**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC ĐẠI NAM**

Logo, company name

Description automatically generated

**BÀI TẬP LỚN**

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

**ĐỀ TÀI: ỨNG DỤNG BẢO MẬT TIN NHẮN VĂN BẢN VỚI**

**MÃ HÓA DES VÀ XÁC THỰC RSA**

**Giáo viên hướng dẫn: Trần Đức Thắng**

**Sinh viên thực hiện:**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Mã SV** | **Họ và tên** | **Lớp** |
| **1** | **1671020180** | **Nguyễn Thị Thùy Linh** | **CNTT 16-05** |
| **2** | **1671020199** | **Hồ Đức Mạnh** | **CNTT 16-05** |
| **3** | **1671020184** | **Đỗ Duy Long** | **CNTT 16-05** |
| **4** | **1671020194** | **Nguyễn Thị Viết Lợi** | **CNTT 16-05** |

**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: AN TOÀN, BẢO MẬT THÔNG TIN**

**ĐỀ TÀI: ỨNG DỤNG BẢO MẬT TIN NHẮN VĂN BẢN VỚI MÃ HÓA DES VÀ XÁC THỰC RSA**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **STT** | **Mã Sinh Viên** | **Họ và Tên** | **Ngày Sinh** | **Điểm** | |
| **Bằng Số** | **Bằng Chữ** |
| **1** | **1671020180** | **Nguyễn Thị Thùy Linh** | **01/08/2004** |  |  |
| **2** | **1671020199** | **Hồ Đức Mạnh** | **01/05/2004** |  |  |
| **3** | **1671020184** | **Đỗ Duy Long** | **22/11/2004** |  |  |
| **4** | **1671020194** | **Nguyễn Thị Viết Lợi** | **18/09/2004** |  |  |

**CÁN BỘ CHẤM THI**

**Hà Nội, năm 2025**

# **LỜI NÓI ĐẦU**

Trong thời đại công nghệ phát triển mạnh mẽ, nhu cầu trao đổi thông tin qua các ứng dụng nhắn tin trực tuyến ngày càng phổ biến, mang lại nhiều tiện ích cho người sử dụng. Tuy nhiên, cùng với sự phát triển đó là những rủi ro tiềm ẩn về bảo mật thông tin, khiến người dùng dễ bị nghe lén, giả mạo hoặc chỉnh sửa nội dung tin nhắn trong quá trình truyền tải. Do đó, việc bảo vệ thông tin người dùng và đảm bảo an toàn khi truyền dữ liệu trở thành một yêu cầu quan trọng và cấp thiết.

Môn An toàn và Bảo mật thông tin không chỉ cung cấp kiến thức lý thuyết về các thuật toán mã hóa và kỹ thuật bảo mật, mà còn rèn luyện khả năng vận dụng kiến thức đó vào thực tế thông qua các bài tập lớn. Trong bài tập lớn lần này, nhóm em thực hiện đề tài “Ứng dụng bảo mật tin nhắn văn bản với mã hóa DES và xác thực RSA”, với mục tiêu xây dựng một hệ thống nhắn tin văn bản bảo mật, đảm bảo tính bí mật, tính toàn vẹn dữ liệu và xác thực danh tính người gửi, người nhận thông qua các thuật toán bảo mật đã học.

Thông qua bài tập lớn, nhóm em mong muốn hiểu rõ hơn cách triển khai các thuật toán bảo mật trong thực tế, đồng thời rèn luyện kỹ năng lập trình, tư duy phân tích, giải quyết vấn đề và kỹ năng làm việc nhóm, góp phần củng cố kiến thức đã học trên giảng đường.

Nhóm xin gửi lời cảm ơn đến Quý thầy cô bộ môn An toàn và Bảo mật thông tin đã tận tình hướng dẫn và tạo điều kiện cho nhóm thực hiện bài tập lớn này. Chúng em kính mong nhận được sự góp ý, nhận xét từ thầy cô để bài báo cáo được hoàn thiện hơn.

# **MỤC LỤC**

[LỜI NÓI ĐẦU 1](#_Toc202552606)

[MỤC LỤC 2](#_Toc202552607)

[CHƯƠNG 1: GIỚI THIỆU ĐỀ TÀI 4](#_Toc202552608)

[1.1. Đặt vấn đề 4](#_Toc202552609)

[1.2. Mục tiêu đề tài 4](#_Toc202552610)

[1.3. Phạm vi nghiên cứu 5](#_Toc202552611)

[1.4. Ý nghĩa và ứng dụng thực tiễn 5](#_Toc202552612)

[CHƯƠNG 2: CƠ SỞ LÝ THUYẾT 6](#_Toc202552613)

[2.1. Mã hóa đối xứng – Thuật toán DES 6](#_Toc202552614)

[2.1.1. Khái niệm mã hóa đối xứng 6](#_Toc202552615)

[2.1.2. Thuật toán DES 6](#_Toc202552616)

[2.1.3. Chế độ hoạt động CFB 6](#_Toc202552617)

[2.2. Mã hóa bất đối xứng – Thuật toán RSA 7](#_Toc202552618)

[2.2.1. Khái niệm mã hóa bất đối xứng 7](#_Toc202552619)

[2.2.2. Thuật toán RSA 7](#_Toc202552620)

[2.2.3. Mã hóa khóa DES bằng RSA 7](#_Toc202552621)

[2.3. Hàm băm SHA-256 – Kiểm tra tính toàn vẹn 8](#_Toc202552622)

[2.3.1. Khái niệm hàm băm 8](#_Toc202552623)

[2.3.2. Thuật toán SHA-256 8](#_Toc202552624)

[2.4. Ký số và xác thực chữ ký số 8](#_Toc202552625)

