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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | Mã Sinh Viên | Họ và Tên | Ngày Sinh | Lớp |
| 1 | 1771020706 | Trần Anh Tú | 04/05/2005 | CNTT 17-03 |
| 2 | 1771020230 | Nguyễn Xuân Giang | 30/03/2005 | CNTT 17-03 |
| 3 | 1771020709 | Đỗ Thành Tú | 04/02/2005 | CNTT 17-03 |
| 4 | 1771020733 | Dương Anh Tường | 29/06/2005 | CNTT17-03 |

### 

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STT | Mã Sinh Viên | Họ và Tên | Ngày Sinh | Điểm | |
| Bằng Số | Bằng Chữ |
| 1 | 1771020706 | Trần Anh Tú | 04/05/2005 |  |  |
| 2 | 1771020230 | Nguyễn Xuân Giang | 30/03/2005 |  |  |
| 3 | 1771020709 | Đỗ Thành Tú | 04/02/2005 |  |  |
| 4 | 1771020733 | Dương Anh Tường | 29/06/2005 |  |  |

### 

### CÁN BỘ CHẤM THI 1 CÁN BỘ CHẤM THI 2

**Hà Nội, năm 2025**

**LỜI NÓI ĐẦU**

Thế giới hiện đại vận hành trên nền tảng của dữ liệu. Mỗi ngày, hàng tỷ gigabyte thông tin được trao đổi qua mạng Internet, từ những tin nhắn cá nhân, giao dịch tài chính cho đến các tài liệu quan trọng của doanh nghiệp và chính phủ. Tuy nhiên, sự tiện lợi của việc kết nối toàn cầu cũng đi kèm với những rủi ro khôn lường. Dữ liệu trên đường truyền có thể dễ dàng trở thành mục tiêu của các hành vi nghe lén, đánh cắp, chỉnh sửa hoặc giả mạo, gây ra những hậu quả nghiêm trọng và không thể lường trước. Chính vì vậy, việc trang bị kiến thức và kỹ năng để xây dựng các hệ thống truyền tin an toàn đã trở thành một yêu cầu cấp thiết đối với bất kỳ kỹ sư công nghệ nào.

Xuất phát từ thực tiễn đó, bài tập lớn với đề tài "Gửi bài tập chia thành nhiều phần" được thực hiện nhằm mục đích vận dụng những kiến thức nền tảng của học phần Nhập môn An toàn, bảo mật thông tin để giải quyết một bài toán cụ thể. Đề tài tập trung vào việc thiết kế và xây dựng một chương trình mô phỏng quá trình truyền một file dữ liệu lớn qua mạng một cách an toàn. Đây không chỉ là một bài tập ứng dụng lý thuyết, mà còn là cơ hội để sinh viên trực tiếp đối mặt và tìm ra giải pháp cho các thách thức về bảo mật trong thực tế.

Mục tiêu cốt lõi của đề tài là xây dựng một hệ thống đảm bảo được ba trụ cột của an toàn thông tin: Tính bảo mật (Confidentiality), Tính toàn vẹn (Integrity), và Tính xác thực (Authentication). Để đạt được điều này, dự án sẽ triển khai một giao thức bảo mật riêng, kết hợp sức mạnh của các thuật toán mật mã kinh điển: mã hóa đối xứng DES cho tốc độ, mã hóa bất đối xứng RSA cho việc trao đổi khóa và xác thực danh tính, cùng hàm băm SHA-512 để đảm bảo dữ liệu không bị thay đổi trên đường truyền.

Nội dung bài báo cáo sẽ được trình bày một cách có hệ thống, bắt đầu bằng việc phân tích chi tiết yêu cầu của bài toán, sau đó đi sâu vào mô tả thiết kế và quy trình hoạt động của giao thức. Tiếp theo, báo cáo sẽ trình bày các kết quả thử nghiệm, đánh giá hiệu quả của chương trình, và cuối cùng là các phân tích, nhận xét về đặc điểm của các thuật toán đã sử dụng cũng như đề xuất những hướng phát triển tiềm năng trong tương lai.

**MỤC LỤC**

[**CHƯƠNG 1. TỔNG QUAN ĐỀ TÀI** 9](#_Toc202295971)

[**1.1.** **Đặt vấn đề** 9](#_Toc202295972)

[**1.2.** **Phân tích yêu cầu bài toán** 9](#_Toc202295973)

[**1.3.** **Tổng quan về mật mã học** 9](#_Toc202295974)

[**1.4.** **Mã hóa đối xứng và Thuật toán DES** 11](#_Toc202295975)

[**1.5.** **Mã hóa bất đối xứng và Thuật toán RSA** 12](#_Toc202295976)

[**1.6.** **Hàm băm và Thuật toán SHA-512** 13](#_Toc202295977)

[**1.7.** **Chữ ký số** 14](#_Toc202295978)

[**CHƯƠNG 2. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG** 17](#_Toc202295979)

[**2.1.** **Phân tích yêu cầu** 17](#_Toc202295980)

[**2.2.** **Mô hình kiến trúc** 17](#_Toc202295981)

[**2.3.** **Thiết kế luồng xử lý chi tiết** 18](#_Toc202295982)

[**2.3.1.** **Giai đoạn Bắt tay (Handshake): Khởi tạo kết nối** 19](#_Toc202295983)

[**2.3.2.** **Giai đoạn xác thực và trao đổi khóa** 19](#_Toc202295984)

[**2.3.3.** **Giai đoạn Truyền dữ liệu** 20](#_Toc202295985)

[**2.3.4.** **Giai đoạn Xử lý tại phía nhận** 21](#_Toc202295986)

