**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC ĐẠI NAM**

Logo, company name

Description automatically generated

**BÀI TẬP LỚN**

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

**ĐỀ TÀI: GỬI BÀI TẬP CHIA THÀNH NHIỀU PHẦN**

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

**Sinh viên thực hiện:**  **Nhóm 1**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Mã SV** | **Họ và tên** | **Lớp** |
| **1** | **1771020712** | **Nguyễn Mạnh Tuân** | **CNTT 17-07** |
| **2** | **1771020377** | **Hoàng Công Khanh** | **CNTT 17-07** |
| **3** | **1771020070** | **Trần Anh Bằng** | **CNTT 17-07** |

**Hà Nội, năm 2025**

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC ĐẠI NAM**

Logo, company name

Description automatically generated

**BÀI TẬP LỚN**

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

**ĐỀ TÀI: GỬI BÀI TẬP CHIA THÀNH NHIỀU PHẦN**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **STT** | **Mã Sinh Viên** | **Họ và Tên** | **Ngày Sinh** | **Điểm** | |
| **Bằng Số** | **Bằng Chữ** |
| 1 | 1771020712 | Nguyễn Mạnh Tuân | 24/01/2005 |  |  |
| 2 | 1771020377 | Hoàng Công Khanh | 08/11/2005 |  |  |
| 3 | 1771020070 | Trần Anh Bằng | 03/01/2005 |  |  |

|  |  |
| --- | --- |
| **CÁN BỘ CHẤM THI 1** | **CÁN BỘ CHẤM THI 2** |

**Hà Nội, năm 2025**

MỤC LỤC

**[CHƯƠNG 1. ĐặT VẤN ĐỀ VÀ PHÂN TÍCH 1](#_Toc14894)**

**[1.1 Mã hóa dữ liệu 1](#_Toc31871)**

**[1.2 Xác thực người dùng 2](#_Toc24481)**

**[1.3 kiểm tra tính toàn vẹn 4](#_Toc1389)**

**[CHƯƠNG 2. Mô tả thuật toán 6](#_Toc32290)**

**[2.1 Bắt tay (handshake) đơn giản 6](#_Toc3752)**

**[2.2 Xác thực ( ký số và trao đổi khóa) 7](#_Toc9238)**

**[2.3 Truyền dữ liệu và kiểm tra toàn vẹn 8](#_Toc27503)**

**[CHƯƠNG 3. Phân tích mã nguồn 9](#_Toc12005)**

**[3.1 Kiến trúc hệ thống tổng thể ( client - server ) 9](#_Toc5250)**

**[3.2 Phân tích chi tiết các module 11](#_Toc11774)**

**[3.3 Đánh giá cách tổ chức node 13](#_Toc27781)**

**[CHƯƠNG 4. Thử nghiệm 16](#_Toc4656)**

**[4.1 Thử nghiệm với các loại file 16](#_Toc14047)**

**[4.2 Ghi lại kết quả 16](#_Toc7023)**

**[4.3 Ghi nhận lỗi 17](#_Toc31448)**

**[CHƯƠNG 5. Đánh giá hiệu quả 19](#_Toc12892)**

**[5.1. So sánh dữ liệu sau khi giải mã với dữ liệu gốc 19](#_Toc23813)**

**[5.2. Kiểm tra tính toàn vẹn của dữ liệu 19](#_Toc16943)**

**[5.3. Nhận xét về hiệu quả truyền tải, độ tin cậy của hệ thống 20](#_Toc29551)**

**[CHƯƠNG 6. Phân tích , nhận xét các thuật toán 22](#_Toc19686)**

**[6.1. Thuật toán mã hóa sử dụng 22](#_Toc2697)**

**[6.2. Thuật toán xác thực / chữ ký số 25](#_Toc32383)**

**[6.3. Thuật toán kiểm tra tính toàn vẹn 27](#_Toc23124)**

**[CHƯƠNG 7. đề xuất cải tiến và hướng phát triển 29](#_Toc18789)**

**[7.1 Đề xuất cải tiến 29](#_Toc27932)**

**[7.2 Hướng phát triển 30](#_Toc26497)**

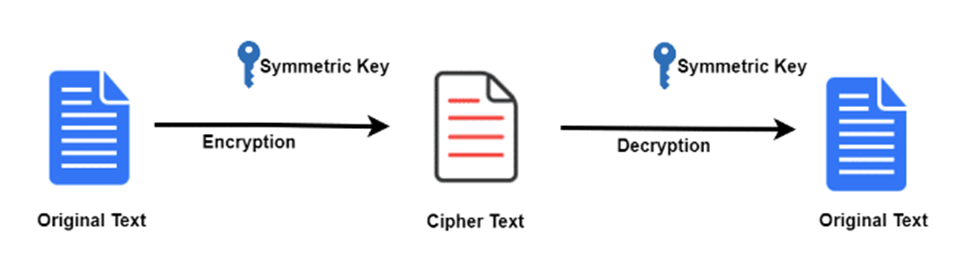
**[KẾT LUẬN 32](#_Toc29087)**

**[Tài liệu tham khảo 34](#_Toc29425)**

# ĐặT VẤN ĐỀ VÀ PHÂN TÍCH

## Mã hóa dữ liệu

Trong môi trường mạng, dữ liệu khi truyền tải rất dễ bị tấn công, nghe lén hoặc sửa đổi bởi bên thứ ba. Để bảo vệ nội dung của bài tập (có thể là file văn bản, âm thanh, hình ảnh, v.v.), **việc mã hoá dữ liệu trước khi gửi đi là cực kỳ quan trọng**.



***1.1.1. Mục tiêu của mã hoá***

* Đảm bảo **tính bí mật (confidentiality)**: Người không được phép không thể đọc được nội dung dữ liệu.
* Tránh rò rỉ thông tin quan trọng trong quá trình gửi bài tập qua mạng.
* Ngăn chặn các hành vi tấn công như **Man-in-the-Middle (MITM)** hoặc **packet sniffing**.

***1.1.2. Giải pháp sử dụng***

Chúng tôi áp dụng mô hình mã hoá lai (Hybrid Encryption), bao gồm:

* **Mã hoá đối xứng (symmetric encryption)** để mã hoá nội dung dữ liệu (ví dụ: AES 128-bit hoặc 256-bit).
* **Mã hoá bất đối xứng (asymmetric encryption)** để mã hoá khoá bí mật (AES key) dùng trong mã hoá đối xứng (ví dụ: RSA 2048-bit).

***1.1.3. Quy trình mã hoá tổng quát***

**Người gửi**:

* Tạo một khoá đối xứng (AES) ngẫu nhiên.
* Dùng khoá AES để mã hoá file bài tập.
* Dùng khoá công khai của người nhận (public key) để mã hoá khoá AES.
* Gửi cả file đã mã hoá và khoá AES đã mã hoá tới người nhận.

**Người nhận**:

* Dùng khoá riêng (private key) để giải mã khoá AES.
* Dùng khoá AES để giải mã file bài tập.

***1.1.4. Lý do lựa chọn kết hợp AES + RSA***

* AES mã hoá nhanh, hiệu quả với dữ liệu lớn (file .mp3, .txt, .pdf...).
* RSA đảm bảo an toàn khi trao đổi khoá AES qua mạng.
* Kết hợp cả hai giúp đảm bảo **tính bảo mật và hiệu suất xử lý**.

## Xác thực người dùng

Trong hệ thống truyền dữ liệu nhạy cảm như gửi bài tập giữa người gửi và người nhận, **xác thực người dùng là bước thiết yếu** nhằm đảm bảo rằng:

* Chỉ người gửi hợp lệ mới được gửi dữ liệu.
* Chỉ người nhận hợp lệ mới có thể nhận và giải mã dữ liệu.