[2.4.1. Ký số với RSA 8](#_Toc202552626)

[2.4.2. Xác minh chữ ký số 8](#_Toc202552627)

[2.4.3. Tính pháp lý & ứng dụng thực tiễn 9](#_Toc202552628)

[CHƯƠNG 3: TRIỂN KHAI VÀ KẾT QUẢ THỰC NGHIỆM 10](#_Toc202552629)

[3.1. Kiến trúc tổng thể của hệ thống 10](#_Toc202552630)

[3.2. Mô hình dữ liệu và luồng xử lý 10](#_Toc202552631)

[3.3. Cấu trúc gói tin truyền đi 11](#_Toc202552632)

[3.4. Phân tích chi tiết từng bước 12](#_Toc202552633)

[3.5. Một số điểm bảo mật đáng chú ý 14](#_Toc202552634)

[CHƯƠNG 4: CÀI ĐẶT HỆ THỐNG 15](#_Toc202552635)

[4.1. Ngôn ngữ và công cụ lập trình 15](#_Toc202552636)

[4.2. Cấu trúc thư mục và file nguồn 16](#_Toc202552637)

[4.3. Tạo khóa RSA (generate\_keys\_socket.py) 16](#_Toc202552638)

[4.4. Giao diện người gửi (client\_socket.py) 17](#_Toc202552639)

[4.4.1. Handshake – Kết nối và trao đổi khóa công khai RSA 17](#_Toc202552640)

[4.4.2. Mã hóa tin nhắn với DES và tạo chữ ký RSA 18](#_Toc202552641)

[4.4.3. Tạo gói tin JSON và gửi đi 18](#_Toc202552642)

[4.5. Giao diện người nhận (server\_socket.py) 19](#_Toc202552643)

[4.5.1. Nhận gói tin và tách dữ liệu 19](#_Toc202552644)

[4.5.2. Giải mã khóa DES bằng RSA private key 19](#_Toc202552645)

[4.5.3. Xác thực chữ ký số bằng khóa công khai của người gửi 19](#_Toc202552646)

[4.5.4. Hiển thị kết quả và phản hồi 20](#_Toc202552647)

[4.6. Kết nối và chạy thử 20](#_Toc202552648)

[CHƯƠNG 5: KẾT QUẢ THỰC NGHIỆM 22](#_Toc202552649)

[5.1. Môi trường thực nghiệm 22](#_Toc202552650)

[5.2. Tình huống gửi tin thành công 22](#_Toc202552651)

[5.3. Tình huống tin nhắn bị thay đổi 23](#_Toc202552652)

[5.4. Tình huống chữ ký bị sai 24](#_Toc202552653)

[5.5. Tình huống thiếu dữ liệu 25](#_Toc202552654)

[5.6. Bảng tổng kết các tình huống 25](#_Toc202552655)

[5.7. Nhận xét 26](#_Toc202552656)

[KẾT LUẬN 27](#_Toc202552657)

[TÀI LIỆU THAM KHẢO 29](#_Toc202552658)

# **CHƯƠNG 1: GIỚI THIỆU ĐỀ TÀI**

## **1.1. Đặt vấn đề**

Trong kỷ nguyên số ngày nay, thông tin là một tài sản quý giá và việc bảo vệ thông tin cá nhân, dữ liệu nhạy cảm trở thành một yêu cầu bắt buộc trong hầu hết các hệ thống thông tin. Một trong những hình thức phổ biến nhất của việc trao đổi thông tin chính là nhắn tin văn bản. Tuy nhiên, các nền tảng truyền thông điệp đơn giản thông qua mạng Internet hay mạng LAN nếu không được mã hóa hoặc xác thực, rất dễ trở thành mục tiêu của các cuộc tấn công như nghe lén (eavesdropping), giả mạo danh tính (impersonation), hoặc thay đổi nội dung thông điệp (message tampering).

Từ thực trạng đó, nhóm thực hiện đề tài “Ứng dụng bảo mật tin nhắn văn bản với mã hoá DES và xác thực RSA” nhằm xây dựng một hệ thống có thể mã hóa nội dung tin nhắn, xác thực danh tính người gửi và kiểm tra tính toàn vẹn dữ liệu thông qua các thuật toán mã hóa hiện đại như DES (Data Encryption Standard), RSA (Rivest-Shamir-Adleman) và SHA-256 (Secure Hash Algorithm).

Đề tài này không chỉ có ý nghĩa về mặt học thuật khi áp dụng kiến thức đã học vào một bài toán thực tiễn, mà còn đóng vai trò như một bước chuẩn bị quan trọng cho sinh viên bước vào lĩnh vực bảo mật thông tin - một lĩnh vực đang ngày càng phát triển và được chú trọng.

## **1.2. Mục tiêu đề tài**

Mục tiêu chính của đề tài là thiết kế và xây dựng một ứng dụng bảo mật cho hệ thống nhắn tin văn bản đơn giản giữa hai máy, trong đó đảm bảo:

* Tính bí mật (Confidentiality): Nội dung tin nhắn được mã hóa bằng thuật toán mã hóa đối xứng DES ở chế độ CFB trước khi được gửi đi, giúp ngăn chặn việc đọc lén nếu dữ liệu bị đánh cắp.
* Tính xác thực (Authentication): Người gửi được xác thực bằng cách ký điện tử ID người dùng bằng thuật toán RSA (với độ dài khóa 2048 bit), cho phép người nhận kiểm tra nguồn gốc thông điệp.
* Tính toàn vẹn (Integrity): Dữ liệu được kiểm tra tính toàn vẹn bằng cách sử dụng hàm băm SHA-256 trên phần tin nhắn đã mã hóa. Điều này giúp phát hiện nếu thông điệp bị thay đổi trong quá trình truyền tải.