[**2.3.5.** **Giai đoạn phản hồi (ACK/NACK)** 21](#_Toc202295987)

[**CHƯƠNG 3. KẾT QUẢ THỰC NGHIỆM** 23](#_Toc202295988)

[3.1. Môi trường thực nghiệm 23](#_Toc202295989)

[3.2. Mô tả hệ thống 23](#_Toc202295990)

[3.3. Luồng hoạt động tổng thể 25](#_Toc202295991)

[code send\_file(): 26](#_Toc202295992)

[3.4. Giao diện người dùng 27](#_Toc202295993)

[3.5. Kết quả thử nghiệm 29](#_Toc202295994)

[3.6. Phân tích bảo mật 30](#_Toc202295995)

[3.7. Đánh giá tổng quan 30](#_Toc202295996)

**MỤC LỤC HÌNH ẢNH**

[Hình 1:Mã hóa công khai 11](#_Toc202092784)

[Hình 2: Sơ đồ tạo chữ ký số 13](#_Toc202092785)

[Hình 3: Sơ đồ xác định chữ ký số 14](#_Toc202092786)

[Hình 4: Sơ đồ hoạt động của hệ thống chia nhỏ, mã hóa và xác thực bài tập 17](#_Toc202092787)

**CHƯƠNG 1. TỔNG QUAN ĐỀ TÀI**

* 1. **Đặt vấn đề**

Trong bối cảnh trao đổi thông tin qua mạng ngày càng phổ biến, việc đảm bảo an toàn cho dữ liệu truyền đi là một thách thức lớn. Các rủi ro như nghe lén, thay đổi nội dung, hay giả mạo nguồn gửi có thể gây ra những hậu quả nghiêm trọng. Bài toán của Đề tài: Gửi bài tập chia thành nhiều phần đặt ra một tình huống thực tế: xây dựng một hệ thống cho phép một người dùng gửi một file dữ liệu, đặc biệt là các file có dung lượng lớn, qua mạng một cách an toàn bằng cách chia nhỏ file thành nhiều phần để truyền đi.

Đề tài nhằm xây dựng một hệ thống gửi file qua mạng vừa nhanh chóng vừa đảm bảo an toàn thông tin toàn diện về bảo mật, xác thực và toàn vẹn.

* 1. **Phân tích yêu cầu bài toán**

Mã hoá dữ liệu để đảm bảo tính bảo mật (Confidentiality): Toàn bộ nội dung của file phải được mã hóa trước khi gửi lên mạng. Điều này đảm bảo rằng ngay cả khi kẻ tấn công bắt được các gói tin trên đường truyền, chúng cũng không thể đọc và hiểu được nội dung gốc.

Xác thực người dùng để đảm bảo người dùng đúng quyền (Authentication): Hệ thống phải có cơ chế để bên nhận có thể xác minh danh tính của bên gửi. Điều này nhằm chống lại các cuộc tấn công giả mạo, đảm bảo rằng dữ liệu được nhận thực sự đến từ một nguồn hợp pháp và đáng tin cậy.

Kiểm tra tính toàn vẹn của dữ liệu (Integrity): Phía nhận phải có khả năng kiểm tra xem dữ liệu mình nhận được có bị thay đổi (dù là vô tình hay cố ý) so với dữ liệu gốc mà bên gửi đã truyền đi hay không.

* 1. **Tổng quan về mật mã học**

Mật mã học (Cryptography) là ngành khoa học nghiên cứu về các phương pháp truyền đạt thông tin một cách an toàn, trong đó chỉ có những người nhận được ủy quyền mới có thể đọc và xử lý nó. Để hiểu rõ hơn về lĩnh vực này, chúng ta cần nắm vững các khái niệm nền tảng sau:

* Mã hóa (Encryption) : Mã hóa là quá trình biến đổi thông tin từ dạng có thể đọc được (gọi là văn bản rõ hay plaintext) thành một dạng không thể hiểu được (gọi là văn bản mã hóa hay ciphertext). Mục đích chính của mã hóa là để bảo vệ tính bí mật của thông tin. Quá trình này sử dụng một thuật toán mật mã (gọi là cipher) và một hoặc nhiều khóa (key) để thực hiện việc chuyển đổi. Chỉ những ai sở hữu khóa giải mã tương ứng mới có thể đảo ngược quá trình này.
* Giải mã (Decryption) : Giải mã là quá trình ngược lại của mã hóa. Nó biến đổi thông tin từ dạng văn bản mã hóa (ciphertext) trở lại dạng văn bản rõ ban đầu (plaintext). Quá trình giải mã cũng cần sử dụng một thuật toán và một khóa tương ứng. Nếu không có khóa chính xác, việc giải mã gần như là không thể.
* Khóa (Key) : Khóa là một mẩu thông tin (thường là một chuỗi bit, một số hoặc một chuỗi ký tự) đóng vai trò như một tham số đầu vào cho thuật toán mật mã. Khóa sẽ quyết định kết quả đầu ra của quá trình mã hóa và giải mã. Cùng một văn bản rõ và cùng một thuật toán, nhưng nếu sử dụng các khóa khác nhau sẽ tạo ra các văn bản mã hóa hoàn toàn khác nhau.

Độ an toàn của một hệ thống mật mã phụ thuộc rất lớn vào độ mạnh và tính bí mật của khóa. Có hai loại khóa chính:

* Khóa đối xứng (Symmetric Key): Cùng một khóa được sử dụng cho cả quá trình mã hóa và giải mã. Bên gửi và bên nhận phải chia sẻ khóa này một cách bí mật. (Ví dụ: DES, AES).
* Khóa bất đối xứng (Asymmetric Key): Sử dụng một cặp khóa có quan hệ toán học với nhau:
* Khóa công khai (Public Key): Được chia sẻ công khai cho mọi người, dùng để mã hóa dữ liệu.
* Khóa bí mật (Private Key): Được giữ bí mật tuyệt đối bởi chủ sở hữu, dùng để giải mã dữ liệu đã được mã hóa bằng khóa công khai tương ứng. (Ví dụ: RSA).