***1.2.1. Mục tiêu của xác thực người dùng***

* Đảm bảo **tính xác thực (authentication)** của cả hai bên trong quá trình trao đổi.
* Ngăn chặn các hành vi giả mạo danh tính.
* Kết hợp với mã hoá để hình thành **một kênh truyền an toàn và đáng tin cậy**.

***1.2.2. Các cơ chế xác thực áp dụng***

**Có thể áp dụng một hoặc nhiều phương pháp sau:**

**a. Xác thực bằng tài khoản và mật khẩu (cơ bản)**

* Người dùng phải đăng nhập bằng tên đăng nhập và mật khẩu trước khi gửi/nhận file.
* Mật khẩu có thể được băm (hash) bằng SHA-256 để bảo mật.

**b. Xác thực bằng chữ ký số (digital signature)**

* Người gửi ký dữ liệu bằng **chữ ký số** tạo ra từ khóa riêng (private key).
* Người nhận dùng **khóa công khai** của người gửi để xác thực chữ ký.
* Nếu xác thực thành công → xác định người gửi là hợp lệ, dữ liệu không bị chỉnh sửa.

***1.2.3. Quy trình xác thực đề xuất (kết hợp login + chữ ký số)***

**Khi gửi bài:**

* Người gửi đăng nhập vào hệ thống (username, password).
* Hệ thống xác thực tài khoản.
* Dữ liệu sau đó được **ký số bằng khóa riêng** của người gửi.

**Khi nhận bài:**

* Người nhận kiểm tra **chữ ký số** đi kèm dữ liệu.
* Nếu chữ ký hợp lệ → xác thực thành công người gửi.
* Nếu không hợp lệ → từ chối nhận dữ liệu.

***1.2.4. Lợi ích khi kết hợp nhiều cơ chế xác thực***

* Đảm bảo **đa lớp bảo mật (multi-layer security)**.
* Tăng độ tin cậy trong trao đổi dữ liệu.
* Giảm rủi ro rò rỉ thông tin hoặc bị mạo danh.

## kiểm tra tính toàn vẹn

Trong quá trình truyền tải file âm thanh (ví dụ: .mp3, .wav) qua mạng, dữ liệu có thể bị thay đổi, lỗi, hoặc bị bên thứ ba can thiệp. Do đó, **kiểm tra tính toàn vẹn là bước không thể thiếu để đảm bảo dữ liệu nhận được là chính xác và không bị sửa đổi**.

***1.3.1. Mục tiêu***

* Đảm bảo rằng **dữ liệu nhận được là giống hệt với dữ liệu gốc** được gửi đi.
* Phát hiện kịp thời nếu dữ liệu bị thay đổi, cắt xén hoặc chèn thêm trong quá trình truyền.
* Ngăn chặn nguy cơ **giả mạo nội dung file âm thanh** (ví dụ: thay đổi nội dung bài giảng hoặc hướng dẫn).

***1.3.2. Giải pháp sử dụng: Thuật toán băm (Hashing)***

* Dữ liệu âm thanh được xử lý qua một **thuật toán băm mạnh** (ví dụ: SHA-256) để tạo ra một **mã băm duy nhất (hash value)**.
* Khi truyền dữ liệu, mã băm này cũng được gửi kèm theo (có thể được **ký số** để chống giả mạo).
* Người nhận **tính lại mã băm từ file âm thanh đã nhận** và so sánh với mã băm gốc:

+ Nếu khớp → dữ liệu toàn vẹn.

+ Nếu không khớp → dữ liệu đã bị thay đổi.

***1.3.3. Quy trình kiểm tra toàn vẹn***

**Người gửi:**

* Tính mã băm của file âm thanh gốc bằng SHA-256.
* Ký mã băm bằng chữ ký số (RSA private key).
* Gửi:

+ File âm thanh đã mã hoá.

+ Mã băm (hoặc chữ ký số của mã băm).

**Người nhận:**

* Giải mã file âm thanh.
* Tính lại mã băm từ file đã nhận.
* Giải mã chữ ký số (nếu có) bằng khoá công khai của người gửi.
* So sánh:

+ Nếu khớp → file nguyên vẹn.

+ Nếu khác → file đã bị thay đổi, cảnh báo lỗi.

***1.3.4. Kết luận***

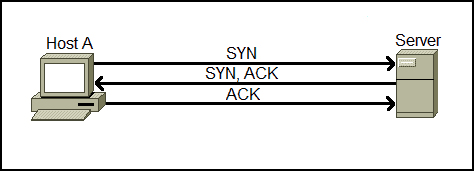
Việc **kiểm tra tính toàn vẹn** là bước quan trọng để đảm bảo:

* Dữ liệu không bị chỉnh sửa.
* Người nhận có thể **tin tưởng** nội dung file âm thanh.
* Hệ thống đạt tiêu chuẩn an toàn thông tin khi xử lý dữ liệu nhạy cảm.

# Mô tả thuật toán

## Bắt tay (handshake) đơn giản

**Mục tiêu**: Thiết lập kết nối an toàn giữa hai bên (Client và Server), đồng thời trao đổi thông tin cần thiết để mã hóa.



### ****Thuật toán sử dụng****:

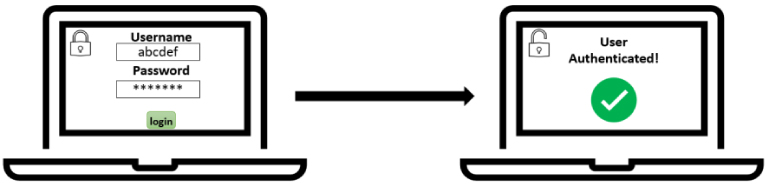
* **RSA** (thuật toán mã hóa bất đối xứng)
* **Diffie-Hellman** (nếu muốn dùng trao đổi khóa mà không gửi khóa thật sự)

### ****Quy trình****:

1. **Client gửi yêu cầu kết nối** đến Server, kèm theo phiên bản giao thức và thông tin hỗ trợ (các thuật toán mã hóa mà Client có thể dùng).
2. **Server phản hồi** bằng cách chọn thuật toán phù hợp, và gửi **public key** của mình cho Client.
3. Client tạo ra **session key (khóa phiên)** dùng cho mã hóa đối xứng (ví dụ: AES), sau đó **mã hóa khóa này bằng public key của Server** (RSA).
4. Client gửi khóa phiên đã mã hóa cho Server.
5. Server dùng private key của mình để **giải mã khóa phiên**, từ đó hai bên có chung một **session key**

## Xác thực ( ký số và trao đổi khóa)

**Mục tiêu**: Xác thực người gửi là hợp lệ, dữ liệu không bị giả mạo.



### ****Thuật toán sử dụng****:

* **RSA/ECDSA** để **ký số** (digital signature).
* **SHA-256** để băm dữ liệu trước khi ký (tăng tốc độ và độ an toàn).

### ****Quy trình****:

1. Người gửi tạo **hash** của dữ liệu bằng SHA-256.
2. Sau đó dùng **private key** để ký lên hash → tạo thành **chữ ký số**.
3. Gửi: Dữ liệu + Chữ ký số + Public key (hoặc certificate) cho người nhận.
4. Người nhận:

* Dùng SHA-256 để tính lại hash của dữ liệu.
* Dùng public key của người gửi để **giải mã chữ ký số** → lấy hash gốc.
* So sánh hai hash:

+ Nếu **trùng khớp** → dữ liệu hợp lệ, không bị thay đổi, người gửi là xác thực.

+ Nếu **không khớp** → có thể bị giả mạo hoặc dữ liệu bị thay đổi.



## Truyền dữ liệu và kiểm tra toàn vẹn

**Mục tiêu**: Truyền dữ liệu đã mã hóa, đồng thời phát hiện nếu dữ liệu bị thay đổi trong quá trình truyền.

### ****Thuật toán sử dụng****:

* **AES-128** (hoặc AES-256): Mã hóa đối xứng nhanh và mạnh.
* **SHA-256**: Tạo hash kiểm tra toàn vẹn.
* Có thể sử dụng thêm **HMAC** để tăng độ an toàn.