## **1.3. Phạm vi nghiên cứu**

Đề tài tập trung vào việc xây dựng ứng dụng mô phỏng quá trình truyền nhận tin nhắn bảo mật giữa một người gửi (Sender) và một người nhận (Receiver), thông qua kết nối mạng LAN sử dụng socket TCP.

Các thành phần chính bao gồm:

* Chương trình người gửi: Có giao diện nhập nội dung tin nhắn, ID, thực hiện ký số, mã hóa tin nhắn và gửi qua socket.
* Chương trình người nhận: Giao diện nhận tin, tự động giải mã và xác thực nội dung, kiểm tra hash và phản hồi kết quả (ACK/NACK).
* Cơ chế handshake: Hai bên trao đổi khóa công khai RSA trước khi truyền tin.
* Không đề cập đến các khía cạnh như mã hóa đầu cuối đa điểm (end-to-end multi-peer), chống phát lại (replay attack), hay lưu trữ tin nhắn.

## **1.4. Ý nghĩa và ứng dụng thực tiễn**

Mặc dù chỉ là một mô hình đơn giản, đề tài này mang lại nhiều ý nghĩa thiết thực trong việc:

* Hiểu sâu về ứng dụng các thuật toán mã hóa trong bảo vệ dữ liệu truyền thông.
* Mô phỏng quy trình bảo mật trong các hệ thống thực tế, ví dụ như ứng dụng chat, email, hệ thống thanh toán hoặc chữ ký điện tử.
* Tăng cường kỹ năng lập trình bảo mật, xử lý dữ liệu nhị phân, làm việc với socket và quản lý khóa công khai/riêng tư.

Về mặt ứng dụng, hệ thống có thể được mở rộng thành một ứng dụng chat bảo mật nội bộ cho doanh nghiệp nhỏ, hoặc làm nền tảng học tập, demo cho các mô hình đào tạo liên quan đến an toàn thông tin và mã hóa dữ liệu.

**CHƯƠNG 2: CƠ SỞ LÝ THUYẾT**

## **2.1. Mã hóa đối xứng – Thuật toán DES**

### ***2.1.1. Khái niệm mã hóa đối xứng***

Mã hóa đối xứng (symmetric encryption) là phương pháp mã hóa trong đó cùng một khóa được sử dụng để mã hóa (encrypt) và giải mã (decrypt) dữ liệu. Việc bảo mật dữ liệu trong mã hóa đối xứng phụ thuộc vào việc giữ kín khóa bí mật.

### ***2.1.2. Thuật toán DES***

DES (Data Encryption Standard) là một trong những thuật toán mã hóa đối xứng cổ điển, do IBM phát triển và được chính phủ Mỹ sử dụng rộng rãi trong nhiều năm. DES sử dụng khóa 56-bit để mã hóa các khối dữ liệu có độ dài 64-bit.

Mặc dù hiện nay DES được xem là không còn đủ mạnh để bảo mật trong môi trường thực tế quy mô lớn (vì khóa ngắn, dễ bị brute-force), nhưng trong các ứng dụng học thuật, demo hoặc truyền tin nội bộ, DES vẫn là công cụ hữu ích để hiểu cơ bản về mã hóa khối.

### ***2.1.3. Chế độ hoạt động CFB***

Trong đề tài này, DES được sử dụng ở chế độ CFB (Cipher Feedback Mode). Đây là một trong những chế độ mã hóa khối cho phép biến một thuật toán mã hóa khối thành mã hóa dòng. CFB sử dụng chuỗi phản hồi từ kết quả mã hóa trước đó cùng với một vector khởi tạo (IV) để đảm bảo rằng ngay cả khi tin nhắn giống nhau được gửi 2 lần thì ciphertext cũng khác nhau.

Ưu điểm của CFB:

* Không cần đệm (padding) như ECB hay CBC.
* Có khả năng xử lý dữ liệu theo byte (dòng).
* Dễ triển khai khi truyền dữ liệu theo luồng (stream-based).

## **2.2. Mã hóa bất đối xứng – Thuật toán RSA**

### ***2.2.1. Khái niệm mã hóa bất đối xứng***

Khác với mã hóa đối xứng, mã hóa bất đối xứng (asymmetric encryption) sử dụng một cặp khóa:

* Khóa công khai (public key) để mã hóa dữ liệu.
* Khóa riêng (private key) để giải mã dữ liệu.

Ưu điểm lớn nhất của mã hóa bất đối xứng là không cần truyền khóa bí mật qua mạng. Người gửi có thể mã hóa bằng khóa công khai của người nhận, và chỉ người nhận mới có thể giải mã bằng khóa riêng của họ.

### ***2.2.2. Thuật toán RSA***

RSA là thuật toán mã hóa bất đối xứng được phát minh bởi Rivest, Shamir và Adleman vào năm 1977. RSA là một trong những thuật toán phổ biến nhất hiện nay trong việc:

* Trao đổi khóa một cách an toàn.
* Xác thực người dùng qua chữ ký số.

RSA dựa trên nguyên lý khó phân tích thừa số của một số nguyên lớn, do đó độ bảo mật phụ thuộc vào độ dài khóa. Trong đề tài này, nhóm sử dụng RSA với độ dài khóa 2048-bit, một mức bảo mật tiêu chuẩn và đủ mạnh trong các ứng dụng học thuật.

### ***2.2.3. Mã hóa khóa DES bằng RSA***

**Trong đề tài, người gửi sẽ:**

1. Sinh một khóa DES ngẫu nhiên.
2. Mã hóa khóa DES này bằng khóa công khai RSA của người nhận.
3. Gửi khóa DES đã mã hóa đi cùng với tin nhắn.

