**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, BẢO MẬT THÔNG TIN**

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

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

**Sinh viên thực hiện:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Stt** | **Mã sv** | **Họ và tên** | **Lớp** |
| **1** | **1771020535** | **Phan Lưu Phong** | **CNTT 17-03** |
| **2** | **1771020536** | **Trần Văn Phong** | **CNTT 17-03** |
| **3** | **1771020530** | **Lê Hồng Phi** | **CNTT 17-03** |
| **4** | **1771020544** | **Nguyễn Văn Phúc** | **CNTT 17-03** |

**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, BẢO MẬT THÔNG TIN**

**ĐỀ TÀI: ỨNG DỤNG BẢO MẬT TINH 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** | **1771020535** | **Phan Lưu Phong** | **13/11/2005** |  |  |
| **2** | **1771020536** | **Trần Văn Phong** | **08/09/2005** |  |  |
| **3** | **1771020530** | **Lê Hồng Phi** | **06/05/2005** |  |  |
| **3** | **1771020544** | **Nguyễn Văn Phúc** | **17/09/1005** |  |  |

**CÁN BỘ CHẤM THI 1 CÁN BỘ CHẤM THI 2**

**Hà Nội, năm 2025**

**LỜI NÓI ĐẦU**

Trong thời đại công nghệ số phát triển mạnh mẽ, việc bảo mật thông tin trong quá trình truyền tải dữ liệu trở nên vô cùng cấp thiết. Các cuộc tấn công mạng ngày càng tinh vi, gây ra nhiều hậu quả nghiêm trọng nếu dữ liệu nhạy cảm bị rò rỉ hoặc bị giả mạo.

Nhằm giúp sinh viên tiếp cận và hiểu rõ hơn về các phương pháp bảo vệ thông tin cá nhân trong môi trường truyền thông số, nhóm chúng em thực hiện đề tài: "Ứng dụng bảo mật tin nhắn văn bản với mã hóa DES và xác thực RSA". Đề tài tập trung vào việc xây dựng một mô hình truyền tin an toàn giữa hai người dùng, trong đó nội dung tin nhắn được mã hóa bằng thuật toán DES, đi kèm là việc sử dụng RSA để xác thực người gửi và kiểm tra tính toàn vẹn thông qua hàm băm SHA-256.

Trong quá trình thực hiện, nhóm đã áp dụng kiến thức lý thuyết đã học về mật mã học, cũng như thực hành triển khai hệ thống mã hóa – xác thực – kiểm tra toàn vẹn dữ liệu trên ngôn ngữ lập trình Python, tích hợp cùng giao diện người dùng đơn giản.

Nhóm xin chân thành cảm ơn các thầy cô trong bộ môn An toàn và Bảo mật thông tin, đặc biệt là thầy/cô hướng dẫn đã tận tình hỗ trợ và định hướng trong suốt quá trình thực hiện bài tập lớn.

Chúng em kính mong nhận được ý kiến đóng góp để bài làm được hoàn thiện hơn.

**MỤC LỤC**

[CHƯƠNG 1: MỤC TIÊU VÀ YÊU CẦU CỦA ĐỀ TÀI 5](#_Toc202111442)

[1.1 Mục tiêu Mục tiêu 5](#_Toc202111443)

[1.2 Yêu cầu 5](#_Toc202111444)

[1.3 Tiêu chí đánh giá 5](#_Toc202111445)

[1.4 Ý nghĩa thực tiễn của đề tài 5](#_Toc202111446)

[CHƯƠNG 2: PHÂN TÍCH VÀ MÔ HÌNH HỆ THỐNG 7](#_Toc202111447)

[2.1 Mô tả tổng quát 7](#_Toc202111448)

[2.2 Mô hình luồng dữ liệu 7](#_Toc202111449)

[2.3 Các thành phần hệ thống 7](#_Toc202111450)

[2.4 Phân tích yêu cầu bảo mật 7](#_Toc202111451)

[2.5 Sơ đồ use case 8](#_Toc202111452)

[2.6 Kịch bản hoạt động chi tiết 8](#_Toc202111454)

[2.7 Mở rộng hướng triển khai thực tế 11](#_Toc202111456)

[CHƯƠNG 3: MÔ TẢ THUẬT TOÁN VÀ QUY TRÌNH THỰC HIỆN 12](#_Toc202111457)

[3.1 Thuật toán DES 12](#_Toc202111458)

[3.2 Thuật toán RSA 12](#_Toc202111459)

[3.3 SHA-256 12](#_Toc202111460)

[3.4 Quy trình tổng thể (Pseudocode) 12](#_Toc202111461)

[3.5 Định dạng gói tin gửi 13](#_Toc202111462)

[CHƯƠNG 4: THỬ NGHIỆM, ĐÁNH GIÁ VÀ PHÂN TÍCH 14](#_Toc202111463)

[4.1 Cài đặt hệ thống 14](#_Toc202111464)

[4.2 Giao diện người dùng 14](#_Toc202111465)

