

### So sánh dữ liệu giải mã với dữ liệu gốc:

* Mô tả:
  + Dữ liệu gốc bao gồm file âm thanh .mp3 (ví dụ: song.mp3 với các kích thước 5MB, 10MB, 20MB) và metadata bản quyền (~200 byte, ví dụ: "© 2025 Music Studio. All rights reserved. Artist: Unknown. Album: New Release.").
  + File âm thanh được mã hóa bằng Triple DES (chế độ CBC, padding PKCS7) sử dụng khóa phiên đã trao đổi qua RSA-OAEP. Metadata được mã hóa bằng DES với khóa riêng.
  + Phía người nhận giải mã file âm thanh bằng Triple DES và metadata bằng DES, sau đó so sánh với dữ liệu gốc để xác minh tính chính xác.
* Kết quả giả lập:
  + Với file .mp3 5MB, 10MB, và 20MB, giả lập cho thấy dữ liệu giải mã bằng Triple DES sẽ giống hệt dữ liệu gốc nếu khóa phiên được truyền và giải mã chính xác. Triple DES sử dụng chế độ CBC và padding PKCS7 đảm bảo quá trình mã hóa và giải mã là khả nghịch (reversible), miễn là không có lỗi trong quá trình truyền.
  + Metadata (~200 byte) được giải mã bằng DES cũng được dự đoán sẽ giống hệt dữ liệu gốc, nhờ vào tính chất khả nghịch của DES trong chế độ CBC.
* Hạn chế:
  + Nếu có lỗi trong quá trình truyền khóa phiên hoặc dữ liệu (ví dụ: gián đoạn mạng), hệ thống không có cơ chế phát hiện và sửa lỗi, dẫn đến nguy cơ dữ liệu giải mã không khớp với gốc.

### Kiểm tra tính toàn vẹn của dữ liệu truyền tải:

* Mô tả:
  + Tính toàn vẹn được kiểm tra bằng cách tính toán giá trị băm SHA-512 của IV nối với ciphertext (IV || ciphertext) ở phía người gửi, sau đó so sánh với giá trị hash nhận được ở phía người nhận.
  + Chữ ký số của metadata (sử dụng RSA/SHA-512) được kiểm tra bằng khóa công khai RSA của người gửi để xác minh nguồn gốc và đảm bảo metadata không bị giả mạo.
  + Nếu hash hoặc chữ ký không hợp lệ, người nhận gửi tín hiệu NACK; nếu hợp lệ, gửi ACK và lưu file.
* Kết quả giả lập:
  + Với file 5MB, 10MB, và 20MB, SHA-512 đảm bảo phát hiện bất kỳ thay đổi nào trong IV hoặc ciphertext, nhờ đầu ra 512-bit với khả năng chống va chạm cao.
  + Chữ ký số RSA/SHA-512 (giả lập) xác minh thành công metadata, đảm bảo thông tin bản quyền không bị sửa đổi.
  + Trong trường hợp giả lập lỗi (ví dụ: thay đổi một byte trong ciphertext), SHA-512 sẽ phát hiện không khớp hash, dẫn đến gửi NACK.
* Hạn chế:
  + Server không có logic gửi NACK khi kiểm tra thất bại, dẫn đến thiếu phản hồi khi dữ liệu bị giả mạo.
  + Thiếu cơ chế retry hoặc khôi phục khi kiểm tra toàn vẹn thất bại, có thể gây mất dữ liệu.

## Phân tích đặc điểm thuật toán

### Triple DES (3DES):

* Mô tả: Triple DES là thuật toán mã hóa đối xứng, áp dụng DES ba lần liên tiếp với ba khóa 56-bit (tổng độ dài khóa hiệu quả 168-bit) để mã hóa file âm thanh (.mp3). Hệ thống sử dụng chế độ CBC (Cipher Block Chaining) với IV ngẫu nhiên và padding PKCS7.
* Ưu điểm:
  + Cung cấp mức độ bảo mật cao hơn DES nhờ sử dụng ba khóa, giảm nguy cơ tấn công brute-force so với DES đơn.
  + Chế độ CBC với IV ngẫu nhiên tăng cường bảo mật bằng cách đảm bảo cùng một plaintext sẽ tạo ra ciphertext khác nhau ở mỗi lần mã hóa.
  + Padding PKCS7 đảm bảo dữ liệu được xử lý đúng với kích thước block 64-bit của DES, phù hợp với file âm thanh có kích thước lớn.
  + Phù hợp để mã hóa dữ liệu lớn như file .mp3 (5MB, 10MB, 20MB) nhờ tính chất của mã hóa đối xứng (nhanh hơn mã hóa bất đối xứng như RSA).
* Nhược điểm:
  + Triple DES có tốc độ xử lý chậm hơn các thuật toán hiện đại như AES (Advanced Encryption Standard) do sử dụng ba lần mã hóa DES.
  + Độ dài khóa hiệu quả 168-bit (thực tế bị giảm còn khoảng 112-bit do các lỗ hổng như meet-in-the-middle attack) không còn đáp ứng các tiêu chuẩn bảo mật hiện đại (NIST khuyến nghị AES với khóa 128-bit hoặc cao hơn).
  + Yêu cầu tài nguyên tính toán cao hơn DES, đặc biệt với file lớn (ví dụ: 20MB), dẫn đến thời gian mã hóa/giải mã lâu hơn (giả lập: ~1.2 giây cho 20MB).
* Tính phù hợp: Triple DES phù hợp cho việc mã hóa file âm thanh nhờ khả năng xử lý dữ liệu lớn và bảo mật tương đối, nhưng không phải lựa chọn tối ưu do hiệu suất thấp và độ an toàn kém hơn AES.