→ Việc này giúp chỉ người nhận có thể giải mã và sử dụng khóa DES để đọc nội dung tin nhắn.

## **2.3. Hàm băm SHA-256 – Kiểm tra tính toàn vẹn**

### ***2.3.1. Khái niệm hàm băm***

Hàm băm (hash function) là một hàm toán học nhận đầu vào là dữ liệu bất kỳ và tạo ra một chuỗi có độ dài cố định gọi là giá trị băm (hash value). Mục đích của hàm băm là:

* Xác minh tính toàn vẹn của dữ liệu.
* Không thể đảo ngược từ giá trị băm về dữ liệu gốc.

### ***2.3.2. Thuật toán SHA-256***

SHA-256 là một thành viên của gia đình SHA-2, được phát triển bởi NSA (Hoa Kỳ). Nó cho ra kết quả 256-bit (64 ký tự hex). Trong đề tài, SHA-256 được sử dụng để:

* Tính hash của tin nhắn sau khi mã hóa (ciphertext).
* Gửi kèm hash trong gói tin để người nhận so sánh.

→ Nếu hash người nhận tự tính ra không trùng khớp với hash nhận được → dữ liệu đã bị thay đổi.

## **2.4. Ký số và xác thực chữ ký số**

### ***2.4.1. Ký số với RSA***

Ký số là quá trình người gửi sử dụng khóa riêng RSA để ký một thông điệp (ví dụ: ID người gửi hoặc bản băm). Trong đề tài, người gửi:

* Dùng SHA-256 băm ID.
* Dùng RSA (khóa riêng) để ký bản băm → tạo chữ ký số (sig).

### ***2.4.2. Xác minh chữ ký số***

Người nhận sử dụng khóa công khai của người gửi để xác minh:

* Dữ liệu có đúng là do người gửi tạo ra hay không?
* Dữ liệu có bị thay đổi sau khi ký hay không?

Việc xác thực chữ ký trong đề tài này nhằm đảm bảo:

* Tin nhắn thực sự đến từ người gửi, không phải kẻ giả mạo.
* ID gửi đi chưa bị chỉnh sửa.

### ***2.4.3. Tính pháp lý & ứng dụng thực tiễn***

Chữ ký số không chỉ là công cụ kỹ thuật mà còn có tính pháp lý trong thực tế như:

* Chứng thực hóa đơn điện tử
* Ký văn bản pháp lý
* Giao dịch ngân hàng điện tử

Dù đề tài chỉ áp dụng ở mức mô phỏng, nhưng cách thức tương tự hoàn toàn có thể triển khai trong các hệ thống thực tế.

# **CHƯƠNG 3: TRIỂN KHAI VÀ KẾT QUẢ THỰC NGHIỆM**

## **3.1. Kiến trúc tổng thể của hệ thống**

Hệ thống được xây dựng theo mô hình **Client–Server ngang hàng (P2P đơn giản)** gồm **hai thành phần chính**:

* **Người gửi (Sender)**: tạo và gửi tin nhắn văn bản, thực hiện mã hóa, ký số và gửi gói tin.
* **Người nhận (Receiver)**: tiếp nhận gói tin, kiểm tra tính toàn vẹn, xác thực và giải mã để đọc nội dung.

Hai bên **giao tiếp trực tiếp với nhau** thông qua **socket TCP**, đảm bảo luồng dữ liệu ổn định, không bị mất mát.

Hệ thống **hoạt động theo trình tự các bước bảo mật** chuẩn:

1. **Handshake** và trao đổi khóa công khai RSA.
2. **Ký số & mã hóa khóa DES**.
3. **Mã hóa nội dung và tạo hash kiểm tra toàn vẹn**.
4. **Gửi gói tin** và phản hồi ACK/NACK.

## **3.2. Mô hình dữ liệu và luồng xử lý**

Dưới đây là sơ đồ mô tả **luồng xử lý logic** từ lúc bắt đầu gửi tin đến khi nhận phản hồi:

**Sơ đồ tổng quan xử lý:**

Ảnh có chứa văn bản, biểu đồ, Phông chữ, chữ viết tay

Nội dung do AI tạo ra có thể không chính xác.

## **3.3. Cấu trúc gói tin truyền đi**

Gói tin được gửi từ người gửi sang người nhận có định dạng JSON như sau:

{

"iv": "<Base64 của vector khởi tạo>",

"cipher": "<Base64 ciphertext tin nhắn>",

"hash": "<SHA-256 hash dạng hex>",

"sig": "<Base64 chữ ký số của ID>",

"signed\_info": "<Base64 của bản băm ID>",

"encrypted\_des\_key": "<Base64 DES key đã mã hóa bằng RSA>",

"sender\_id": "NguyenVanA",

"sender\_pub": "<Public key RSA của người gửi dạng PEM>"

}

**Ý nghĩa từng trường:**

| **Trường** | **Mục đích** |
| --- | --- |
| iv | Vector khởi tạo (Initialization Vector) cho DES-CFB |
| cipher | Tin nhắn sau khi mã hóa bằng DES |
| hash | Mã băm SHA-256 của ciphertext để kiểm tra toàn vẹn |
| sig | Chữ ký RSA của ID người gửi |
| signed\_info | Bản băm gốc trước khi ký, để kiểm tra lại |
| encrypted\_des\_key | Khóa DES đã được mã hóa bằng RSA |
| sender\_id | Tên hoặc ID của người gửi |
| sender\_pub | Khóa công khai RSA để xác thực chữ ký |

## **3.4. Phân tích chi tiết từng bước**

**Bước 1: Handshake và trao đổi khóa**

* Người gửi gửi "Hello!" đến người nhận để khởi động kết nối.
* Người nhận phản hồi "Ready!" nếu chấp nhận kết nối.
* Hai bên trao đổi khóa công khai RSA.
  + Người gửi gửi sender\_pub.
  + Người nhận gửi lại receiver\_pub.