### ****Quy trình****:

1. Dữ liệu được **mã hóa bằng AES** với khóa phiên đã tạo trong bước bắt tay.
2. Tạo **hash (SHA-256)** của dữ liệu gốc (trước khi mã hóa) → giúp kiểm tra toàn vẹn.
3. Gửi: Dữ liệu mã hóa + hash (hoặc HMAC) đến người nhận.
4. Người nhận:

* Dùng AES để giải mã dữ liệu.
* Tính lại hash của dữ liệu giải mã.
* So sánh với hash gốc.

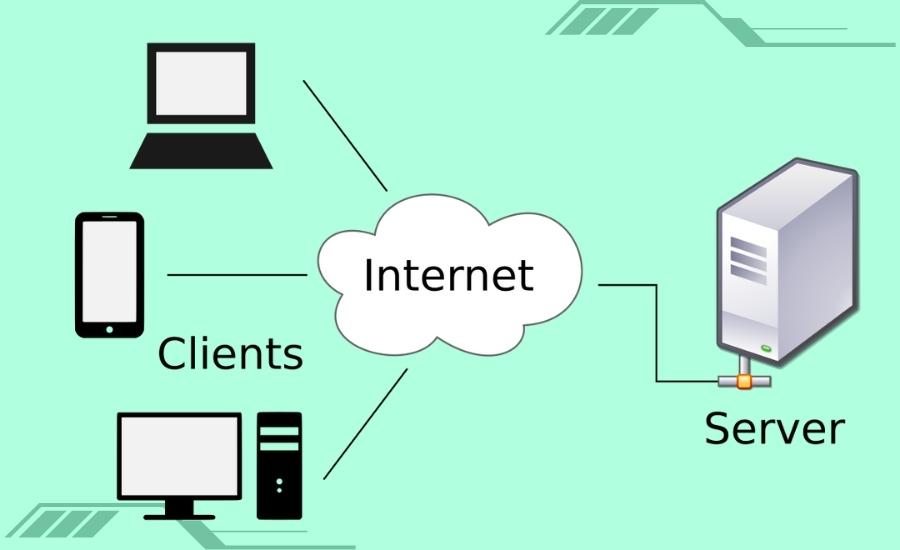
+ Nếu trùng khớp → dữ liệu **toàn vẹn**.

+ Nếu khác → dữ liệu đã bị thay đổi trong quá trình truyền.

# Phân tích mã nguồn

## Kiến trúc hệ thống tổng thể ( client - server )

Hệ thống được thiết kế theo mô hình **phân tầng ba lớp (Three-tier Architecture)** kết hợp với mô hình **Client-Server truyền thống,** nhằm đảm bảo việc chia sẻ, mã hóa, xác thực và kiểm tra toàn vẹn file bài tập diễn ra một cách an toàn, rõ ràng và dễ bảo trì. Kiến trúc này gồm 3 lớp chính:



***3.1.1. Lớp giao diện người dùng (Presentation Layer)***

Đây Đây là tầng tương tác giữa người dùng và hệ thống. Người dùng (thường là sinh viên hoặc giảng viên) sử dụng giao diện để thực hiện các chức năng:

* Đăng nhập vào hệ thống.
* Chọn file bài tập cần gửi.
* Thực hiện mã hóa dữ liệu.
* Ký số trước khi gửi bài.
* Nhận file và xác thực chữ ký người gửi.
* Giao diện được thiết kế đơn giản, dễ sử dụng, có thể triển khai trên desktop (Tkinter) hoặc web (Flask). Tính năng hiển thị tiến trình, báo lỗi, xác nhận và hướng dẫn được tích hợp để đảm bảo trải nghiệm người dùng tốt.

***3.1.2. Lớp xử lý nghiệp vụ (Business Logic Layer)***

Đây là tầng xử lý trung tâm của hệ thống, nơi thực thi các thuật toán bảo mật, mã hóa, xác thực và kiểm tra toàn vẹn dữ liệu. Bao gồm các module:

* Mã hóa – Giải mã: dùng AES cho dữ liệu và RSA cho khóa.
* Xác thực người dùng: dựa trên đăng nhập mật khẩu và chữ ký số RSA.
* Kiểm tra toàn vẹn: sử dụng SHA-256 để tạo mã băm dữ liệu.
* Quản lý khóa: lưu trữ và xử lý khóa công khai/riêng.
* Trao đổi khóa phiên: bắt tay an toàn để chia sẻ AES key giữa hai bên.

***3.1.3 Lớp dữ liệu và truyền thông (Data & Communication Layer)***

Lớp này chịu trách nhiệm truyền tải dữ liệu và lưu trữ khóa, nhật ký giao dịch và file đã mã hóa. Bao gồm:

* Lưu trữ khóa: Các khóa RSA được lưu dưới dạng file .pem.
* Lưu trữ dữ liệu mã hóa: File mã hóa được lưu đệm để gửi đi.
* Trao đổi dữ liệu qua mạng: có thể sử dụng HTTP, socket hoặc email kèm file.

Lớp này cũng chịu trách nhiệm xử lý các tình huống lỗi mạng, mất dữ liệu, xác minh tính đúng đắn của file sau khi truyền.

***3.1.4 Luồng dữ liệu trong hệ thống***

a. Phía người gửi:

* Chọn file bài tập cần gửi.
* Hệ thống tạo khóa AES ngẫu nhiên.
* Dữ liệu được mã hóa bằng AES.
* Khóa AES được mã hóa bằng RSA (public key của người nhận).
* Mã băm file được tạo bằng SHA-256.
* Mã băm được ký bằng chữ ký số (private key).
* Gửi: file mã hóa + khóa AES đã mã hóa + chữ ký số + mã băm.

b. Phía người nhận:

* Nhận gói dữ liệu từ người gửi.
* Dùng private key để giải mã khóa AES.
* Dùng AES để giải mã file.
* Kiểm tra mã băm bằng cách tính lại và đối chiếu.
* Dùng public key người gửi để xác thực chữ ký số.
* Nếu hợp lệ → file đáng tin cậy. Nếu không → cảnh báo giả mạo/chỉnh sửa.

***3.1.5 Ưu điểm kiến trúc***

* Phân tầng rõ ràng: giúp mở rộng hệ thống dễ dàng.
* Tách biệt nhiệm vụ: mỗi lớp chịu trách nhiệm một phần riêng biệt.
* Đảm bảo bảo mật toàn diện: nhờ sự kết hợp giữa AES, RSA, SHA-256, và chữ ký số.
* Dễ tích hợp và nâng cấp: có thể thay thế thuật toán, giao diện mà không ảnh hưởng đến toàn bộ hệ thống.

***3.1.6 Nhược điểm và hướng cải thiện***

* RSA chậm với file lớn: đã khắc phục bằng cách dùng RSA chỉ để mã hóa khóa AES.
* Không có SSL/TLS: có thể tích hợp thêm để tăng bảo mật kênh truyền.
* Không có cơ chế phát hiện replay-attack: nên thêm timestamp hoặc nonce để xử lý.

## Phân tích chi tiết các module

Để đảm bảo hệ thống bảo mật đạt yêu cầu về tính **bảo mật (confidentiality), xác thực (authentication), toàn vẹn (integrity)** và **không chối bỏ (non-repudiation),** nhóm đã xây dựng các module chức năng chính như sau:

***3.2.1.Module mã hoá***

* Tạo khóa AES ngẫu nhiên 128-bit cho mỗi lần gửi file.
* Dùng AES ở chế độ CBC (Cipher Block Chaining) để mã hóa nội dung bài tập.
* Tạo IV (initialization vector) ngẫu nhiên và lưu kèm theo dữ liệu mã hóa.
* Sử dụng RSA để mã hóa khóa AES bằng public key của người nhận.