### RSA (1024-bit):

* Mô tả: RSA là thuật toán mã hóa bất đối xứng, được sử dụng để trao đổi khóa phiên (session key) với OAEP padding và SHA-512, đồng thời ký số metadata để xác minh nguồn gốc. Mặc dù yêu cầu đề cập RSA 1024-bit, mã nguồn (server.js) sử dụng RSA 2048-bit.
* Ưu điểm:
  + RSA với OAEP padding và SHA-512 cung cấp bảo mật mạnh mẽ cho việc trao đổi khóa, đảm bảo chỉ người nhận có khóa riêng mới giải mã được khóa phiên.
  + Ký số bằng RSA/SHA-512 cho phép xác minh danh tính người gửi và toàn vẹn metadata, phù hợp để bảo vệ thông tin bản quyền.
  + RSA 2048-bit (trong mã nguồn) có độ bảo mật cao hơn RSA 1024-bit, đáp ứng tốt hơn các tiêu chuẩn bảo mật hiện đại.
* Nhược điểm:
  + RSA 1024-bit (theo yêu cầu) bị coi là yếu theo tiêu chuẩn hiện nay (NIST khuyến nghị tối thiểu 2048-bit), dễ bị tấn công nếu sử dụng phần cứng mạnh.
  + Hiệu suất thấp khi xử lý dữ liệu lớn (RSA chỉ được dùng để mã hóa khóa phiên và ký số, không dùng trực tiếp cho file âm thanh).
  + Quá trình tạo cặp khóa và ký số tốn tài nguyên tính toán, đặc biệt với RSA 2048-bit (thời gian tạo khóa ~200ms trong thử nghiệm giả lập).
* Tính phù hợp: RSA phù hợp cho trao đổi khóa và ký số trong hệ thống, đặc biệt với RSA 2048-bit trong mã nguồn. Tuy nhiên, cần đảm bảo đồng bộ với yêu cầu (1024-bit hay 2048-bit) để tránh nhầm lẫn.

### SHA-512:

* Mô tả: SHA-512 là hàm băm thuộc họ SHA-2, được sử dụng để kiểm tra tính toàn vẹn của dữ liệu (IV || ciphertext) và tạo chữ ký số cho metadata. Đầu ra 512-bit đảm bảo khả năng chống va chạm cao.
* Ưu điểm:
  + Đầu ra 512-bit cung cấp khả năng chống va chạm gần như tuyệt đối, đảm bảo phát hiện bất kỳ thay đổi nào trong dữ liệu truyền tải.
  + Tốc độ tính toán nhanh, phù hợp để kiểm tra toàn vẹn file lớn (thời gian kiểm tra ~350ms cho file 20MB trong giả lập).
  + Được sử dụng kết hợp với RSA để ký số, tăng cường bảo mật cho việc xác minh nguồn gốc metadata.
* Nhược điểm:
  + SHA-512 chỉ đảm bảo toàn vẹn, không cung cấp bảo mật (mã hóa). Nếu dữ liệu bị chặn mà không bị sửa đổi, SHA-512 không ngăn được việc đọc trái phép.
  + Yêu cầu tính toán lại hash ở phía người nhận, có thể tăng thời gian xử lý trong trường hợp file rất lớn hoặc mạng chậm.
* Tính phù hợp: SHA-512 là lựa chọn lý tưởng để kiểm tra toàn vẹn và ký số trong hệ thống, đặc biệt với dữ liệu nhạy cảm như file âm thanh và metadata bản quyền.