1. Các mục tiêu của An toàn thông tin : An toàn thông tin không chỉ dừng lại ở việc mã hóa. Một hệ thống được xem là an toàn khi nó đáp ứng được các mục tiêu cốt lõi sau đây, thường được biết đến với tên gọi "Bộ ba CIA" và các mục tiêu mở rộng:

* Tính bảo mật (Confidentiality): Đảm bảo rằng thông tin chỉ có thể được truy cập bởi những người được ủy quyền. Đây là mục tiêu chính mà mã hóa hướng tới, giúp chống lại hành vi nghe lén.
* Tính toàn vẹn (Integrity): Đảm bảo rằng thông tin không bị sửa đổi một cách trái phép trong quá trình lưu trữ hoặc truyền tải. Các cơ chế như hàm băm (hashing) được sử dụng để kiểm tra tính toàn vẹn của dữ liệu.
* Tính sẵn sàng (Availability): Đảm bảo rằng người dùng được ủy quyền có thể truy cập thông tin và các tài nguyên liên quan bất cứ khi nào họ cần. Mục tiêu này chống lại các cuộc tấn công từ chối dịch vụ (Denial of Service - DoS).
* Ngoài bộ ba CIA, còn có hai mục tiêu quan trọng khác:
* Tính xác thực (Authentication): Là quá trình xác minh danh tính của một người dùng, một thiết bị hoặc một hệ thống. Chữ ký số là một ví dụ điển hình của cơ chế xác thực.
* Tính chống chối bỏ (Non-repudiation): Đảm bảo rằng một cá nhân hay thực thể không thể từ chối một hành động mà họ đã thực hiện. Ví dụ, khi một người đã gửi đi một thông điệp có ký số, họ không thể chối bỏ rằng chính mình đã gửi thông điệp đó.
  1. **Mã hóa đối xứng và Thuật toán DES**

Mã hóa đối xứng là phương pháp mật mã sử dụng chỉ một khóa duy nhất cho cả hai quá trình mã hóa và giải mã.

* Nguyên lý: Bên gửi và bên nhận phải chia sẻ chung một khóa bí mật. Bên gửi dùng khóa này để mã hóa thông điệp, và bên nhận dùng chính khóa đó để giải mã.
* Ưu điểm: Tốc độ xử lý rất nhanh, hiệu quả để mã hóa lượng lớn dữ liệu (file, video).
* Thách thức: Việc phân phối khóa bí mật một cách an toàn cho cả hai bên.

DES là một thuật toán mã hóa đối xứng theo khối (block cipher), từng là tiêu chuẩn trong ngành.

* Cấu trúc: Dựa trên Mạng Feistel với 16 vòng lặp phức tạp để xáo trộn dữ liệu.
* Kích thước khối (Block Size): Hoạt động trên các khối dữ liệu cố định 64 bit.
* Kích thước khóa (Key Size): Sử dụng khóa có độ dài hiệu dụng là 56 bit. Chính kích thước khóa nhỏ này đã khiến DES không còn an toàn trước các cuộc tấn công vét cạn hiện đại.
* Vai trò: Mục đích chính của DES là đảm bảo tính bảo mật (Confidentiality) cho nội dung tin nhắn. Mặc dù hiện đã lỗi thời, DES là nền tảng quan trọng cho sự phát triển của các thuật toán mã hóa hiện đại hơn như AES.
  1. **Mã hóa bất đối xứng và Thuật toán RSA**

Mã hóa bất đối xứng (hay mã hóa khóa công khai) sử dụng một cặp khóa bao gồm:

* Khóa Công khai (Public Key): Dùng để mã hóa, được chia sẻ rộng rãi.
* Khóa Bí mật (Private Key): Dùng để giải mã, được chủ sở hữu giữ bí mật.
* Nguyên lý: Ai cũng có thể dùng Khóa công khai của bạn để mã hóa và gửi tin nhắn cho bạn, nhưng chỉ bạn mới có Khóa bí mật tương ứng để giải mã và đọc được tin nhắn đó.
* Ưu điểm: Giải quyết được vấn đề phân phối khóa của mã hóa đối xứng.
* Nhược điểm: Tốc độ xử lý chậm hơn nhiều so với mã hóa đối xứng.

RSA là thuật toán mã hóa bất đối xứng phổ biến nhất, có hai ứng dụng chính:

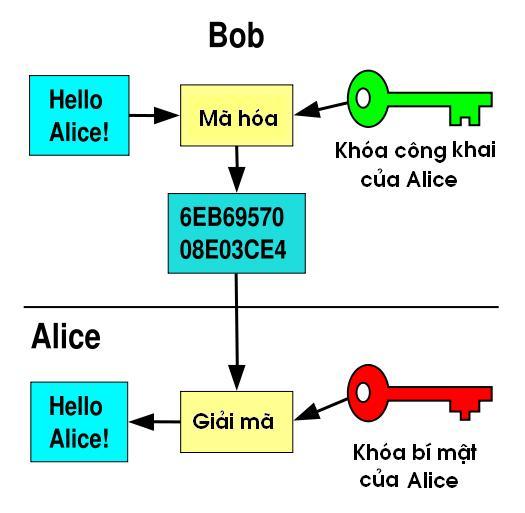
1. Mã hóa (Đảm bảo bí mật):

Mã hóa: Dùng Khóa công khai của người nhận.