Sau bước này, cả hai bên đều có public key của nhau.

**Bước 2: Ký số và trao khóa DES**

* Người gửi:
  + Tạo ID người gửi (ví dụ: NguyenVanA)
  + Tính SHA-256(ID)
  + Ký bản băm này bằng **RSA private key** của mình ⇒ tạo sig
  + Gửi bản băm gốc (signed\_info) kèm theo.
* Đồng thời, người gửi:
  + Sinh một khóa DES 8 byte ngẫu nhiên.
  + Mã hóa khóa DES bằng **public key RSA của người nhận** ⇒ encrypted\_des\_key.

Người nhận có thể dùng private key của mình để giải mã khóa DES.

**Bước 3: Mã hóa nội dung & kiểm tra toàn vẹn**

* Tin nhắn văn bản được mã hóa bằng thuật toán **DES-CFB** sử dụng khóa DES và IV.
* Tính toán giá trị **SHA-256** của ciphertext.
* Các trường cipher và hash được đưa vào gói tin.

Điều này giúp người nhận kiểm tra xem dữ liệu có bị thay đổi hay không.

**Bước 4: Người nhận xử lý**

* Sau khi nhận được gói tin:
  + Giải mã khóa DES bằng private key RSA.
  + Kiểm tra SHA-256(ciphertext) có trùng với trường hash không.
  + Kiểm tra chữ ký số (sig) bằng cách:
    - Băm lại sender\_id bằng SHA-256
    - So sánh với signed\_info
    - Dùng public key của người gửi để xác minh chữ ký.
* **Nếu hợp lệ:** giải mã ciphertext bằng DES-CFB → hiển thị tin nhắn → gửi phản hồi ACK.
* **Nếu không hợp lệ:** hiển thị lỗi → gửi phản hồi NACK kèm lý do.

## **3.5. Một số điểm bảo mật đáng chú ý**

* **Bảo mật khóa đối xứng (DES key)**: không gửi trực tiếp, mà mã hóa bằng RSA ⇒ chỉ người nhận giải được.
* **Bảo mật danh tính**: dùng chữ ký số để xác minh người gửi thực sự là ai.
* **Toàn vẹn dữ liệu**: kiểm tra hash để chống sửa đổi trong quá trình truyền.
* **Đảm bảo dữ liệu không lặp**: IV ngẫu nhiên giúp ciphertext mỗi lần khác nhau, ngay cả khi gửi cùng một nội dung.

# **CHƯƠNG 4: CÀI ĐẶT HỆ THỐNG**

## **4.1. Ngôn ngữ và công cụ lập trình**

Đề tài được hiện thực bằng **ngôn ngữ Python 3**, kết hợp với một số thư viện hỗ trợ mạnh về **mã hóa, giao diện, và xử lý mạng**. Python được lựa chọn vì:

* Hỗ trợ thư viện mã hóa đa dạng và mạnh mẽ (pycryptodome, hashlib)
* Dễ tạo giao diện người dùng đơn giản với tkinter
* Có sẵn thư viện socket để xử lý mạng TCP

**Các thư viện chính sử dụng:**

| **Thư viện** | **Vai trò** |
| --- | --- |
| socket | Kết nối TCP giữa client và server |
| tkinter | Giao diện đồ họa người dùng (GUI) |
| base64 | Mã hóa/giải mã dữ liệu nhị phân dưới dạng chuỗi |
| Crypto.Cipher.DES | Mã hóa nội dung tin nhắn (DES-CFB) |
| Crypto.PublicKey.RSA | Sinh và sử dụng khóa RSA |
| Crypto.Signature.pkcs1\_15 | Ký và xác minh chữ ký số |
| Crypto.Hash.SHA256 | Tính toán giá trị băm SHA-256 |
| json | Xử lý dữ liệu gói tin theo định dạng JSON |

Để cài đặt thư viện mã hóa, nhóm sử dụng lệnh:

***pip install pycryptodome***

## **4.2. Cấu trúc thư mục và file nguồn**

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, thiết kế

Nội dung do AI tạo ra có thể không chính xác.

* **generate\_keys\_socket.py:** Sinh cặp khóa RSA cho người gửi và người nhận
* **server\_socket.py:** Giao diện và xử lý phía người nhận (Receiver)
* **client\_socket.py:** Giao diện và xử lý phía người gửi (Sender)
* **common.py:** Một số hàm xử lý dùng chung (nếu có)
* **receiver\_private.pem:** Khóa riêng của người nhận
* **sender\_private.pem:** Khóa riêng của người gửi

## **4.3. Tạo khóa RSA (generate\_keys\_socket.py)**

Trước khi chạy hệ thống, nhóm cần sinh cặp khóa RSA cho hai bên:

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.

Sau khi chạy file, 2 khóa sẽ được tạo:

* receiver\_private.pem: chứa cả khóa công khai và riêng của người nhận
* sender\_private.pem: tương tự cho người gửi

Cả hai file này **cần bảo mật**, chỉ chia sẻ **khóa công khai** cho đối phương.

## **4.4. Giao diện người gửi (client\_socket.py)**

Phần gửi tin nhắn được xây dựng bằng Python, sử dụng thư viện tkinter để tạo giao diện đồ họa. Giao diện bao gồm các thành phần nhập liệu như: ID người gửi, tin nhắn cần mã hóa, và các nút thao tác như *"Handshake"*, *"Mã hóa – Ký – Gửi"*.

### ***4.4.1. Handshake – Kết nối và trao đổi khóa công khai RSA***

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm

Nội dung do AI tạo ra có thể không chính xác.