[4.3 Thử nghiệm thực tế 16](#_Toc202111468)

**MỤC LỤC HÌNH ẢNH**

[Ảnh 1. Sơ đồ Use - Case 8](#_Toc202111453)

[Ảnh 2. Sơ đồ quy trình 9](#_Toc202111455)

[*Ảnh 3. Giao diện của người nhận 13*](#_Toc202111466)

[*Ảnh 4. Giao diện của người gửi 14*](#_Toc202111467)

**BẢNG CÁC TỪ VIẾT TẮT**

|  |  |  |
| --- | --- | --- |
| **STT** | **TỪ VIẾT TẮT** | **VIẾT ĐẦY ĐỦ** |
| **1** | **DES** | **Data Encryption Standard** |
| **2** | **AES** | |  | | --- | |  |   **Advanced Encryption Standard** |
| **3** | **LAN** | |  | | --- | |  |   **Local Area Network** |
| **4** | **2FA** | |  | | --- | |  |   **Two-Factor Authentication** |
| **5** | **SHA-256** | **Secure Hash Algorithm 256-bit** |
| **6** | **CFB** | |  | | --- | |  |   **Cipher Feedback Mode** |
| **7** | **IV** | **Initialization Vector** |
| **8** | **GUI** | |  | | --- | |  |   **Graphical User Interface** |
| **9** | **IDE** | **Integrated Development Environment** |
| **10** | **TCP** | **Transmission Control Protocol** |

# CHƯƠNG 1: MỤC TIÊU VÀ YÊU CẦU CỦA ĐỀ TÀI

## 1.1 Mục tiêu

* Xây dựng hệ thống mô phỏng truyền tin nhắn giữa hai người dùng có áp dụng các cơ chế bảo mật.
* Sử dụng thuật toán mã hóa DES để bảo vệ nội dung tin nhắn.
* Sử dụng chữ ký số RSA để xác thực người gửi.
* Sử dụng hàm băm SHA-256 để kiểm tra tính toàn vẹn của dữ liệu.
* Phát triển giao diện đơn giản để người dùng có thể thao tác gửi – nhận tin nhắn.

## 1.2 Yêu cầu

* Làm việc theo nhóm từ 2–3 sinh viên.
* Ngôn ngữ lập trình tự chọn: Python, Java, hoặc C++.
* Xây dựng ứng dụng đảm bảo các bước: mã hóa – ký số – truyền – kiểm tra – giải mã.
* Báo cáo trình bày đầy đủ các nội dung: phân tích bài toán, mô tả thuật toán, mã nguồn, thử nghiệm và đánh giá.
* Khuyến khích xây dựng giao diện đồ họa và đăng tải mã nguồn công khai.

## 1.3 Tiêu chí đánh giá

* Tính chính xác của thuật toán và quy trình.
* Giao diện thân thiện, dễ sử dụng.
* Báo cáo rõ ràng, khoa học, có thử nghiệm và phân tích đánh giá.

## 1.4 Ý nghĩa thực tiễn của đề tài

* Cung cấp giải pháp mô phỏng bảo mật truyền thông đơn giản nhưng hiệu quả.
* Làm nền tảng để phát triển các ứng dụng chat bảo mật chuyên nghiệp hơn trong tương lai.
* Góp phần nâng cao nhận thức về an toàn thông tin trong sinh viên ngành CNTT.

## 1.5. Mô tả bài toán

Xây dựng một **ứng dụng gửi – nhận tin nhắn văn bản** giữa hai bên qua mạng, có khả năng:

* **Mã hóa** tin nhắn để đảm bảo **chỉ người nhận mới đọc được nội dung**
* **Xác minh danh tính người gửi**, tránh giả mạo.
* **Đảm bảo tin nhắn không bị thay đổi** trong quá trình truyền tải.

Để giải quyết bài toán này, nhóm sử dụng các kỹ thuật mật mã học như:

* **Mã hóa đối xứng DES** để mã hóa nội dung tin nhắn.
* **Thuật toán băm SHA-256** để đảm bảo toàn vẹn bản mã.
* **Hệ mật mã khóa công khai RSA** để:
  + Mã hóa khóa phiên (DES key) khi trao đổi.
  + Tạo chữ ký số đảm bảo xác thực.

**Mục tiêu của bài toán:**

1. **Hiện thực hóa quá trình bảo mật**: Kết hợp DES, SHA-256 và RSA trong một luồng xử lý thống nhất.
2. **Xây dựng giao diện người dùng**: Cho phép nhập, gửi và nhận tin nhắn dễ dàng.
3. **Mô phỏng quá trình truyền tin an toàn**: Trên nền tảng TCP socket, mô phỏng gửi – nhận như môi trường mạng thực tế.