Giải mã: Người nhận dùng Khóa bí mật của chính mình.

1. Chữ ký số (Đảm bảo xác thực & toàn vẹn)

* Tạo chữ ký:
* Băm (hash) thông điệp.
* Dùng Khóa bí mật của chính mình để mã hóa giá trị băm đó.
* Xác thực chữ ký:
* Người nhận dùng Khóa công khai của người gửi để giải mã chữ ký (thu được giá trị băm A).
* Tự băm lại thông điệp (thu được giá trị băm B).
* So sánh hash A và hash B. Nếu trùng nhau, chữ ký hợp lệ.



Hình 1:Mã hóa công khai

* 1. **Hàm băm và Thuật toán SHA-512**

Hàm băm là một thuật toán biến đổi dữ liệu có độ dài bất kỳ thành một chuỗi đầu ra có độ dài cố định, gọi là giá trị băm (hash).

Các đặc tính quan trọng:

* Tính một chiều: Dễ dàng tạo ra giá trị băm từ dữ liệu, nhưng không thể tìm lại dữ liệu gốc từ giá trị băm.
* Tính kháng va chạm: Cực kỳ khó để tìm thấy hai khối dữ liệu khác nhau mà lại có cùng một giá trị băm.
* Hiệu ứng thác đổ: Chỉ cần một thay đổi nhỏ ở dữ liệu đầu vào cũng sẽ tạo ra một giá trị băm hoàn toàn khác.

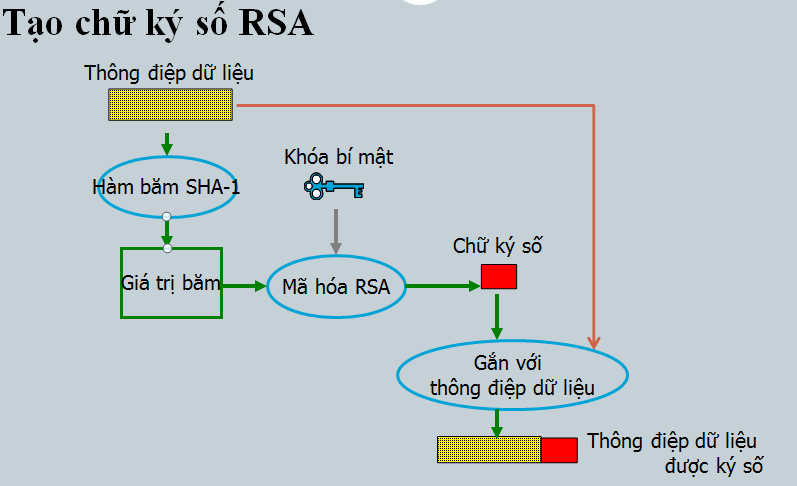
SHA-512 là một thuật toán băm an toàn thuộc họ SHA-2.

* Đầu vào: Dữ liệu có độ dài bất kỳ.
* Đầu ra: Luôn tạo ra một giá trị băm có độ dài cố định là 512 bit.
* Ứng dụng chính: Kiểm tra tính toàn vẹn của dữ liệu. Bằng cách so sánh giá trị băm của file trước và sau khi truyền, ta có thể chắc chắn rằng file không bị thay đổi.
  1. **Chữ ký số**

Chữ ký số là sự kết hợp giữa hàm băm và mã hóa bất đối xứng để đảm bảo tính xác thực và tính toàn vẹn cho dữ liệu số.

1. Quy trình tạo chữ ký số (Phía người gửi)

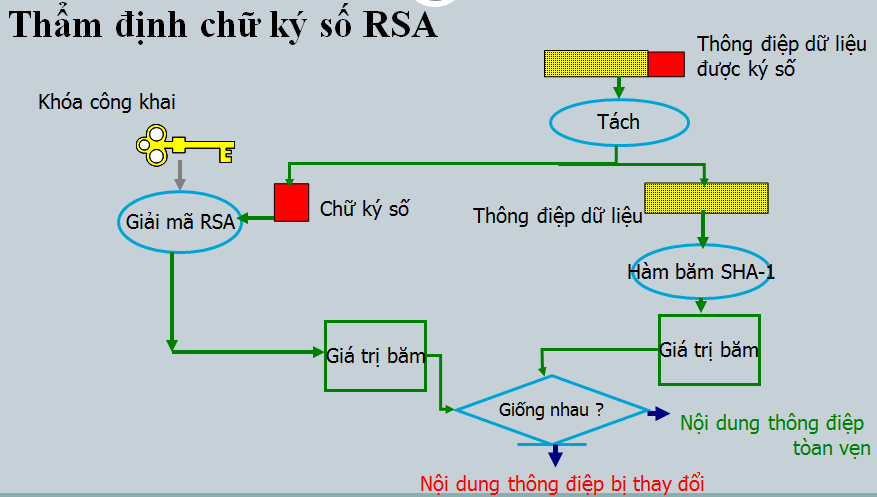
* Băm (Hash): Dùng một hàm băm (ví dụ: SHA-512) để tạo ra một bản tóm lược (giá trị băm) duy nhất từ thông điệp gốc.
* Ký (Sign): Dùng Khóa bí mật (Private Key) của chính mình để mã hóa giá trị băm đó. Kết quả chính là chữ ký số.
* Gửi đi: Gửi thông điệp gốc kèm theo chữ ký số.