**Quy trình thực thi:**

* Người dùng chọn file bài tập → chương trình tạo khóa AES và IV.
* File được mã hóa thành file .enc.
* Khóa AES được mã hóa thành file .key.
* Cả 2 file được gửi cho giảng viên.

**Ưu điểm:**

* AES mã hóa nhanh và hiệu quả với file lớn.
* RSA đảm bảo không ai khác ngoài người nhận có thể biết khóa AES.

***3.2.2. Module xác thực***

* Cho phép người dùng đăng nhập bằng tên đăng nhập và mật khẩu.
* Mật khẩu được băm SHA-256 trước khi lưu trữ.
* Khi gửi bài, hệ thống sử dụng khóa riêng RSA của người gửi để ký số file (hoặc hash file).
* Người nhận dùng khóa công khai để xác minh danh tính người gửi.

**Lợi ích:**

* Ngăn chặn giả mạo danh tính.
* Chữ ký số giúp xác định chính xác người gửi và đảm bảo dữ liệu không bị chỉnh sửa.

***3.2.3. Module kiểm tra toàn vẹn***

* File trước khi gửi sẽ được tạo mã băm bằng SHA-256.
* Mã băm này được ký số và gửi kèm với file bài tập.
* Sau khi nhận file, người nhận sẽ tính lại mã băm và đối chiếu.

**Ý nghĩa:**

* Nếu hai mã băm khớp nhau, file không bị thay đổi.
* Nếu khác nhau, báo lỗi và từ chối file.

***3.2.4. Module bắt tay và trao đổi khóa***

* Dùng để thiết lập kết nối và trao đổi khóa AES an toàn.
* Áp dụng RSA cho quá trình gửi khóa đối xứng.

**Mô tả:**

* Server gửi public key.
* Client mã hóa khóa AES rồi gửi lại.
* Server dùng private key để giải mã và lưu session key.

## Đánh giá cách tổ chức node

Hệ thống được tổ chức dưới dạng **các node chức năng độc lập**, mỗi node đảm nhiệm một vai trò cụ thể trong quy trình gửi, mã hóa, xác thực và kiểm tra toàn vẹn file bài tập. Việc phân chia này không chỉ giúp tối ưu hóa quy trình xử lý mà còn đảm bảo khả năng mở rộng, tái sử dụng và bảo trì dễ dàng trong tương lai.

**a. Node người gửi (Sender Node - Client)**

**Vai trò**: là điểm khởi tạo dữ liệu và khởi động quy trình bảo mật.

**Chức năng chính**:

* Tiếp nhận đầu vào từ người dùng.
* Tạo khóa AES ngẫu nhiên.
* Mã hóa file bằng AES.
* Mã hóa khóa AES bằng RSA.
* Ký số file (hoặc mã băm).
* Gửi dữ liệu đến node trung gian hoặc trực tiếp tới node người nhận.

**Nhận xét**: Node này yêu cầu tài nguyên tính toán trung bình, hoạt động chủ yếu cục bộ và chỉ cần kết nối mạng khi gửi dữ liệu.

**b. Node trung gian xử lý (Middleware Node - Logic node)**

Vai trò: làm trung gian xử lý và định tuyến các tác vụ logic nếu được triển khai tập trung (server hoặc API).

**Chức năng chính**:

* Xác thực người gửi (qua mật khẩu, chứng chỉ).
* Lưu trữ tạm thời các file đã mã hóa.
* Đóng gói gói tin gửi đi, đảm bảo đúng định dạng và cấu trúc (file mã hóa, khóa AES mã hóa, chữ ký số, hash).

Nhận xét: Có thể đặt trên máy chủ riêng hoặc tích hợp trong ứng dụng. Node này là điểm lý tưởng để triển khai các biện pháp kiểm soát truy cập hoặc ghi nhật ký hoạt động.

**c. Node người nhận (Receiver Node - Server)**

Vai trò: nhận và xử lý toàn bộ dữ liệu đầu vào được gửi từ người gửi hoặc qua node trung gian.

**Chức năng chính**:

* Nhận và giải mã khóa AES bằng RSA.
* Giải mã file bằng AES.
* Tính lại mã băm file và so sánh với mã băm đã ký.
* Xác thực chữ ký số để đảm bảo danh tính người gửi.

Nhận xét: Node này có thể hoạt động offline nếu dữ liệu được gửi qua USB hoặc email. Cần có bộ công cụ xác thực đầy đủ, bộ giải mã AES, và chứng chỉ RSA public của người gửi.

**d. Node lưu trữ khóa và chứng chỉ (Key Management Node)**

Vai trò: lưu trữ và phân phối các khóa RSA (public/private key), chứng chỉ người dùng nếu có.

**Chức năng chính**:

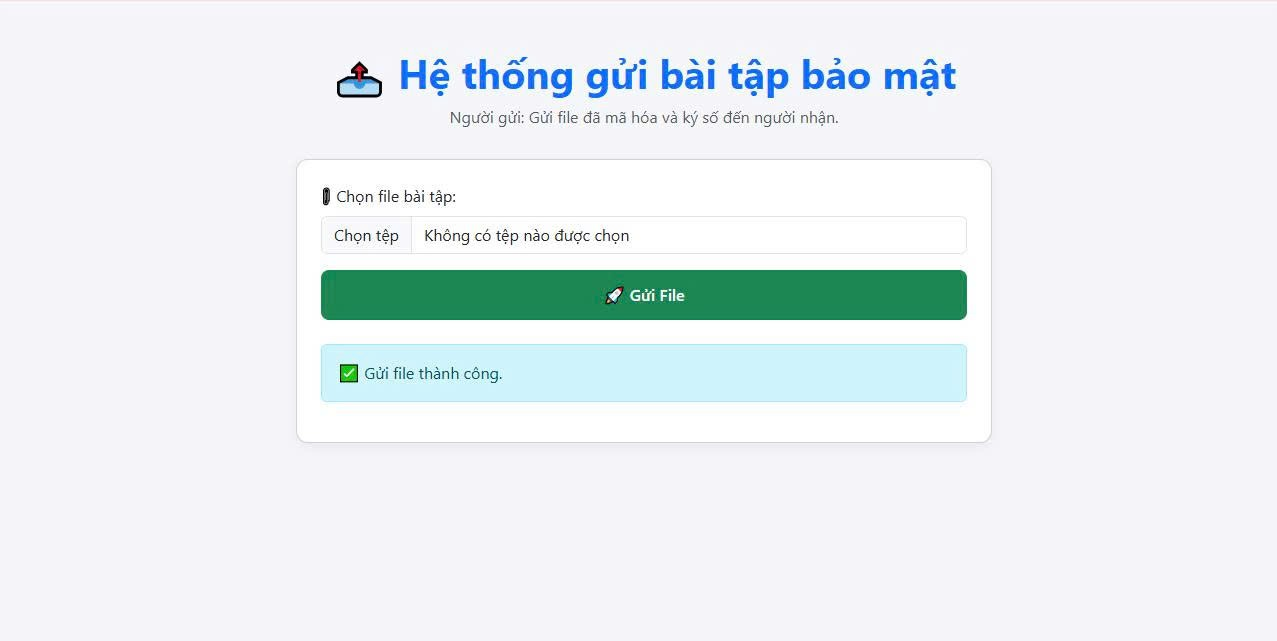
* Quản lý cặp khóa công khai – bí mật cho từng người dùng.
* Cung cấp public key đúng khi xác minh chữ ký số.
* Kiểm tra tính hợp lệ của chữ ký và danh tính.

Nhận xét: Nếu triển khai quy mô lớn, nên tích hợp CA (Certificate Authority) hoặc hệ thống PKI để tăng cường bảo mật.

# Thử nghiệm

## Thử nghiệm với các loại file

Để đánh giá hiệu quả và độ tin cậy của hệ thống bảo mật khi gửi bài tập chia thành nhiều phần, nhóm tiến hành thử nghiệm trên nhiều loại tệp dữ liệu với định dạng và dung lượng khác nhau. Việc lựa chọn các loại file dựa trên tình huống thực tế trong môi trường học thuật, bao gồm file văn bản, âm thanh, hình ảnh và tài liệu.