# ****CHƯƠNG 2: PHÂN TÍCH VÀ MÔ HÌNH HỆ THỐNG****

**2.1 Mô tả tổng quát**

* Hệ thống bao gồm hai bên: người gửi và người nhận.
* Người gửi nhập một tin nhắn bất kỳ, hệ thống sẽ tiến hành:
* Mã hóa nội dung bằng DES (CFB).
* Tính toán hàm băm SHA-256 từ bản mã (ciphertext).
* Ký bản mã bằng RSA (2048-bit, PSS, SHA-256).
* Gửi gói tin chứa: bản mã (cipher), hash, chữ ký số (signature), IV, và khóa công khai (public key).
* Người nhận sẽ:
* Kiểm tra chữ ký số bằng public key để xác thực người gửi.
* Kiểm tra hash để đảm bảo toàn vẹn dữ liệu.
* Giải mã bản mã bằng DES để khôi phục tin nhắn gốc.

**2.2 Mô hình luồng dữ liệu**

* Module mã hóa DES: DES với chế độ CFB để đảm bảo bảo mật cao hơn ECB.
* Module ký số RSA: Sử dụng RSA 2048-bit với padding PSS và SHA-256.
* Module hàm băm SHA-256: Đảm bảo tính toàn vẹn của bản mã.
* Module giao tiếp: sửdụng socket TCP.

Giao diện người dùng: GUI bằng Tkinter.

**2.3 Các thành phần hệ thống**

* Module mã hóa DES
* Module tạo và xác minh chữ ký số RSA
* Module hash SHA-256
* Module giao tiếp gửi – nhận (giả lập hoặc dùng socket)
* Giao diện người dùng (GUI tkinter)

**2.4 Phân tích yêu cầu bảo mật**

* Tính bí mật: chỉ người nhận có khóa mới giải mã được.
* Tính xác thực: chữ ký số đảm bảo danh tính người gửi.
* Tính toàn vẹn: SHA-256 đảm bảo dữ liệu không bị chỉnh sửa.

**2.5 Sơ đồ use case**

Ảnh có chứa hàng, biểu đồ, văn bản, Phông chữ

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

Ảnh 1. Sơ đồ Use - Case

**2.6 Kịch bản hoạt động chi tiết**

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

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

Ảnh 2. Sơ đồ quy trình

1. Người gửi khởi tạo ứng dụng và tạo cặp khóa RSA.
2. Kết nối socket đến người nhận, bắt tay bằng thông điệp "Hello!"
3. Gửi khóa công khai.
4. Sinh khóa DES, mã hóa và ký.
5. Gửi gói dữ liệu (JSON) gồm cipher, hash, signature, IV.
6. Người nhận xác minh chữ ký.
7. So sánh hash.
8. Giải mã nội dung DES và hiển thị

**2.7 Luồng xử lí chi tiết**

**1. Khởi tạo hệ thống (Trước khi gửi)**

Người gửi:

* Sinh khóa RSA (private\_key, public\_key)
* Sinh khóa DES ngẫu nhiên (8 byte)
* Kết nối socket tới người nhận (TCP)

Người nhận:

* Khởi tạo socket lắng nghe kết nối
* Sinh khóa RSA (private\_key, public\_key)

**2. Giai đoạn Handshake & Trao đổi khóa**

1. Người gửi gửi chuỗi "Hello!" để bắt đầu handshake
2. Người nhận phản hồi "Ready!" nếu handshake hợp lệ
3. Người gửi nhận public key RSA từ người nhận
4. Người gửi:
   * Mã hóa khóa DES bằng public key nhận được (RSA + OAEP + SHA-256)
   * Ký định danh "sender\_id" bằng khóa riêng RSA (RSA-PSS + SHA-256)
   * Gửi gói tin xác thực gồm:
     + encrypted\_des\_key: khóa DES đã mã hóa
     + signed\_info: chữ ký định danh "sender\_id"
5. Người gửi tiếp tục gửi public key của chính mình
6. Người nhận:
   * Xác thực chữ ký "signed\_info" bằng public key của người gửi
   * Nếu hợp lệ: Giải mã được khóa DES bằng private\_key của mình
   * → Bắt đầu sẵn sàng nhận tin nhắn bảo mật

**3. Gửi và nhận tin nhắn bảo mật**

Người gửi:

1. Nhập tin nhắn văn bản
2. Mã hóa bằng DES-CFB:
   * Tạo IV ngẫu nhiên
   * Mã hóa văn bản → ciphertext
3. Tính hash = SHA256(ciphertext)
4. Ký ciphertext bằng RSA-PSS → signature
5. Đóng gói payload gồm:
   * cipher: bản mã (Base64)
   * iv: vector khởi tạo (Base64)
   * hash: giá trị hash SHA256 (hex)
   * sig: chữ ký số (Base64)
6. Gửi payload tới người nhận

Người nhận:

1. Nhận payload, giải mã JSON
2. Giải mã base64 → các thành phần nhị phân
3. Kiểm tra hash:
   * Tính lại SHA256(ciphertext)
   * So sánh với hash nhận được
   * Nếu không khớp → báo lỗi toàn vẹn
4. Xác minh signature:
   * Dùng public key người gửi
   * Nếu không hợp lệ → báo lỗi xác thực
5. Giải mã DES (CFB) → hiện tin nhắn gốc lên giao diện
6. Ghi log hoạt động vào nhật ký

**4. Kết thúc phiên**

* Khi một bên đóng ứng dụng, socket tự động ngắt.
* Có thể bổ sung nút “Ngắt kết nối” nếu cần rõ ràng hơn.