Hình 2: Sơ đồ tạo chữ ký số

1. Quy trình xác thực chữ ký số (Phía người nhận)

* Giải mã chữ ký: Dùng Khóa công khai (Public Key) của người gửi để giải mã chữ ký số. Kết quả là giá trị băm ban đầu (gọi là Hash\_1).
* Băm lại thông điệp: Tự mình băm lại thông điệp nhận được bằng cùng một thuật toán. Kết quả là một giá trị băm thứ hai (gọi là Hash\_2).
* So sánh:
* Nếu Hash\_1 == Hash\_2: Chữ ký hợp lệ. Thông điệp được xác thực là đến từ đúng người gửi và nội dung không bị thay đổi.
* Nếu Hash\_1 == Hash\_2: Chữ ký không hợp lệ. Thông điệp đã bị thay đổi hoặc bị giả mạo.



Hình 3: Sơ đồ xác định chữ ký số

**CHƯƠNG 2. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG**

* 1. **Phân tích yêu cầu**

1. Yêu cầu chức năng

Người gửi (Client): Phải kết nối được đến Server, đọc và chia file, tạo khóa phiên DES, mã hóa khóa phiên bằng RSA, ký số bằng RSA, mã hóa và băm từng phần file, sau đó gửi đi và chờ phản hồi.

Người nhận (Server): Phải lắng nghe kết nối, nhận và xác thực (giải mã khóa DES, xác thực chữ ký), kiểm tra tính toàn vẹn của từng phần file (so sánh hash), giải mã và ghép lại file hoàn chỉnh, rồi gửi thông báo thành công.

1. Yêu cầu phi chức năng

Bảo mật: Đảm bảo đủ 3 tính chất: bí mật (mã hóa), toàn vẹn (hashing), và xác thực (chữ ký số).

Đúng đắn: File sau khi giải mã phải giống hệt file gốc.

Hiệu năng: Tốc độ xử lý hợp lý.

Xử lý lỗi: Chương trình phải phát hiện và thông báo lỗi rõ ràng.

1. Vai trò của từng thuật toán

DES (Mã hóa đối xứng): Đảm bảo TÍNH BÍ MẬT cho nội dung file vì có tốc độ nhanh, phù hợp mã hóa dữ liệu lớn.

RSA (Mã hóa bất đối xứng):

* Trao đổi khóa: Đảm bảo việc gửi khóa DES một cách an toàn.
* Chữ ký số: Đảm bảo TÍNH XÁC THỰC của người gửi.

SHA-512 (Hàm băm): Đảm bảo TÍNH TOÀN VẸN cho dữ liệu bằng cách tạo ra một "dấu vân tay số" duy nhất cho mỗi gói tin.

* 1. **Mô hình kiến trúc**

1. Mô hình Client-Server

Hệ thống được xây dựng theo mô hình Client-Server.

* Client (Người gửi): Là bên chủ động, có nhiệm vụ khởi tạo kết nối, xử lý file (chia nhỏ, mã hóa, ký số) và gửi đi.
* Server (Người nhận): Là bên bị động, có nhiệm vụ lắng nghe, chấp nhận kết nối, xác thực, kiểm tra, giải mã và ghép file.
* Mô hình này giúp tách biệt vai trò, làm cho hệ thống có cấu trúc rõ ràng và dễ quản lý.

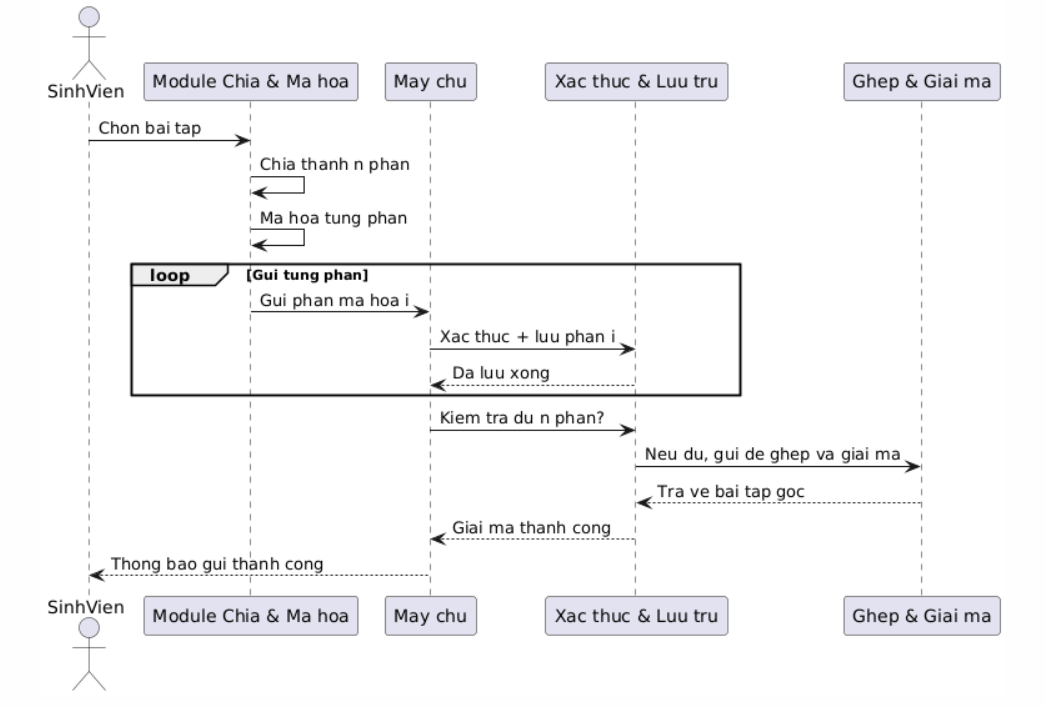
1. Luồng tương tác chính

* Bắt tay: Client và Server kiểm tra sự sẵn sàng của nhau.
* Xác thực & Trao đổi khóa: Client gửi khóa phiên (đã được mã hóa bằng Public Key của Server) và chữ ký số (tạo bằng Private Key của Client) cho Server. Server xác thực và lấy khóa phiên.
* Truyền dữ liệu: Client gửi tuần tự các phần file đã được mã hóa (bằng DES) và bảo vệ toàn vẹn (bằng SHA-512).
* Hoàn tất: Server kiểm tra, giải mã, ghép file hoàn chỉnh và gửi lại thông báo thành công (ACK).