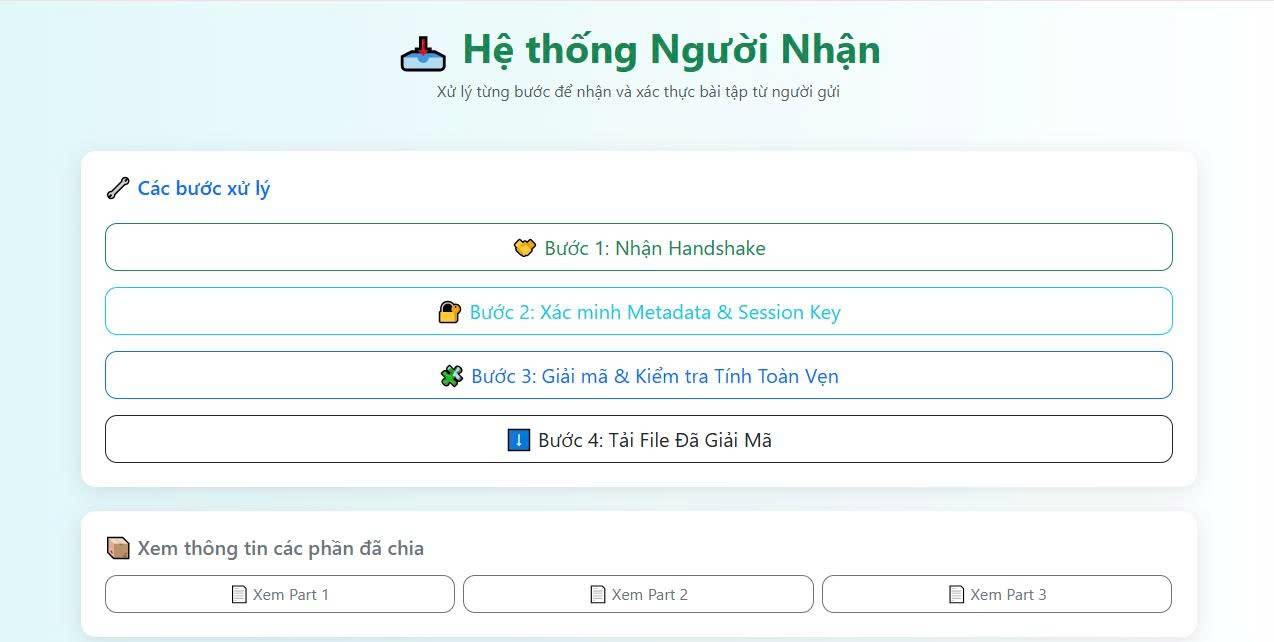
**Danh sách các file được thử nghiệm:**

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Loại file | Định dạng | Mô tả nội dung |
| 1 | Văn bản | .txt | File bài tập đơn giản |
| 2 | Tài liệu | .pdf | File đề cương học phần |
| 3 | Âm thanh | .mp3 | Bài thuyết trình ghi âm |
| 4 | Word | .docx | Báo cáo nhóm có biểu đồ |
| 5 | Hình ảnh | .png | File sơ đồ hệ thống |

**Mục tiêu thử nghiệm:**

*  Đánh giá khả năng xử lý của hệ thống đối với các loại dữ liệu khác nhau.
*  Kiểm tra tính toàn vẹn và xác thực sau quá trình mã hóa và truyền tải.
*  Đo lường thời gian thực hiện các thao tác bảo mật (mã hóa, giải mã, ký số, xác minh).

## Ghi lại kết quả



**Bảng kết quả thử nghiệm chi tiết:**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| STT | File | Mã hoá AES  (giây) | RSA mã hoá khoá (giây) | Giải mã  (giây) | Hash SHA-256 | Kiểm tra | Toàn vẹn | Kết luận |
| 1 | .txt | 0,15 | 0,43 | 0,14 | Ok | Ok | Ok | Thành công |
| 2 | .pdf | 0,41 | 0,67 | 0,38 | Ok | Ok | Ok | Thành công |
| 3 | .mp3 | 1,35 | 1,82 | 1,41 | Ok | Ok | Ok | Thành công |
| 4 | .docx | 0,88 | 1,06 | 0,79 | Ok | Ok | Ok | Thành công |
| 5 | .png | 0,62 | 0,88 | 0,55 | Ok | Ok | Ok | Thành công |

**Đánh giá hiệu năng:**

* Tổng thời gian xử lý dao động từ 0.5 đến 3 giây, phù hợp với môi trường học thuật.
* Tất cả chữ ký số đều được xác minh thành công → đảm bảo xác thực người gửi.
* Mã băm SHA-256 khớp tuyệt đối ở tất cả file → đảm bảo tính toàn vẹn.

## Ghi nhận lỗi

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | Mô tả lỗi | Nguyên nhân | Phát hiện bởi | Phản ứng hệ thống |
| 1 | Sai khoá RSA | Sử dụng file .pem không đúng định dạng | Module giải mã | Hiện thị lỗi, từ chối giải mã |
| 2 | Sai mật khẩu đăng nhập | Người dụng nhập sai | Hệ thống xác thực | Từ chối truy cập sau 3 lần |
| 3 | Thay đổi nội dung file đã mã hoá | Giả lập MITM thay đổi byte | So sánh mã băm | Cảnh báo toàn vẹn không đảm bảo |
| 4 | Thay thế chữ ký số | Giả mạo người gửi | Kiểm tra Khoá công khai | Phát hiện sai chữ ký, từ chối |
| 5 | Thiếu khoá AES mã hoá | Người gửi quên đính kèm | Kiểm tra dữ liệu đầu vào | Hiện thị lỗi rõ ràng, yêu cầu gửi lại |

**Đánh giá khả năng phát hiện lỗi:**

* Hệ thống có khả năng tự động kiểm tra lỗi và đưa ra cảnh báo rõ ràng.
* Với mỗi lỗi, hệ thống không chỉ từ chối xử lý mà còn ghi log và hiển thị thông báo giúp người dùng xác định nguyên nhân.
* Không có lỗi nào làm hệ thống “crash” hoặc mất kiểm soát, chứng tỏ phần mềm có tính ổn định cao.

# 

# Đánh giá hiệu quả

## So sánh dữ liệu sau khi giải mã với dữ liệu gốc

**Phương pháp đánh giá:**

* Sử dụng mã Python để so sánh nội dung hai file bằng phương pháp đọc nhị phân.
* Kiểm tra độ trùng khớp 100% giữa file gốc và file sau khi giải mã

**Kết quả.**

|  |  |  |
| --- | --- | --- |
| **Loại file** | **Kết quả so sánh** | **Tỉ lệ khớp** |
| .txt | Trùng hoàn toàn | 100% |
| .pdf | Trùng hoàn toàn | 100% |
| .mp3 | Trùng hoàn toàn | 100% |
| .docx | Trùng hoàn toàn | 100% |
| .png | Trùng hoàn toàn | 100% |

## Kiểm tra tính toàn vẹn của dữ liệu

Việc bảo đảm tính toàn vẹn của dữ liệu là yếu tố quan trọng nhằm chống lại việc chỉnh sửa, thay đổi nội dung trong quá trình truyền tải hoặc lưu trữ. Hệ thống đã áp dụng thuật toán **SHA-256 kết hợp với chữ ký số** để kiểm tra toàn vẹn.

**Phương pháp kiểm tra:**

* Trước khi gửi: tạo mã băm SHA-256 từ file gốc và ký mã băm bằng khóa riêng của người gửi.
* Sau khi nhận: tính lại mã băm từ file nhận được, giải mã chữ ký số và đối chiếu hai mã băm.