**2.8 Mở rộng hướng triển khai thực tế**

* Hỗ trợ truyền file thay vì chỉ văn bản.
* Sử dụng AES thay DES.
* Bổ sung xác thực người dùng bằng mật khẩu.
* Lưu lịch sử chat theo phiên.
* Kết nối mạng LAN/Internet thay vì nội bộ.

# CHƯƠNG 3: MÔ TẢ THUẬT TOÁN VÀ QUY TRÌNH THỰC HIỆN

## 3.1 Thuật toán DES

DES là một thuật toán mã hóa khối với khóa dài 56 bit, hoạt động trên các khối 64 bit. Trong đề tài này, chế độ mã hóa CFB (Cipher Feedback Mode) được sử dụng để đảm bảo bảo mật cao hơn và có khả năng xử lý dữ liệu không cần chia khối rõ ràng.

* Khóa DES được sinh ngẫu nhiên (8 byte).
* IV (vector khởi tạo) cũng được tạo ngẫu nhiên mỗi lần gửi.
* Dữ liệu được mã hóa bằng DES với mode CFB và IV tương ứng.

## 3.2 Thuật toán RSA

* RSA là một thuật toán mã hóa bất đối xứng:
* Người gửi và người nhận đều tạo cặp khóa RSA 2048-bit (private/public key).
* Public key được gửi cho phía đối phương để mã hóa hoặc xác minh.
* Chữ ký số được tạo bằng private key (RSA + PSS + SHA-256).
* Dữ liệu được mã hóa bằng khóa công khai và giải mã bằng khóa riêng.
* Trong đề tài, RSA dùng để:
  + Trao đổi khóa DES qua OAEP + SHA-256.
  + Ký bản mã DES để đảm bảo xác thực.

## 3.3 SHA-256

SHA-256 là một hàm băm mã hóa một chiều có độ dài đầu ra 256 bit. Tính chất không thể đảo ngược và cực kỳ nhạy cảm với thay đổi đầu vào giúp SHA-256 trở thành công cụ quan trọng để kiểm tra toàn vẹn dữ liệu.

* Mỗi khi tin nhắn được mã hóa, hệ thống sẽ tính toán SHA-256 của ciphertext.
* Phía người nhận tính lại và so sánh với hash nhận được để phát hiện chỉnh sửa.

## 3.4 Quy trình tổng thể (Pseudocode)

Người gửi:

1. Nhập tin nhắn.
2. Sinh khóa RSA, khóa DES và IV.
3. Mã hóa tin nhắn bằng DES-CFB.
4. Băm bản mã bằng SHA-256.
5. Ký bản mã bằng RSA.
6. Gửi JSON: {cipher, iv, hash, signature, public\_key}.

Người nhận:

1. Nhận JSON và khóa công khai.
2. Xác minh chữ ký RSA.
3. So sánh hash SHA-256.
4. Giải mã bằng DES-CFB và hiển thị kết quả.

## 3.5 Định dạng gói tin gửi

* cipher: bản mã hóa bằng DES (base64)
* iv: vector khởi tạo (base64)
* hash: SHA-256 của bản mã (hex string)
* signature: chữ ký số RSA của bản mã (base64)
* public\_key: khóa công khai người gửi (PEM)

# CHƯƠNG 4: THỬ NGHIỆM, ĐÁNH GIÁ VÀ PHÂN TÍCH

## 4.1 Cài đặt hệ thống

* Ngôn ngữ: Python 3.10
* IDE sử dụng: VS Code / PyCharm
* Giao tiếp mạng: socket TCP
* Thư viện sử dụng:
* pycryptodome: hỗ trợ DES, RSA, SHA-256
* tkinter: giao diện người dùng
* base64, json, socket, hashlib

## 4.2 Giao diện người dùng

Hệ thống gồm 2 phần mềm riêng biệt:

* Ứng dụng **Người gửi**: có khung nhập tin nhắn, nút kết nối và nút gửi.
* Ứng dụng **Người nhận**: có nút lắng nghe, vùng hiển thị tin nhắn và nhật ký hệ thống.