Việc chia rõ vai trò Client–Server giúp hệ thống mở rộng dễ dàng và tăng cường khả năng bảo trì.

* 1. **Thiết kế luồng xử lý chi tiết**

Toàn bộ quá trình giao tiếp giữa Client và Server được thiết kế theo một giao thức chặt chẽ, tuần tự. Sơ đồ tuần tự (Sequence Diagram) dưới đây mô tả tổng quan các bước tương tác chính giữa hai bên.



Hình 4: Sơ đồ hoạt động của hệ thống chia nhỏ, mã hóa và xác thực bài tập

* + 1. **Giai đoạn Bắt tay (Handshake): Khởi tạo kết nối**

Đây là giai đoạn đầu tiên, nhằm mục đích thiết lập một kênh giao tiếp và xác nhận sự sẵn sàng của cả hai bên.

* Bước 1 (Client → Server): Client khởi tạo kết nối TCP đến địa chỉ IP và cổng của Server, sau đó gửi một thông điệp đơn giản là hello!.
* Bước 2 (Server → Client): Server, sau khi nhận được yêu cầu và chấp nhận kết nối, sẽ gửi lại thông điệp Ready! để báo hiệu rằng nó đã sẵn sàng cho các bước tiếp theo. Nếu Server không sẵn sàng, nó có thể đóng kết nối.
  + 1. **Giai đoạn xác thực và trao đổi khóa**

Giai đoạn này là trái tim của giao thức bảo mật, thực hiện đồng thời hai nhiệm vụ quan trọng: xác thực danh tính của Client và thiết lập một khóa bí mật chung (khóa phiên) cho việc mã hóa dữ liệu.

* Hoạt động phía Client:
  1. Tạo khóa phiên: Client tạo ra một khóa đối xứng ngẫu nhiên, có độ dài phù hợp với thuật toán DES (56 bit). Đây là khóa phiên (Session Key), chỉ được sử dụng cho phiên làm việc hiện tại.
  2. Mã hóa khóa phiên: Client sử dụng Khóa công khai (Public Key) của Server để mã hóa khóa phiên vừa tạo. Kết quả là Encrypted\_Session\_Key.
  3. Tạo chữ ký số:
* Client chuẩn bị một chuỗi dữ liệu định danh (metadata),
* Sử dụng thuật toán SHA-512 để băm chuỗi metadata này, thu được một giá trị băm H\_meta.
* Sử dụng Khóa bí mật (Private Key) của chính mình để mã hóa giá trị băm H\_meta. Kết quả là Chữ ký số (Digital Signature).
  1. Gửi đi: Client gửi Encrypted\_Session\_Key và Digital\_Signature cho Server.
  + Hoạt động phía Server:
* Giải mã khóa phiên: Server nhận được Encrypted\_Session\_Key và dùng Khóa bí mật của chính mình để giải mã, thu được khóa phiên gốc. Từ thời điểm này, cả hai bên đều đã có chung một khóa bí mật.
  + Xác thực chữ ký:
* Server dùng Khóa công khai của Client để giải mã Digital\_Signature, thu được giá trị băm H\_meta ban đầu.
* Server tự tạo lại chuỗi metadata (dựa trên thông tin đã thỏa thuận) và dùng SHA-512 để tính toán giá trị băm của nó.
* So sánh hai giá trị băm. Nếu trùng khớp, danh tính Client được xác thực thành công. Nếu không, Server sẽ gửi NACK (Authentication Error) và đóng kết nối.
  + 1. **Giai đoạn Truyền dữ liệu**

Sau khi xác thực thành công, Client bắt đầu gửi nội dung file.

* Chia nhỏ file: Client đọc nội dung và chia thành 3 phần (chunks) có kích thước phù hợp.
* Mã hóa và đóng gói: Với mỗi phần file, Client thực hiện:
* Tạo một Vector khởi tạo (IV) ngẫu nhiên, duy nhất cho mỗi phần để tăng tính an toàn cho chế độ mã hóa khối.
* Sử dụng thuật toán DES cùng với khóa phiên và IV để mã hóa nội dung của phần file.
* Sử dụng thuật toán SHA-512 để tính toán giá trị băm của chuỗi.
* Đóng gói toàn bộ thông tin vào một cấu trúc dữ liệu (ví dụ: JSON) và gửi cho Server.
* Gửi tuần tự: Client gửi lần lượt gói tin của Phần 1, Phần 2, và Phần 3.
  + 1. **Giai đoạn Xử lý tại phía nhận**

Server nhận và xử lý từng gói tin một cách cẩn thận.

* Kiểm tra toàn vẹn: Với mỗi gói tin nhận được:
* Server tách IV, Ciphertext\_Part và Integrity\_Hash.
* Nó tự tính toán lại giá trị băm SHA-512 trên chuỗi IV || Ciphertext\_Part.
* So sánh giá trị băm vừa tính với Integrity\_Hash nhận được. Nếu không khớp, Server kết luận dữ liệu đã bị thay đổi, gửi NACK và hủy phiên.
* Giải mã: Nếu kiểm tra toàn vẹn thành công, Server dùng khóa phiên và IV tương ứng để giải mã Ciphertext\_Part, thu được nội dung gốc của phần file đó.
* Ghép file: Server lưu trữ các phần đã giải mã. Sau khi nhận và xử lý thành công cả 3 phần, nó sẽ ghép chúng lại theo đúng thứ tự để tái tạo lại file assignment.txt hoàn chỉnh.
* Lưu file: Server lưu file đã được khôi phục vào hệ thống của mình.
  + 1. **Giai đoạn phản hồi (ACK/NACK)**

Đây là giai đoạn cuối cùng, nhằm thông báo kết quả của toàn bộ quá trình.