Khi nhấn nút **“Handshake”**, client sẽ gửi chuỗi "Hello!" tới server qua socket TCP. Nếu server phản hồi "Ready!", client tiếp tục gửi khóa công khai RSA của mình, rồi nhận lại khóa công khai của server. Hai bên nhờ đó có thể mã hóa và xác thực lẫn nhau.

### ***4.4.2. Mã hóa tin nhắn với DES và tạo chữ ký RSA***

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.

Sau khi handshake thành công, người gửi sẽ:

* Sinh một khóa DES 8 byte bằng hàm get\_random\_bytes().
* Mã hóa nội dung tin nhắn bằng thuật toán DES ở chế độ CFB.
* Tạo hash của ID người gửi bằng SHA-256.
* Ký số hash này bằng private key RSA của mình.

### ***4.4.3. Tạo gói tin JSON và gửi đi***

Sau khi mã hóa và ký số xong, chương trình đóng gói các trường như cipher, iv, hash, signature, sender\_id, public key... thành một gói JSON và gửi tới server.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.

Phản hồi từ server (ACK/NACK) sẽ được hiển thị trực tiếp lên giao diện.

## **4.5. Giao diện người nhận (server\_socket.py)**

Phía người nhận cũng sử dụng tkinter để xây dựng giao diện hiển thị các thông tin nhận được như ciphertext, chữ ký, ID người gửi, kết quả giải mã và xác thực. Socket server được khởi chạy song song với GUI trong một luồng riêng (threading).

### ***4.5.1. Nhận gói tin và tách dữ liệu***

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.

Server chấp nhận kết nối từ client, nhận chuỗi JSON, sau đó tách ra từng phần để xử lý. Các giá trị nhận được gồm: cipher, iv, signature, sender\_id, public key...

### ***4.5.2. Giải mã khóa DES bằng RSA private key***

Ảnh có chứa văn bản, Phông chữ, ảnh chụp màn hình, hàng

Nội dung do AI tạo ra có thể không chính xác.

Phần encrypted\_des\_key nhận được từ client sẽ được giải mã bằng private key RSA của người nhận. Sau đó, khóa DES này được dùng để giải mã nội dung tin nhắn.

### ***4.5.3. Xác thực chữ ký số bằng khóa công khai của người gửi***

Ảnh có chứa văn bản, Phông chữ, ảnh chụp màn hình, hàng

Nội dung do AI tạo ra có thể không chính xác.

Để đảm bảo tin nhắn thực sự đến từ đúng người gửi, server sẽ:

* Băm lại ID nhận được bằng SHA-256.
* Dùng public key của người gửi để xác thực chữ ký (sig) kèm theo.

### ***4.5.4. Hiển thị kết quả và phản hồi***

Sau khi server nhận được gói tin từ client và phân tích thành công các trường dữ liệu như cipher, iv, hash, sig, sender\_id, server sẽ thực hiện hai bước chính: **giải mã nội dung** và **xác thực chữ ký**.

Nếu cả hai bước đều hợp lệ, nội dung tin nhắn sẽ được giải mã và hiển thị lên giao diện người dùng. Đồng thời, server sẽ gửi một phản hồi "ACK: Đã nhận gói tin" trở lại cho client để xác nhận rằng dữ liệu hợp lệ.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

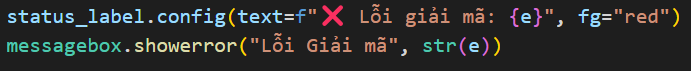
Nội dung do AI tạo ra có thể không chính xác.

Nếu quá trình giải mã gặp lỗi hoặc hàm băm (SHA-256) không khớp với nội dung ciphertext, hệ thống sẽ hiển thị thông báo lỗi trên giao diện và trả về phản hồi "NACK" kèm thông báo cụ thể.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, hàng

Nội dung do AI tạo ra có thể không chính xác.

Trong trường hợp xảy ra lỗi như sai chữ ký, hash không trùng khớp hoặc dữ liệu không hợp lệ, phần server sẽ **không giải mã** được tin nhắn và phản hồi lại "NACK" với lý do chi tiết (ví dụ: "NACK: Lỗi chữ ký" hoặc "NACK: Hash không hợp lệ").



Những phản hồi này giúp client xác định được **trạng thái xử lý của server theo thời gian thực** và xử lý lại nếu cần thiết.

## **4.6. Kết nối và chạy thử**

**Trình tự thực hiện:**

1. **Chạy generate\_keys\_socket.py** để tạo file khóa .pem
2. **Chạy server\_socket.py** → hiện giao diện chờ nhận tin
3. **Chạy client\_socket.py**
4. Nhập ID + nội dung → nhấn **Handshake** → nhấn **Gửi**
5. Server nhận, giải mã, xác thực và phản hồi

# **CHƯƠNG 5: KẾT QUẢ THỰC NGHIỆM**

## **5.1. Môi trường thực nghiệm**

Hệ thống được kiểm thử trong môi trường mạng nội bộ (LAN) trên một máy tính chạy Windows 11 với cấu hình:

* **Ngôn ngữ**: Python 3.10
* **Thư viện mã hóa**: pycryptodome
* **Giao diện**: tkinter
* **Kiểu kết nối**: TCP socket (IP nội bộ: 127.0.0.1)
* **Cặp khóa**: RSA 2048-bit cho mỗi bên
* **Chế độ mã hóa DES**: CFB mode

Cả hai chương trình server\_socket.py và client\_socket.py được chạy độc lập, mỗi chương trình có giao diện đồ họa riêng để quan sát quá trình truyền – nhận.

## **5.2. Tình huống gửi tin thành công**

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Trang web

Nội dung do AI tạo ra có thể không chính xác.