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

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

### Ảnh 3. Giao diện của người nhận

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

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

### Ảnh 4. Giao diện của người gửi

## 4.3 Thử nghiệm thực tế

* Tin nhắn ngắn: "Hello World"
  + Thời gian xử lý: < 0.5 giây
  + Gửi – nhận thành công, nội dung hiển thị chính xác
* Tin nhắn dài 1000 ký tự:
  + Thời gian xử lý mã hóa + ký + gửi ~ 2.5 giây
  + Người nhận giải mã đúng, xác minh chữ ký thành công
* Thử sửa bản mã khi gửi:
  + Hệ thống báo "Hash không khớp"
* Dùng public key giả để xác minh:
  + Hệ thống báo "Chữ ký không hợp lệ"

**4.4** **Kiểm thử kiểm tra toàn vẹn (SHA-256)**

* Tạo bản mã rồi tính SHA-256 → Gửi hash đi cùng
* Người nhận tính lại hash từ bản mã → nếu trùng thì OK, sai thì cảnh báo
* Test thay 1 ký tự → hash khác hoàn toàn → hệ thống báo lỗi toàn vẹn

**4.5 Kiểm thử xác thực chữ ký số RSA**

* Gửi đúng public key: hệ thống xác minh OK
* Thay đổi chữ ký → xác minh sai → hệ thống báo lỗi
* Gửi khóa giả mạo → hệ thống không chấp nhận

**4.6 Phân tích hiệu suất**

* Thời gian trao đổi khóa RSA 2048-bit: ~200-400ms
* Thời gian mã hóa DES cho tin nhắn 1000 ký tự: ~100ms
* Tổng vòng đời gửi–nhận 1 tin nhắn: 1.2 – 3.0 giây

**4.7 Nhận xét**

* Hệ thống hoạt động ổn định, đúng chức năng
* Giao diện đơn giản nhưng rõ ràng
* Bảo mật tin nhắn đạt 3 yêu cầu chính: bí mật, toàn vẹn, xác thực
* Thích hợp làm nền tảng nâng cấp thành app chat bảo mật chuyên nghiệp

## 4.8 Code của 2 file thử nhiệm

Có thể tham khảo từ đây: https://github.com/PhanPhong13112005/FIT4012\_Project.git

Dưới đây là đoạn mã nguồn của hệ thống truyền tin an toàn sử dụng Python, với các chức năng mã hóa DES, ký số RSA và kiểm tra toàn vẹn bằng SHA-256.

* Receiver (Nguời nhận):

import tkinter as tk

from tkinter import scrolledtext, messagebox

import socket

import json

import base64

from Crypto.Cipher import DES

from hashlib import sha256

from cryptography.hazmat.primitives.asymmetric import rsa, padding

from cryptography.hazmat.primitives import serialization, hashes

from cryptography.hazmat.primitives.asymmetric.padding import PSS, MGF1

from cryptography.exceptions import InvalidSignature

HOST = 'localhost'

PORT = 23456

# ==================== PHẦN CỦA NGUYỄN VĂN PHÚC ====================

def generate\_rsa\_key\_pair():

private\_key = rsa.generate\_private\_key(public\_exponent=65537, key\_size=2048)

return private\_key, private\_key.public\_key()

def verify\_signature(public\_key, signature: bytes, data: bytes):

try:

public\_key.verify(

signature,

data,

PSS(mgf=MGF1(hashes.SHA256()), salt\_length=PSS.MAX\_LENGTH),

hashes.SHA256()

)

return True

except InvalidSignature:

return False

def serialize\_public\_key(public\_key):

return public\_key.public\_bytes(

encoding=serialization.Encoding.PEM,

format=serialization.PublicFormat.SubjectPublicKeyInfo

)

# ==================== END PHÚC ====================

# ==================== PHẦN CỦA TRẦN VĂN PHONG ====================

def decrypt\_des(des\_key, iv, ciphertext: bytes):

cipher = DES.new(des\_key, DES.MODE\_CFB, iv)

return cipher.decrypt(ciphertext)

# ==================== END PHONG ====================

class SecureMessengerReceiver:

def \_\_init\_\_(self, root):

self.root = root

self.root.title("Secure Messenger - Người nhận")

self.setup\_ui()

# Khởi tạo RSA

self.receiver\_private\_key, self.receiver\_public\_key = generate\_rsa\_key\_pair()

self.des\_key = None

self.sender\_public\_key = None

self.server\_socket = None

self.client\_socket = None

def setup\_ui(self):

# ==================== PHẦN CỦA PHAN LƯU PHONG ====================

# Frame trạng thái

status\_frame = tk.LabelFrame(self.root, text="Trạng thái", padx=10, pady=10)

status\_frame.pack(padx=10, pady=5, fill="x")

tk.Button(status\_frame, text="Bắt đầu lắng nghe", command=self.start\_server).pack(side="left", padx=5)

self.status\_label = tk.Label(status\_frame, text="❌ Chưa kết nối", fg="red")

self.status\_label.pack(side="left", padx=10)

# Frame tin nhắn

msg\_frame = tk.LabelFrame(self.root, text="Tin nhắn nhận được", padx=10, pady=10)

msg\_frame.pack(padx=10, pady=5, fill="both", expand=True)

self.message\_display = scrolledtext.ScrolledText(msg\_frame, height=10, state="disabled")

self.message\_display.pack(fill="both", expand=True)