* ACK (Acknowledgement): Nếu Server nhận, xác thực, giải mã và ghép file thành công mà không gặp bất kỳ lỗi nào, nó sẽ gửi một thông điệp ACK cho Client. Nhận được ACK, Client biết rằng file đã được gửi đi thành công.
* NACK (Negative Acknowledgement): Nếu có bất kỳ lỗi nào xảy ra trong suốt quá trình (lỗi xác thực, lỗi toàn vẹn, lỗi kết nối...), Server sẽ gửi một thông điệp NACK kèm theo mã lỗi tương ứng cho Client. Nhận được NACK, Client biết rằng quá trình đã thất bại và cần phải thực hiện lại.

**CHƯƠNG 3. KẾT QUẢ THỰC NGHIỆM**

### 3.1. Môi trường thực nghiệm

Để triển khai và kiểm thử hệ thống, nhóm sử dụng môi trường phần mềm và phần cứng như sau:

* **Hệ điều hành:** Windows 11
* **Ngôn ngữ lập trình:** Python 3.12 (CPython)
* **Thư viện chính:**
  + pycryptodome: hỗ trợ các thuật toán mã hóa RSA, DES, SHA-512
  + tkinter: tạo giao diện người dùng
  + socket: xử lý truyền thông TCP
  + json, base64: xử lý dữ liệu và mã hóa
* **Công cụ phát triển:** Visual Studio Code, Terminal, Tkinter GUI
* **Cấu hình phần cứng:**
  + CPU: Intel Core i5-1035G1
  + RAM: 8GB
  + Ổ cứng: SSD 256GB

### 3.2. Mô tả hệ thống

Hệ thống truyền file được triển khai theo mô hình **Client - Server**, trong đó:

* **Client (Sender):** Đóng vai trò gửi file.
* **Server (Receiver):** Lắng nghe, nhận và xử lý dữ liệu.

Dữ liệu được mã hóa, chia nhỏ, kiểm tra toàn vẹn và xác thực chữ ký số trong quá trình truyền để đảm bảo tính bảo mật.

Hệ thống bao gồm các thành phần chính:

|  |  |
| --- | --- |
| **Tệp** | **Chức năng chính** |
| generate\_keys.py | Tạo cặp khóa RSA 1024-bit cho người gửi và người nhận. |
| cryptography\_utils.py | Xử lý các thuật toán mã hóa RSA, DES, băm SHA-512 và tiện ích truyền dữ liệu. |
| sender\_gui.py | Giao diện gửi file: chọn file, kết nối, mã hóa, gửi theo từng phần. |
| receiver\_gui.py | Giao diện nhận file: xác thực, kiểm tra hash, giải mã, lưu file. |

from Crypto.PublicKey import RSA

def generate\_and\_save\_keys(name, key\_size=1024):

"""Tạo và lưu cặp khóa RSA."""

try:

key = RSA.generate(key\_size)

# Xuất và lưu private key

private\_key = key.export\_key()

with open(f"{name}\_private.pem", "wb") as f\_priv:

f\_priv.write(private\_key)

# Xuất và lưu public key

public\_key = key.publickey().export\_key()

with open(f"{name}\_public.pem", "wb") as f\_pub:

f\_pub.write(public\_key)

print(f"Đã tạo thành công khóa cho '{name}' tại {name}\_private.pem và {name}\_public.pem")

except Exception as e:

print(f"Lỗi khi tạo khóa cho {name}: {e}")

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

print("Bắt đầu tạo khóa RSA (1024-bit)...")

generate\_and\_save\_keys("sender")

generate\_and\_save\_keys("receiver")

print("Hoàn tất.")

### 3.3. Luồng hoạt động tổng thể

Quy trình truyền file gồm các bước sau:

1. **Tạo khóa RSA:**  
   Tệp generate\_keys.py được chạy đầu tiên, tạo ra các file .pem:
   * sender\_private.pem, sender\_public.pem
   * receiver\_private.pem, receiver\_public.pem
2. **Khởi động bên nhận (Receiver):**
   * Người dùng chạy receiver\_gui.py và nhấn nút **Ready!**
   * Server bắt đầu lắng nghe kết nối đến cổng 9999.
3. **Khởi động bên gửi (Sender):**
   * Người dùng chạy sender\_gui.py, chọn file cần gửi và nhập IP bên nhận (mặc định 127.0.0.1).
   * Nhấn **Gửi File** để bắt đầu truyền file.
4. **Giai đoạn handshake:**
   * Sender gửi chuỗi "Hello!" để kiểm tra kết nối.
   * Receiver xác nhận "Ready!".
5. **Gửi metadata và session key:**
   * Sender tạo khóa DES ngẫu nhiên (Session Key).
   * Session Key được mã hóa bằng RSA (public key của receiver).
   * Metadata gồm tên file, thời gian, số phần chia file.
   * Metadata được ký số bằng private key của sender.
6. **Gửi các phần dữ liệu:**
   * File được chia thành 3 phần gần bằng nhau.
   * Mỗi phần được mã hóa bằng DES-CBC (với IV riêng), băm SHA-512.
   * Gửi từng phần: iv, ciphertext, hash, part\_index.
7. **Bên nhận xác thực và giải mã:**
   * Receiver xác thực chữ ký metadata.
   * Giải mã Session Key.
   * Kiểm tra tính toàn vẹn từng phần bằng hash.
   * Giải mã và ghép các phần lại, cho phép lưu file.
8. **Phản hồi ACK/NACK:**
   * Nếu thành công: Receiver gửi "ACK".
   * Nếu lỗi (hash sai, chữ ký sai...): Receiver gửi "NACK".