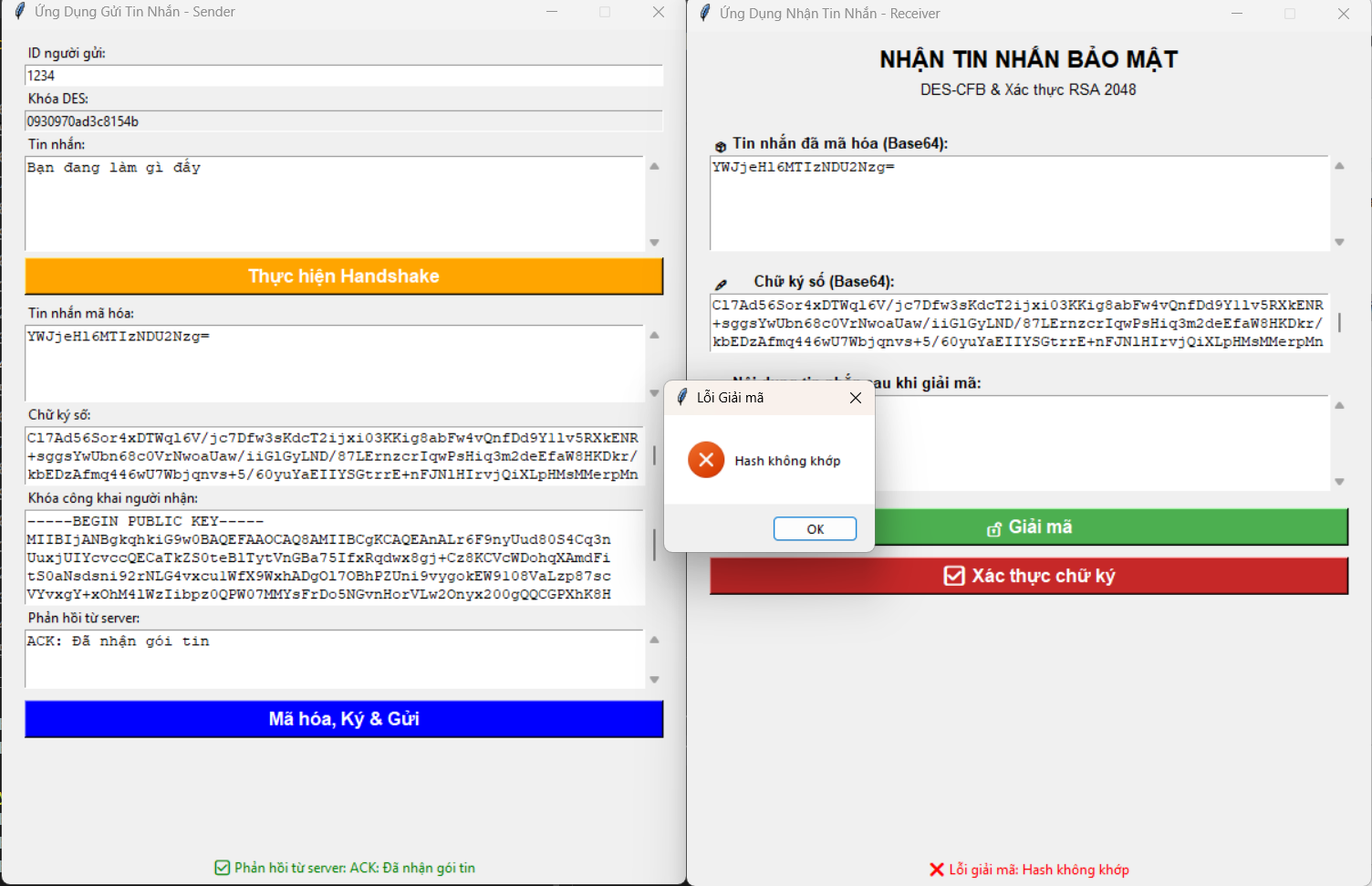
**Mô tả:**

* Người gửi nhập ID và tin nhắn.
* Thực hiện handshake → tạo khóa DES → ký ID → mã hóa tin nhắn.
* Gửi gói tin cho người nhận.

**Kết quả:**

* Người nhận giải mã được tin nhắn.
* Kiểm tra hash và chữ ký đều đúng.
* Phản hồi: ACK: Tin nhắn và chữ ký hợp lệ

## **5.3. Tình huống tin nhắn bị thay đổi**

****

**Mô tả:**

* Sau khi mã hóa, nội dung bị thay đổi (giả sử lỗi đường truyền).
* Hash gửi kèm không khớp với hash tính lại.

**Kết quả:**

* Người nhận phát hiện nội dung không toàn vẹn.
* Không giải mã tin nhắn.
* Phản hồi: NACK: Lỗi giải mã - Hash không khớp

## **5.4. Tình huống chữ ký bị sai**

**Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Trang web

Nội dung do AI tạo ra có thể không chính xác.**

**Mô tả:**

* Dữ liệu đúng nhưng chữ ký số (sig) bị thay đổi.
* Người nhận dùng khóa công khai người gửi để kiểm tra thì không đúng.

**Kết quả:**

* Không xác thực được người gửi.
* Phản hồi: NACK: Lỗi chữ ký - Không hợp lệ

## **5.5. Tình huống thiếu dữ liệu**

**Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Trang web

Nội dung do AI tạo ra có thể không chính xác.**

**Mô tả:**

* Gói tin gửi thiếu trường quan trọng (ví dụ: thiếu encrypted\_des\_key hoặc sender\_pub).