# Frame log

log\_frame = tk.LabelFrame(self.root, text="Nhật ký hoạt động", padx=10, pady=10)

log\_frame.pack(padx=10, pady=5, fill="both", expand=True)

self.log\_area = scrolledtext.ScrolledText(log\_frame, height=10, state="disabled")

self.log\_area.pack(fill="both", expand=True)

# ==================== END PHONG ====================

def log(self, message):

self.log\_area.config(state="normal")

self.log\_area.insert("end", message + "\n")

self.log\_area.config(state="disabled")

self.log\_area.see("end")

def display\_message(self, message):

self.message\_display.config(state="normal")

self.message\_display.insert("end", message + "\n")

self.message\_display.config(state="disabled")

self.message\_display.see("end")

def start\_server(self):

try:

# ==================== PHẦN CỦA LÊ HỒNG PHI ====================

self.server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

self.server\_socket.bind((HOST, PORT))

self.server\_socket.listen(1)

self.status\_label.config(text="🟡 Đang chờ kết nối...", fg="orange")

self.log(f"🧭 Đang lắng nghe tại {HOST}:{PORT}...")

# Chạy trong thread riêng để không block GUI

import threading

threading.Thread(target=self.accept\_connection, daemon=True).start()

# ==================== END PHI ====================

except Exception as e:

messagebox.showerror("Lỗi", f"Khởi động server thất bại: {str(e)}")

self.log(f"❌ Lỗi khởi động server: {str(e)}")

def accept\_connection(self):

try:

self.client\_socket, addr = self.server\_socket.accept()

self.root.after(0, lambda: self.status\_label.config(text=f"✅ Đã kết nối với {addr[0]}", fg="green"))

self.root.after(0, lambda: self.log(f"🔗 Đã kết nối với {addr}"))

# ==================== PHẦN CỦA LÊ HỒNG PHI ====================

hello = self.client\_socket.recv(1024).decode()

if hello != "Hello!":

raise ConnectionError("Handshake failed")

self.client\_socket.sendall(b"Ready!")

self.root.after(0, lambda: self.log("🤝 Handshake thành công"))

# Gửi public key RSA

self.client\_socket.sendall(serialize\_public\_key(self.receiver\_public\_key))

# Nhận thông tin xác thực

secure\_data = self.client\_socket.recv(4096)

auth\_packet = json.loads(secure\_data.decode())

encrypted\_des\_key = base64.b64decode(auth\_packet["encrypted\_des\_key"])

signed\_info = base64.b64decode(auth\_packet["signed\_info"])

# Nhận public key của sender

sender\_pub\_bytes = self.client\_socket.recv(4096)

self.sender\_public\_key = serialization.load\_pem\_public\_key(sender\_pub\_bytes)

# ==================== PHẦN CỦA NGUYỄN VĂN PHÚC ====================

if not verify\_signature(self.sender\_public\_key, signed\_info, b"sender\_id"):

raise ValueError("Chữ ký định danh không hợp lệ")

self.des\_key = self.receiver\_private\_key.decrypt(

encrypted\_des\_key,

padding.OAEP(mgf=padding.MGF1(algorithm=hashes.SHA256()),

algorithm=hashes.SHA256(),

label=None)

)

self.root.after(0, lambda: self.log("🔑 Đã nhận khóa DES an toàn"))

# ==================== END PHÚC ====================

# Bắt đầu lắng nghe tin nhắn

self.receive\_messages()

except Exception as e:

self.root.after(0, lambda: messagebox.showerror("Lỗi", f"Xử lý kết nối thất bại: {str(e)}"))

self.root.after(0, lambda: self.log(f"❌ Lỗi kết nối: {str(e)}"))

def receive\_messages(self):

try:

while True:

data = self.client\_socket.recv(4096)

if not data:

break

# ==================== PHẦN CỦA TRẦN VĂN PHONG & NGUYỄN VĂN PHÚC ====================

payload = json.loads(data.decode())

ciphertext = base64.b64decode(payload["cipher"])

iv = base64.b64decode(payload["iv"])

msg\_hash = payload["hash"]

signature = base64.b64decode(payload["sig"])

# Kiểm tra hash

if sha256(ciphertext).hexdigest() != msg\_hash:

self.root.after(0, lambda: self.log("❌ Hash không khớp! Dữ liệu có thể bị sửa."))

continue

# Xác thực chữ ký

if not verify\_signature(self.sender\_public\_key, signature, ciphertext):

self.root.after(0, lambda: self.log("❌ Chữ ký không hợp lệ!"))

continue

# Giải mã

plaintext = decrypt\_des(self.des\_key, iv, ciphertext)

self.root.after(0, lambda: self.display\_message(f"Người gửi: {plaintext.decode()}"))

self.root.after(0, lambda: self.log(f"📩 Đã nhận tin nhắn bảo mật"))

# ==================== END PHONG & PHÚC ====================

except Exception as e:

self.root.after(0, lambda: self.log(f"❌ Lỗi nhận tin nhắn: {str(e)}"))

if \_\_name\_\_ == "\_\_main\_\_":

root = tk.Tk()

app = SecureMessengerReceiver(root)

root.mainloop()