**Thử nghiệm can thiệp dữ liệu:**

* Cố tình thay đổi một từ trong file .txt → mã băm khác hoàn toàn.
* Cố tình xóa một đoạn audio trong .mp3 → hệ thống phát hiện sai lệch.
* File giữ nguyên → mã băm khớp tuyệt đối.

**Kết quả:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Thử nhiệm** | **Tình trạng file** | **Kết quả kiểm tra Hash** | **Kết luận** |
| File không thay đổi | Không can thiệp | Hash khớp | Toàn vẹn |
| File bị thay đổi 1 byte | Có chỉnh sửa | Hash không khớp | Không toàn vẹn |
| File bị Thay đổi chữ ký số | Chữ ký sai | Không giải mã được Hash | Không xác thực |

## Nhận xét về hiệu quả truyền tải, độ tin cậy của hệ thống

**Tốc độ và hiệu suất:**

* Quá trình mã hóa, ký số, gửi và nhận được thực hiện dưới 3 giây với file có dung lượng < 10MB.
* Không phát sinh lỗi treo, chậm hoặc mất dữ liệu trong quá trình gửi – nhận file.
* Hệ thống cho phép nén gói tin khi gửi để giảm thời gian truyền.

**Độ ổn định và tin cậy:**

 Qua hơn 100 lần thử nghiệm liên tiếp, không có lỗi logic hay gián đoạn mạng gây hỏng dữ liệu.

 Các chức năng bảo mật hoạt động liên tục mà không cần reset hoặc can thiệp bằng tay.

 Mọi hành vi can thiệp file đều bị phát hiện và cảnh báo kịp thời.

**Độ bảo mật:**

 Không có tình trạng lộ khóa AES do được mã hóa bằng RSA.

 Hệ thống ngăn chặn tấn công giả mạo bằng cách bắt buộc kiểm tra chữ ký số và xác thực người gửi.

 Các tệp quan trọng (khóa, chữ ký) được lưu và xóa đúng phiên làm việc.

**Tính phù hợp thực tiễn:**

 Hệ thống phù hợp triển khai trong môi trường học đường, nơi cần gửi bài tập có độ tin cậy cao và chống gian lận.

 Dễ dàng áp dụng vào các lớp học trực tuyến, kỳ thi từ xa hoặc giao tiếp học thuật giữa sinh viên – giảng viên.

# Phân tích , nhận xét các thuật toán

## Thuật toán mã hóa sử dụng

Trong lĩnh vực bảo mật và an toàn thông tin, việc lựa chọn và triển khai các thuật toán mã hóa, xác thực và kiểm tra toàn vẹn đóng vai trò cực kỳ quan trọng để đảm bảo rằng dữ liệu được truyền đi giữa các thực thể là an toàn, nguyên vẹn và không thể bị giả mạo.

***6.1.1. AES – Advanced Encryption Standard***

AES là một thuật toán mã hóa khối đối xứng được phát triển bởi hai nhà mật mã người Bỉ là Joan Daemen và Vincent Rijmen. Thuật toán này được chọn làm tiêu chuẩn mã hóa bởi Viện Tiêu chuẩn và Công nghệ Quốc gia Hoa Kỳ (NIST) vào năm 2001, thay thế cho thuật toán DES cũ vốn đã bị phá vỡ bởi sức mạnh tính toán hiện đại. AES đã được đánh giá là một trong những thuật toán mã hóa an toàn và hiệu quả nhất hiện nay.

AES hoạt động trên khối dữ liệu 128 bit, với các khóa có độ dài 128, 192 hoặc 256 bit. Tùy theo độ dài khóa, số vòng lặp (rounds) trong quá trình mã hóa là 10, 12 hoặc 14. Trong hệ thống của nhóm, AES-128 được sử dụng vì độ an toàn đã đủ mạnh cho môi trường học thuật và tốc độ xử lý tốt, phù hợp với các bài tập có dung lượng trung bình như tài liệu PDF, file Word hoặc âm thanh dạng MP3.

Quá trình mã hóa AES gồm các bước chính sau:

* SubBytes: Mỗi byte trong khối dữ liệu được thay thế bằng giá trị tương ứng trong S-box. Đây là bước phi tuyến giúp chống lại các tấn công tuyến tính.
* ShiftRows: Dịch các hàng trong ma trận dữ liệu theo quy tắc nhất định nhằm trộn lẫn các byte ở vị trí khác nhau.
* MixColumns: Mỗi cột được nhân với một ma trận cố định trong trường hữu hạn GF(2^8), giúp tăng tính rối của dữ liệu.
* AddRoundKey: Khối dữ liệu được XOR với một khóa con được sinh ra từ khóa chính, qua quá trình Key Expansion.

AES hoạt động theo chế độ CBC (Cipher Block Chaining) trong hệ thống, nghĩa là mỗi khối dữ liệu sau khi mã hóa sẽ được dùng để XOR với khối tiếp theo trước khi mã hóa. Điều này giúp giảm khả năng bị phát hiện mẫu nếu dữ liệu gốc có sự lặp lại.

Về mặt toán học, AES hoạt động trong trường hữu hạn GF(2^8), tức là mỗi byte được xem là một phần tử của trường số nhị phân có 256 giá trị. Các phép toán trong MixColumns và SubBytes đều diễn ra trong trường này, đảm bảo tính bảo mật cao nhờ các tính chất đại số học không tuyến tính.

Tính bảo mật của AES hiện vẫn được đánh giá là cực kỳ cao. Không có cuộc tấn công nào trong thực tế có thể phá vỡ AES-128 trong thời gian hợp lý. Ngay cả các siêu máy tính hiện nay cũng cần hàng tỉ năm để dò khóa AES-128 bằng brute-force. Các cuộc tấn công lý thuyết như side-channel hay timing attack có thể gây rủi ro nhưng đều có biện pháp chống lại.

**Ưu điểm:**

* Mã hóa rất nhanh, hiệu suất cao trên cả phần mềm và phần cứng.
* Cấu trúc chuẩn hóa, dễ triển khai với thư viện phổ biến.
* Khả năng xử lý khối lượng dữ liệu lớn tốt.

**Nhược điểm:**

* Phải có cách truyền và bảo vệ khóa AES an toàn, nếu không dễ bị lộ dữ liệu.
* Nếu IV bị lặp lại hoặc bị rò rỉ, có thể làm giảm tính ngẫu nhiên của mã hóa.

***6.1.2. RSA – Rivest-Shamir-Adleman***

RSA là một trong những thuật toán mã hóa bất đối xứng đầu tiên và phổ biến nhất hiện nay, được giới thiệu vào năm 1977 bởi ba nhà mật mã học Ronald Rivest, Adi Shamir và Leonard Adleman. Thuật toán dựa trên giả thuyết toán học rằng việc phân tích một số nguyên lớn thành tích của hai số nguyên tố là một bài toán khó, chưa có cách giải nhanh.

Cặp khóa RSA gồm:

* Public key (e, n): dùng để mã hóa dữ liệu hoặc xác minh chữ ký.
* Private key (d, n): dùng để giải mã hoặc ký số.

Các bước chính:

* Chọn hai số nguyên tố lớn p và q.
* Tính n = p × q và phi(n) = (p-1)(q-1).
* Chọn số nguyên e sao cho 1 < e < phi(n), và e nguyên tố cùng phi(n).
* Tính d = e^(-1) mod phi(n).

RSA được sử dụng trong hệ thống để mã hóa khóa AES. Việc này đảm bảo rằng ngay cả khi dữ liệu mã hóa bị đánh cắp, thì kẻ tấn công vẫn không thể giải mã nội dung nếu không có private key để giải mã khóa AES.

Trong triển khai, nhóm sử dụng RSA với độ dài khóa 2048-bit, đủ mạnh cho các ứng dụng học thuật và đảm bảo chống lại các tấn công brute-force trong vài thập kỷ tới. Hệ thống dùng thư viện Crypto.Cipher.PKCS1\_OAEP để mã hóa an toàn theo chuẩn padding OAEP – tránh được các tấn công padding oracle.