**Kết quả:**

* Người nhận không xử lý được.
* Báo lỗi “Dữ liệu không đầy đủ”.
* Phản hồi: NACK: Gói tin không hợp lệ

## **5.6. Bảng tổng kết các tình huống**

| **Trường hợp** | **Có giải mã được?** | **Hash đúng?** | **Chữ ký đúng?** | **Phản hồi** |
| --- | --- | --- | --- | --- |
| Gửi đúng | ✓ | ✓ | ✓ | ✓ACK |
| Thay đổi nội dung | ✕ | ✕ | - | ✕NACK |
| Sai chữ ký | ✓ (giải mã được) | ✓ | ✕ | ✕NACK |
| Thiếu trường dữ liệu | ✕ | - | - | ✕NACK |

## **5.7. Nhận xét**

Qua các lần thử nghiệm, hệ thống hoạt động ổn định và phản hồi đúng:

* Gửi đúng thì giải mã & xác thực thành công.
* Gửi sai là bị từ chối ngay.
* Hệ thống nhận diện lỗi rõ ràng và báo chi tiết.

# **KẾT LUẬN**

Trong thời đại số hóa hiện nay, việc bảo vệ thông tin trong quá trình truyền tải trở thành yêu cầu bắt buộc, đặc biệt là với những dữ liệu nhạy cảm như tin nhắn, giao dịch, xác thực cá nhân. Qua đề tài “**Ứng dụng bảo mật tin nhắn văn bản với mã hóa DES và xác thực RSA**”, nhóm đã xây dựng thành công một mô hình hệ thống bảo mật tin nhắn đảm bảo ba yếu tố cốt lõi trong bảo mật thông tin: **bí mật nội dung**, **xác thực danh tính**, và **toàn vẹn dữ liệu**.

Cụ thể, nội dung tin nhắn được mã hóa bằng thuật toán **DES ở chế độ CFB** nhằm đảm bảo tính riêng tư. Khóa mã hóa DES được mã hóa thêm một lần nữa bằng **RSA** trước khi gửi, giúp ngăn chặn kẻ thứ ba đọc được dữ liệu ngay cả khi chặn được gói tin. Bên cạnh đó, người gửi sử dụng **RSA và SHA-256** để tạo chữ ký số cho ID của mình, cho phép người nhận xác thực danh tính người gửi một cách chính xác. Ngoài ra, giá trị **băm SHA-256** của ciphertext cũng được gửi kèm để đảm bảo rằng nội dung tin nhắn không bị chỉnh sửa hoặc giả mạo trong quá trình truyền tải.

Thông qua quá trình cài đặt và thực nghiệm, nhóm đã kiểm thử thành công các tình huống:

* Gửi – nhận tin nhắn hợp lệ,
* Tin nhắn bị thay đổi trong quá trình truyền,
* Chữ ký số bị giả mạo,
* Gói tin thiếu dữ liệu cần thiết.

Hệ thống phản hồi đúng với từng tình huống, gửi lại thông báo ACK nếu hợp lệ và NACK kèm lý do nếu phát hiện sai sót. Việc sử dụng **Python kết hợp thư viện PyCryptodome và Tkinter** đã giúp nhóm hiện thực được đầy đủ quy trình bảo mật, đồng thời mang đến một giao diện đơn giản, trực quan và dễ sử dụng.

Bên cạnh những kết quả đạt được, hệ thống vẫn còn một số điểm hạn chế. Thuật toán DES là chuẩn cũ, độ bảo mật thấp hơn các thuật toán hiện đại như AES. Giao thức handshake cũng chỉ mới dừng ở mức mô phỏng đơn giản. Ngoài ra, ứng dụng vẫn chưa hỗ trợ nhiều người dùng hay lưu trữ lịch sử tin nhắn. Tuy nhiên, đây là những vấn đề có thể được khắc phục và mở rộng trong tương lai.

Qua đề tài này, nhóm không chỉ củng cố lại kiến thức về các thuật toán mã hóa và chữ ký số, mà còn rèn luyện kỹ năng xử lý dữ liệu nhị phân, lập trình socket, xây dựng giao diện đồ họa và tư duy logic trong bảo mật hệ thống. Đây là những kỹ năng cực kỳ cần thiết để áp dụng trong các hệ thống thực tế sau này như: xác thực email, bảo mật tin nhắn trong app, chữ ký hợp đồng điện tử, và nhiều ứng dụng khác.

Nhóm xin gửi lời cảm ơn chân thành đến thầy **Trần Đức Thắng-** **giảng viên bộ môn An toàn, bảo mật thông tin** đã tận tình hướng dẫn, tạo điều kiện để nhóm thực hiện và hoàn thiện đề tài. Nhóm cũng cảm ơn các thành viên trong nhóm vì đã cùng nhau nghiên cứu, trao đổi và hoàn thành sản phẩm một cách nghiêm túc và đầy sáng tạo.

# **TÀI LIỆU THAM KHẢO**

1. Trần Đức Dũng (2020), *Giáo trình An toàn và Bảo mật Thông tin*, NXB Bách Khoa Hà Nội.
2. Phạm Hồng Quang (2018), *Mã hóa và Ứng dụng*, Đại học Công nghệ Thông tin TP.HCM.
3. William Stallings (2016), *Cryptography and Network Security – Principles and Practice*, Pearson Education.
4. PyCryptodome Documentation: <https://pycryptodome.readthedocs.io/>
5. Python Socket Programming – <https://docs.python.org/3/library/socket.html>
6. Tkinter GUI Programming – <https://docs.python.org/3/library/tkinter.html>
7. RSA Algorithm – <https://en.wikipedia.org/wiki/RSA_(cryptosystem)>
8. SHA-256 Hash Function – <https://en.wikipedia.org/wiki/SHA-2>
9. Stack Overflow – <https://stackoverflow.com>