* Sender (Người gửi):

import tkinter as tk

from tkinter import scrolledtext, messagebox

import socket

import json

import base64

from Crypto.Cipher import DES

from Crypto.Random import get\_random\_bytes

from hashlib import sha256

from cryptography.hazmat.primitives.asymmetric import rsa, padding

from cryptography.hazmat.primitives import serialization, hashes

from cryptography.hazmat.primitives.asymmetric.padding import PSS, MGF1

HOST = 'localhost'

PORT = 23456

# ==================== PHẦN CỦA NGUYỄN VĂN PHÚC ====================

def generate\_rsa\_key\_pair():

private\_key = rsa.generate\_private\_key(public\_exponent=65537, key\_size=2048)

return private\_key, private\_key.public\_key()

def sign\_data(private\_key, data: bytes):

return private\_key.sign(

data,

PSS(mgf=MGF1(hashes.SHA256()), salt\_length=PSS.MAX\_LENGTH),

hashes.SHA256()

)

def serialize\_public\_key(public\_key):

return public\_key.public\_bytes(

encoding=serialization.Encoding.PEM,

format=serialization.PublicFormat.SubjectPublicKeyInfo

)

# ==================== END PHÚC ====================

# ==================== PHẦN CỦA TRẦN VĂN PHONG ====================

def encrypt\_des(des\_key, plaintext: bytes):

iv = get\_random\_bytes(8)

cipher = DES.new(des\_key, DES.MODE\_CFB, iv)

ciphertext = cipher.encrypt(plaintext)

return iv, ciphertext

# ==================== END PHONG ====================

class SecureMessengerSender:

def \_\_init\_\_(self, root):

self.root = root

self.root.title("Secure Messenger - Người gửi")

self.setup\_ui()

# Khởi tạo RSA

self.sender\_private\_key, self.sender\_public\_key = generate\_rsa\_key\_pair()

self.des\_key = None

self.receiver\_public\_key = None

self.socket = None

def setup\_ui(self):

# ==================== PHẦN CỦA PHAN LƯU PHONG ====================

# Frame kết nối

conn\_frame = tk.LabelFrame(self.root, text="Kết nối", padx=10, pady=10)

conn\_frame.pack(padx=10, pady=5, fill="x")

tk.Button(conn\_frame, text="Kết nối", command=self.connect).pack(side="left", padx=5)

self.conn\_status = tk.Label(conn\_frame, text="❌ Chưa kết nối", fg="red")

self.conn\_status.pack(side="left", padx=10)

# Frame tin nhắn

msg\_frame = tk.LabelFrame(self.root, text="Tin nhắn", padx=10, pady=10)

msg\_frame.pack(padx=10, pady=5, fill="both", expand=True)

self.message\_entry = scrolledtext.ScrolledText(msg\_frame, height=10)

self.message\_entry.pack(fill="both", expand=True)

tk.Button(msg\_frame, text="Gửi tin nhắn", command=self.send\_message).pack(pady=5)

# Frame log

log\_frame = tk.LabelFrame(self.root, text="Nhật ký hoạt động", padx=10, pady=10)

log\_frame.pack(padx=10, pady=5, fill="both", expand=True)

self.log\_area = scrolledtext.ScrolledText(log\_frame, height=10, state="disabled")

self.log\_area.pack(fill="both", expand=True)

# ==================== END PHONG ====================

def log(self, message):

self.log\_area.config(state="normal")

self.log\_area.insert("end", message + "\n")

self.log\_area.config(state="disabled")

self.log\_area.see("end")

def connect(self):

try:

# ==================== PHẦN CỦA LÊ HỒNG PHI ====================

self.socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

self.socket.connect((HOST, PORT))

self.socket.sendall(b"Hello!")

if self.socket.recv(1024).decode() != "Ready!":

raise ConnectionError("Handshake failed")

self.log("🤝 Handshake thành công với người nhận")

self.conn\_status.config(text="✅ Đã kết nối", fg="green")

# Nhận public key từ receiver

receiver\_pub\_bytes = self.socket.recv(4096)

self.receiver\_public\_key = serialization.load\_pem\_public\_key(receiver\_pub\_bytes)

# Trao đổi khóa DES

self.des\_key = get\_random\_bytes(8)

encrypted\_des\_key = self.receiver\_public\_key.encrypt(

self.des\_key,

padding.OAEP(mgf=padding.MGF1(algorithm=hashes.SHA256()),

algorithm=hashes.SHA256(),

label=None)

)

signed\_info = sign\_data(self.sender\_private\_key, b"sender\_id")

auth\_packet = {

"signed\_info": base64.b64encode(signed\_info).decode(),

"encrypted\_des\_key": base64.b64encode(encrypted\_des\_key).decode()

}

self.socket.sendall(json.dumps(auth\_packet).encode())

# Gửi public key của sender

self.socket.sendall(serialize\_public\_key(self.sender\_public\_key))

self.log("🔑 Đã trao đổi khóa DES an toàn")