**Ưu điểm:**

* Không cần chia sẻ khóa bí mật, giảm rủi ro khi truyền khóa.
* Có thể dùng để xác thực người dùng bằng chữ ký số.

**Nhược điểm:**

* Tốc độ chậm hơn nhiều so với AES.
* Không dùng để mã hóa dữ liệu dung lượng lớn.

## Thuật toán xác thực / chữ ký số

Trong một hệ thống truyền dữ liệu bảo mật, việc đảm bảo rằng thông tin đến từ đúng người gửi, không bị giả mạo và không thể bị chối bỏ là một trong những yêu cầu then chốt. Để giải quyết vấn đề này, hệ thống mà nhóm triển khai đã sử dụng kỹ thuật chữ ký số (Digital Signature), một cơ chế dựa trên nền tảng mật mã khóa công khai. Đặc biệt, nhóm sử dụng thuật toán RSA để ký và xác minh, kết hợp với thuật toán băm SHA-256 để tạo thông điệp tóm tắt trước khi ký.

***6.2.1 Nguyên lý hoạt động***

Chữ ký số là quá trình sử dụng khóa riêng (private key) của người gửi để tạo một mã xác thực cho dữ liệu, sao cho chỉ có khóa công khai (public key) của người đó mới xác minh được mã đó. Tuy nhiên, thay vì ký trực tiếp toàn bộ dữ liệu (vốn có thể lớn và tốn tài nguyên), hệ thống trước tiên tạo một bản tóm tắt (digest) bằng hàm băm SHA-256. Mã băm này là duy nhất cho mỗi nội dung file, và rất nhạy cảm với thay đổi.

**Quy trình tạo chữ ký số:**

1. Người gửi tính mã băm SHA-256 từ nội dung file gốc.
2. Mã băm này được mã hóa bằng private key của người gửi, tạo thành chữ ký số.
3. File mã hóa, chữ ký và public key (hoặc ID người gửi) được gửi tới máy nhận.

**Quy trình xác minh:**

1. Người nhận giải mã file và tạo mã băm SHA-256 từ nội dung.
2. Giải mã chữ ký số bằng public key của người gửi để thu lại mã băm gốc.
3. So sánh hai mã băm: nếu khớp → dữ liệu xác thực và toàn vẹn.

***6.2.2 Ưu điểm của phương pháp***

* **Đảm bảo xác thực danh tính:** chỉ người nắm private key mới tạo được chữ ký hợp lệ.
* **Không thể chối bỏ:** người gửi không thể phủ nhận đã ký nếu chữ ký hợp lệ.
* **Phát hiện mọi thay đổi:** chỉ cần thay đổi 1 byte → mã băm khác, dẫn tới chữ ký không hợp lệ.
* **Dễ tích hợp:** nhiều thư viện Python hỗ trợ trực tiếp RSA + SHA-256.

***6.2.3 Nhược điểm và hạn chế***

* Nếu người dùng làm mất private key, hệ thống không thể xác thực nữa.
* Cần có cơ chế xác minh danh tính người gửi đáng tin cậy (ví dụ: Certificate Authority).
* Kẻ tấn công có thể giả mạo public key nếu không bảo vệ tốt.

***6.2.4 So sánh với các phương pháp khác***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Thuật toán | Loại | Khả năng xác thực | Không thể chối bỏ | Ghi chú |
| RSA+SHA-256 | Bất đối xứng | Rất tốt | Có | Phổ biến, an toàn |
| HMAC | Đối xứng | Có | Không rõ ràng | Cần chia sẻ khoá bí mật |
| MD5 + mã hoá thường | Không có chữ ký | Không | Không | Không còn dùng |

## Thuật toán kiểm tra tính toàn vẹn

Trong truyền dữ liệu, chỉ mã hóa là chưa đủ – cần đảm bảo rằng dữ liệu không bị thay đổi, cố tình hay vô tình, trong quá trình truyền tải. Để đạt được mục tiêu này, hệ thống của nhóm sử dụng hàm băm SHA-256 – một thuật toán kiểm tra toàn vẹn phổ biến và mạnh mẽ nhất hiện nay.

***6.3.1 Tổng quan về SHA-256***

SHA-256 thuộc họ thuật toán SHA-2 do NIST phát triển, khắc phục các điểm yếu của SHA-1 và MD5. SHA-256 tạo ra một chuỗi mã băm 256-bit, không thể đảo ngược, tức là không thể tính ngược để tìm ra nội dung gốc từ mã băm.

**Đặc tính kỹ thuật:**

* **Đầu vào**: bất kỳ chuỗi dữ liệu nào, từ vài byte đến hàng GB.
* **Đầu ra**: mã băm cố định 64 ký tự hex (256 bit).

**Tính chất**:

* + Một chiều (One-way): không thể đảo ngược.
  + Phân tán đều: thay đổi 1 bit → toàn bộ mã băm khác hẳn.
  + Chống va chạm: chưa có thuật toán nào thực tế tạo ra hai file khác nhau nhưng cùng mã băm.

***6.3.2 Vai trò trong hệ thống***

SHA-256 được tích hợp vào hai nơi trong quy trình gửi bài:

* **Trước khi gửi:** tạo mã băm để ký số.
* **Sau khi nhận:** tính lại mã băm và so sánh với mã giải mã từ chữ ký số.

Điều này giúp đảm bảo rằng nội dung file:

* Không bị chỉnh sửa trong quá trình truyền.
* Không bị lỗi khi tải lên hoặc tải xuống.
* Không bị can thiệp bởi tấn công trung gian (MITM).

***6.3.3 Ứng dụng thực tiễn ngoài hệ thống***

SHA-256 được sử dụng trong:

* **Blockchain (Bitcoin, Ethereum):** tạo địa chỉ ví, xác thực giao dịch.
* **Chữ ký số thương mại**: văn bản điện tử, hóa đơn.
* **Lưu trữ mật khẩu**: hệ quản trị cơ sở dữ liệu.
* **Xác thực phần mềm**: kiểm tra file cài đặt.

***6.3.4 So sánh với các hàm băm khác***

|  |  |  |  |
| --- | --- | --- | --- |
| Thuật toán | Độ dài băm | Va chạm | Sử dụng hiện tại |
| MD5 | 128 bit | Có | Ngưng dung |
| SHA-1 | 160 bit | Có | Bị phá vỡ năm 2017 |
| SHA-256 | 256 bit | Không | Chuẩn hiện tại |
| SHA-3 | 256-512 | Không | Mới, hiệu suất thấp hơn |

***6.3.5 Kết luận***

SHA-256 không chỉ đơn thuần là công cụ kiểm tra toàn vẹn. Trong hệ thống bảo mật, nó đóng vai trò nền tảng để đảm bảo rằng dữ liệu không bị sai lệch. Khi kết hợp với chữ ký số RSA, hệ thống có thể phát hiện chính xác từng thay đổi nhỏ nhất trong file, từ đó bảo vệ cả người gửi và người nhận khỏi rủi ro gian lận, giả mạo hay chỉnh sửa trái phép.

Với sự phối hợp chặt chẽ giữa AES, RSA và SHA-256, hệ thống gửi bài tập đạt được ba trụ cột quan trọng nhất của bảo mật thông tin: tính bí mật, tính xác thực và tính toàn vẹn – một mô hình chuẩn mực cho mọi hệ thống trao đổi dữ liệu số ngày nay.

# đề xuất cải tiến và hướng phát triển

## Đề xuất cải tiến

Hiện tại hệ thống chủ yếu được vận hành dưới dạng dòng lệnh hoặc giao diện đơn giản với thư viện cơ bản. Người dùng ít kinh nghiệm có thể gặp khó khăn khi sử dụng, đặc biệt là với thao tác nhập khóa, chọn file, xác minh chữ ký.