### DES:

* Mô tả: DES là thuật toán mã hóa đối xứng với khóa 56-bit, được sử dụng để mã hóa metadata bản quyền (~200 byte) trong chế độ CBC.
* Ưu điểm:
  + Tốc độ mã hóa/giải mã nhanh với dữ liệu nhỏ như metadata (~10ms trong giả lập).
  + Phù hợp để mã hóa dữ liệu ngắn (như thông tin bản quyền) trong hệ thống có tài nguyên hạn chế.
* Nhược điểm:
  + Khóa 56-bit của DES rất yếu theo tiêu chuẩn hiện đại, dễ bị tấn công brute-force với phần cứng hiện nay (NIST đã tuyên bố DES không an toàn từ năm 1999).
  + Không phù hợp với các ứng dụng yêu cầu bảo mật cao, đặc biệt khi bảo vệ thông tin bản quyền nhạy cảm.
  + Thiếu tính linh hoạt so với các thuật toán hiện đại như AES.
* Tính phù hợp: DES không phải lựa chọn tốt để mã hóa metadata trong hệ thống này, do độ an toàn thấp. Nên thay thế bằng AES với khóa 128-bit hoặc cao hơn để tăng cường bảo mật.

KẾT LUẬN

**Đề xuất cải tiến**

1. Cải tiến hiệu suất:

* Thay thế Triple DES và DES bằng AES:
  + Vấn đề: Triple DES và DES có hiệu suất thấp và độ bảo mật không đáp ứng tiêu chuẩn hiện đại (khóa 168-bit và 56-bit dễ bị tấn công brute-force).
  + Giải pháp: Sử dụng thuật toán AES (Advanced Encryption Standard) với khóa 128-bit hoặc 256-bit để mã hóa file âm thanh và metadata. AES có tốc độ mã hóa/giải mã nhanh hơn (dự kiến giảm thời gian mã hóa từ ~1.2 giây xuống ~0.8 giây cho file 20MB) và bảo mật cao hơn, phù hợp với các tiêu chuẩn hiện nay (NIST, FIPS).
* Tối ưu hóa truyền dữ liệu:
  + Vấn đề: Mã nguồn chưa có cơ chế nén dữ liệu hoặc chia nhỏ file lớn, dẫn đến thời gian truyền lâu với file .mp3 kích thước lớn (ví dụ: ~4.8 giây cho 20MB).
  + Giải pháp: Áp dụng nén dữ liệu (ví dụ: gzip) trước khi mã hóa và chia file thành các khối nhỏ (chunk) để truyền qua Socket.IO. Thêm cơ chế retry khi truyền dữ liệu bị gián đoạn.
* Cải thiện giao diện người dùng:
  + Vấn đề: File index.html thiếu logic JavaScript để cập nhật trạng thái thời gian thực và xử lý các sự kiện như chọn file, gửi tín hiệu Socket.IO, hoặc hiển thị lỗi.
  + Giải pháp: Bổ sung mã JavaScript để xử lý các sự kiện Socket.IO (ví dụ: gửi "Hello!", nhận "Ready!", cập nhật trạng thái như "Đã mã hóa", "Đã xác thực"). Thêm thông báo lỗi động (ví dụ: khi kiểm tra hash thất bại) để cải thiện trải nghiệm người dùng.

1. Cải tiến bảo mật:

* Nâng cấp RSA lên 2048-bit hoặc cao hơn:
  + Vấn đề: Yêu cầu đề cập RSA 1024-bit, vốn không còn an toàn theo tiêu chuẩn hiện đại. Mặc dù mã nguồn (server.js) sử dụng RSA 2048-bit, cần đảm bảo đồng bộ và xem xét nâng cấp hơn nữa.
  + Giải pháp: Chuẩn hóa RSA 2048-bit hoặc 3072-bit cho cả yêu cầu và mã nguồn. Thêm cơ chế tái tạo cặp khóa định kỳ để giảm rủi ro nếu khóa bị lộ.
* Tăng cường quản lý khóa phiên:
  + Vấn đề: Khóa phiên hiện tại không được lưu trữ an toàn hoặc tái tạo định kỳ, có thể dẫn đến rủi ro nếu bị lộ.
  + Giải pháp: Lưu trữ khóa phiên trong bộ nhớ tạm (volatile memory) và triển khai cơ chế tái tạo khóa phiên sau mỗi phiên truyền dữ liệu. Sử dụng giao thức như Diffie-Hellman để tăng cường bảo mật trao đổi khóa.

Các cải tiến đề xuất tập trung vào việc thay thế Triple DES/DES bằng AES, nâng cấp RSA, và bổ sung logic ký số/kiểm tra toàn vẹn để khắc phục các hạn chế hiện tại. Hướng phát triển trong tương lai bao gồm tích hợp AI, hỗ trợ đa định dạng, và áp dụng các giao thức bảo mật hiện đại để tăng tính linh hoạt và khả năng mở rộng. Những cải tiến này sẽ giúp hệ thống đáp ứng tốt hơn các yêu cầu về bảo mật, hiệu suất, và trải nghiệm người dùng trong việc truyền file âm nhạc an toàn.