# ==================== END PHI ====================

except Exception as e:

messagebox.showerror("Lỗi", f"Kết nối thất bại: {str(e)}")

self.log(f"❌ Lỗi kết nối: {str(e)}")

def send\_message(self):

if not self.socket:

messagebox.showwarning("Cảnh báo", "Vui lòng kết nối trước khi gửi tin nhắn")

return

message = self.message\_entry.get("1.0", "end-1c").strip()

if not message:

return

try:

# ==================== PHẦN CỦA TRẦN VĂN PHONG ====================

iv, ciphertext = encrypt\_des(self.des\_key, message.encode())

msg\_hash = sha256(ciphertext).hexdigest()

# ==================== END PHONG ====================

# ==================== PHẦN CỦA NGUYỄN VĂN PHÚC ====================

signature = sign\_data(self.sender\_private\_key, ciphertext)

# ==================== END PHÚC ====================

payload = {

"cipher": base64.b64encode(ciphertext).decode(),

"iv": base64.b64encode(iv).decode(),

"hash": msg\_hash,

"sig": base64.b64encode(signature).decode()

}

self.socket.sendall(json.dumps(payload).encode())

self.log(f"✉️ Đã gửi tin nhắn bảo mật: {message}")

self.message\_entry.delete("1.0", "end")

except Exception as e:

messagebox.showerror("Lỗi", f"Gửi tin nhắn thất bại: {str(e)}")

self.log(f"❌ Lỗi khi gửi: {str(e)}")

if \_\_name\_\_ == "\_\_main\_\_":

root = tk.Tk()

app = SecureMessengerSender(root)

root.mainloop()

**PHU LỤC**

|  |  |  |
| --- | --- | --- |
| **Họ và Tên** | **Nhiệm Vụ** | **Chi Tiết Nhiệm Vụ** |
| Lê Hồng Phi | Giao tiếp & gửi nhận | Xử lý handshake ("Hello" ↔ "Ready"), Gửi/nhận dữ liệu giữa người gửi và người nhận |
| Phan Lưu Phong | Giao diện & báo cáo | Làm giao diện gửi/nhận tin nhắn, Tích hợp chức năng và viết báo cáo (hình ảnh, mô tả) |
| Trần Văn Phong | Mã hóa & hash | Mã hóa văn bản bằng DES (CFB mode), Sinh IV và tính SHA-256(ciphertext) |
| Nguyễn Văn Phúc | Chữ ký số | Tạo khóa RSA 2048-bit, Ký dữ liệu và kiểm tra chữ ký |

**KẾT LUẬN**

Hệ thống truyền tin bảo mật sử dụng DES và RSA do nhóm xây dựng có nhiều ưu điểm nổi bật. Cụ thể, hệ thống đáp ứng đầy đủ ba yêu cầu bảo mật cơ bản là: bí mật (thông qua mã hóa DES), xác thực (sử dụng chữ ký số RSA), và toàn vẹn dữ liệu (bằng hàm băm SHA-256). Ngoài ra, việc tích hợp giao diện đồ họa Tkinter giúp người dùng thao tác dễ dàng, theo dõi được trạng thái gửi – nhận – xác minh một cách trực quan. Quá trình trao đổi khóa cũng được mã hóa an toàn qua RSA-OAEP, đảm bảo tin nhắn không bị nghe lén hay sửa đổi trong quá trình truyền.

Tuy nhiên, hệ thống vẫn tồn tại một số hạn chế. Thuật toán DES hiện nay được xem là đã lỗi thời, độ dài khóa ngắn nên dễ bị tấn công brute-force nếu ứng dụng trong môi trường thực tế. Ngoài ra, hệ thống chưa hỗ trợ xác thực người dùng bằng tài khoản hoặc mật khẩu, chưa có tính năng lưu lịch sử trò chuyện hay gửi tệp đính kèm. Giao diện tuy đơn giản nhưng chưa tối ưu thẩm mỹ và trải nghiệm người dùng.

Trong tương lai, hệ thống có thể được phát triển thêm theo nhiều hướng như: thay thế DES bằng AES-128 hoặc AES-256 để tăng độ bảo mật; bổ sung chức năng xác thực đa yếu tố (2FA); lưu trữ lịch sử tin nhắn dưới dạng mã hóa; kết nối mạng LAN/Internet để hoạt động như một ứng dụng chat thực tế; và nâng cấp giao diện bằng các framework như PyQt hoặc web-based GUI để tăng tính chuyên nghiệp và tiện ích cho người dùng cuối.

**DANH MỤC TÀI LIỆU THAM KHẢO**

1. Nguyễn Hữu Lộc (2018), Giáo trình An toàn và Bảo mật Thông tin, NXB Đại học Quốc gia TP.HCM.
2. Trần Đình Long (2016), An toàn mạng máy tính, NXB Bách Khoa Hà Nội.
3. Lê Huy Hoàng (2015), Mật mã học đại cương, NXB Khoa học và Kỹ thuật.