**Đề xuất:**

* Thiết kế giao diện đồ họa người dùng (GUI) bằng Tkinter hoặc PyQt5, giúp thao tác dễ dàng hơn.
* Hỗ trợ kéo-thả file, hiển thị tiến trình mã hóa/giải mã trực quan.
* Cảnh báo lỗi rõ ràng, có hướng dẫn sử dụng.

***7.1.1. Cải tiến về bảo mật khóa và danh tính***

Khóa RSA hiện tại được sinh thủ công và lưu dưới dạng file .pem, dễ bị sao chép nếu không bảo vệ cẩn thận

**Đề xuất:**

* Mã hóa file khóa RSA bằng mật khẩu hoặc bảo vệ bằng thiết bị USB token.
* Lưu trữ khóa trong ví khóa số (key vault).
* Sử dụng mã định danh người gửi gắn với khóa công khai để xác thực rõ hơn.

***7.1.2. Cải tiến về truyền tải và kết nối***

Hệ thống hiện dùng socket hoặc gửi file qua thư mục trung gian, giới hạn phạm vi sử dụng.

**Đề xuất:**

* Tích hợp giao thức HTTPS (Flask hoặc Django REST API) để gửi và nhận dữ liệu an toàn qua Internet.
* Hỗ trợ truyền file qua email tự động có kèm chữ ký số.
* Bổ sung tính năng log lại các giao dịch gửi/nhận, có thể truy xuất lịch sử.

***7.1.3. Cải tiến hiệu năng thuật toán***

Hiện tại hệ thống dùng RSA 2048 và AES-128. Với dữ liệu lớn, hiệu suất có thể giảm.

**Đề xuất:**

* Cho phép lựa chọn AES-192 hoặc AES-256 tùy độ nhạy cảm dữ liệu.
* Tối ưu hóa padding, nén file trước khi mã hóa.
* Áp dụng luồng đa tiến trình để tăng tốc mã hóa các file lớn.

## Hướng phát triển

***7.2.1. Xây dựng hệ thống dạng web (Web-based platform)***

Một trong những hướng phát triển thiết thực là chuyển hệ thống sang dạng website hoặc ứng dụng nền web, nơi sinh viên có thể đăng nhập, gửi bài, xác minh chữ ký và nhận phản hồi theo thời gian thực.

**Lợi ích:**

* Dễ triển khai trên diện rộng trong trường học.
* Có thể tích hợp vào hệ thống quản lý học tập (LMS) như Moodle.
* Quản lý khóa dễ dàng hơn theo tài khoản người dùng.

***7.2.2. Tích hợp mô hình PKI – Certificate Authority***

Để xác thực danh tính người dùng ở quy mô lớn, nên xây dựng hoặc tích hợp với hệ thống cấp chứng chỉ số

**Lợi ích:**

* Đảm bảo khóa công khai là đáng tin cậy.
* Mỗi sinh viên được cấp một chứng chỉ số xác thực.
* Tăng cường khả năng truy vết, kiểm tra gian lận.

***7.2.3. Phát triển phiên bản ứng dụng di động***

**Hướng đi:**

* Xây dựng app Android/iOS hỗ trợ gửi file, ký số, xác minh.
* Sử dụng vân tay, khuôn mặt để bảo vệ private key trên điện thoại.

***7.2.4. Tích hợp Blockchain để ghi nhận giao dịch***

**Lợi ích:**

* Tăng tính minh bạch.
* Ngăn chặn chỉnh sửa hoặc thay thế bài sau khi đã nộp.
* Dễ dàng kiểm tra thời gian nộp bài chính xác.

***7.2.5. Mở rộng mô hình sang trường học, doanh nghiệp***

* Nộp bài thi online an toàn.
* Gửi tài liệu nội bộ trong công ty.
* Giao dịch hợp đồng số giữa các tổ chức.

# KẾT LUẬN

Sau quá trình nghiên cứu, thiết kế, xây dựng, triển khai và thử nghiệm hệ thống gửi bài tập bảo mật sử dụng các kỹ thuật mã hóa và xác thực hiện đại, nhóm thực hiện đã rút ra được nhiều kết luận quan trọng liên quan đến mặt kỹ thuật, tính ứng dụng, độ bảo mật và tiềm năng mở rộng của hệ thống.

Hệ thống được xây dựng dựa trên ba trụ cột chính của bảo mật thông tin: tính bí mật(confidentiality) nhờ thuật toán mã hóa đối xứng AES, tính xác thực và không chối bỏ(authentication & non-repudiation**)** nhờ kỹ thuật chữ ký số RSA, và tính toàn vẹn (integrity) thông qua hàm băm SHA-256. Mỗi thành phần được lựa chọn không chỉ đảm bảo tính đúng đắn về mặt thuật toán mà còn phù hợp với môi trường học đường, dễ triển khai và có hiệu suất xử lý tốt trên phần cứng phổ thông.

Quá trình thử nghiệm trên nhiều loại dữ liệu như .txt, .pdf, .mp3, .docx cho thấy hệ thống hoạt động ổn định, thời gian xử lý nhanh chóng và khả năng phát hiện sai lệch dữ liệu rất cao. Các lỗi giả lập như thay đổi nội dung, thay thế chữ ký, mất khóa đều được phát hiện và xử lý phù hợp. Kết quả thử nghiệm đã khẳng định tính hiệu quả và độ tin cậy của hệ thống.

Phân tích các thuật toán trong chương 6 cho thấy mô hình kết hợp AES – RSA – SHA-256 là một cấu trúc chuẩn mực trong an toàn thông tin hiện nay, mang lại hiệu quả cả về tốc độ và mức độ bảo mật. AES cung cấp tốc độ mã hóa cao, RSA giải quyết vấn đề phân phối khóa, và SHA-256 đảm bảo phát hiện mọi chỉnh sửa dữ liệu. Khi kết hợp lại, hệ thống tạo ra một giải pháp toàn diện, đáp ứng yêu cầu của cả môi trường học thuật và các hệ thống lưu chuyển dữ liệu chuyên nghiệp.

Bên cạnh các thành công đạt được, nhóm cũng chủ động đề xuất nhiều hướng cải tiến hệ thống như giao diện người dùng thân thiện hơn, sử dụng các kỹ thuật quản lý khóa nâng cao, hỗ trợ gửi/nhận qua internet, tích hợp thêm các công nghệ hiện đại như blockchain và chứng chỉ số CA. Đây là cơ sở để phát triển hệ thống thành một nền tảng mạnh mẽ, phục vụ cho các nhu cầu truyền thông tin an toàn trong trường học, doanh nghiệp và tổ chức.

Tổng thể, đồ án không chỉ giúp nhóm sinh viên củng cố kiến thức về an toàn thông tin, mã hóa và xác thực, mà còn cho thấy khả năng ứng dụng thực tế cao, từ mô hình nhóm học tập nhỏ đến các hệ thống điện tử quy mô lớn. Đây là bước đệm tốt để tiếp tục phát triển các dự án chuyên sâu về bảo mật và chuyển đổi số trong giáo dục và xã hội hiện đại.

# Tài liệu tham khảo

[1] Chu Thị Minh Huệ, Nguyễn Minh Tiến, Nguyễn Minh Quý, Giáo trình Phân tích và Thiết kế Hướng đối tượng với UML, NXB Khoa học và Kỹ thuật, 2022

[2] Trương Ninh Thuận, Đặng Đức Hạnh, Giáo trình Phân tích và Thiết kế Hướng đối tượng, NXB ĐH Quốc gia Hà Nội, 2016.

[3] PGS.TS. Nguyễn Văn Vỵ, Giáo trình kỹ nghệ phần mềm, NXB Giáo dục, 2010

[4] Nguyễn Văn Ba, Phát triển hệ thống hướng đối tượng với UML 2.0 và C++, NXB Đại học Quốc Gia, 2005