### code send\_file():

def send\_thread(self):

host = self.ip\_entry.get()

port = 9999

filepath = self.file\_path.get()

try:

with socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) as s:

self.log\_message(f"Đang kết nối tới {host}:{port}...")

s.connect((host, port))

# 1. Handshake

s.sendall(b"Hello!")

ready = s.recv(1024).decode()

if ready != "Ready!":

raise ConnectionAbortedError("Handshake thất bại.")

self.log\_message("Handshake thành công.")

# 2. Chuẩn bị dữ liệu và gửi control message

session\_key = get\_random\_bytes(8) # DES key is 8 bytes

encrypted\_key\_b64 = encrypt\_rsa(session\_key, self.receiver\_public\_key)

metadata = {

'filename': os.path.basename(filepath),

'timestamp': time.time(),

'parts': 3

}

metadata\_bytes = json.dumps(metadata, sort\_keys=True).encode('utf-8')

signature\_b64 = sign\_data(metadata\_bytes, self.sender\_private\_key)

control\_message = {

'encrypted\_key': encrypted\_key\_b64,

'metadata': metadata,

'signature': signature\_b64

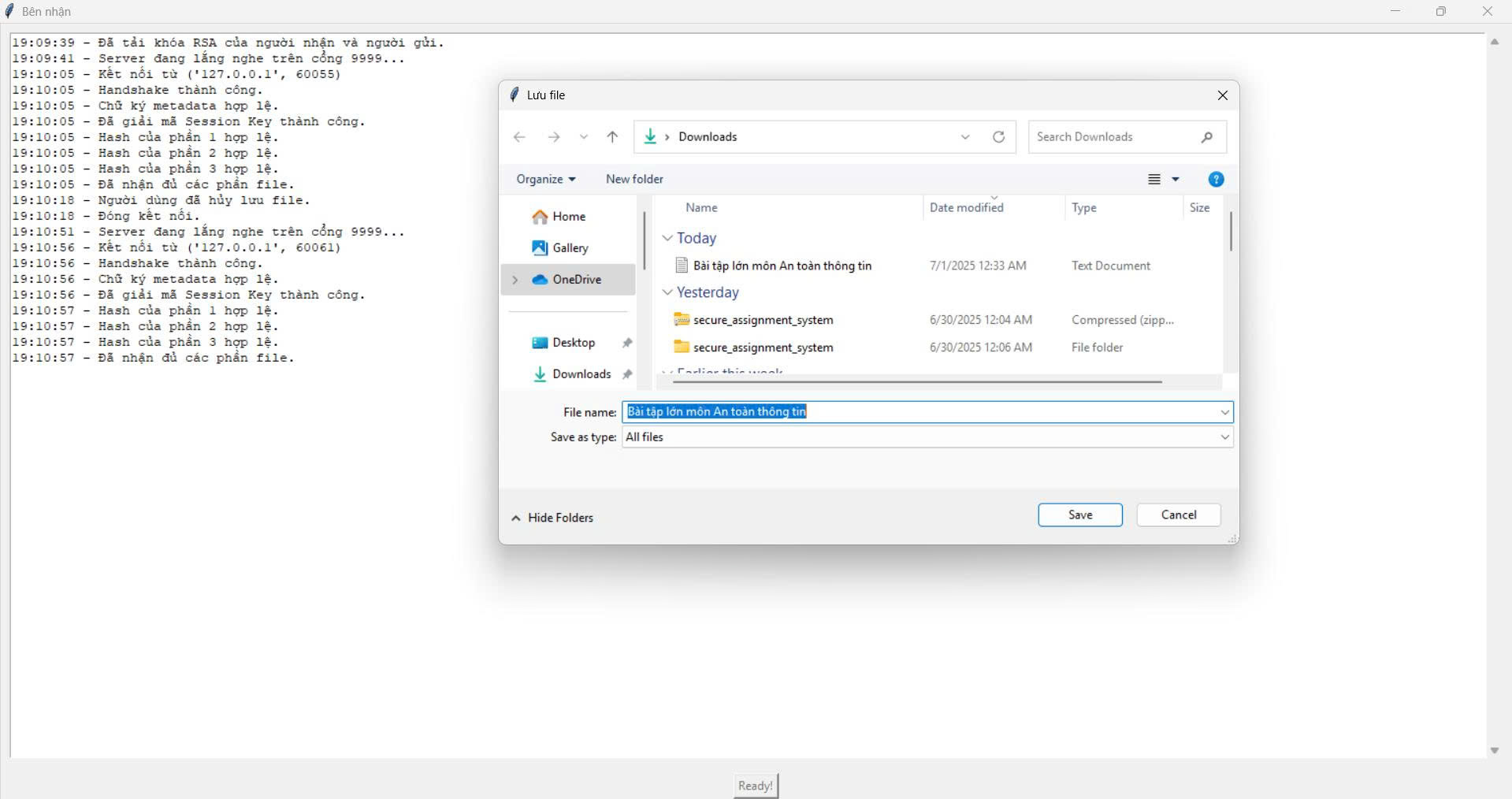
}

send\_message(s, control\_message)

self.log\_message("Đã gửi Session Key (đã mã hóa) và Metadata (đã ký).")

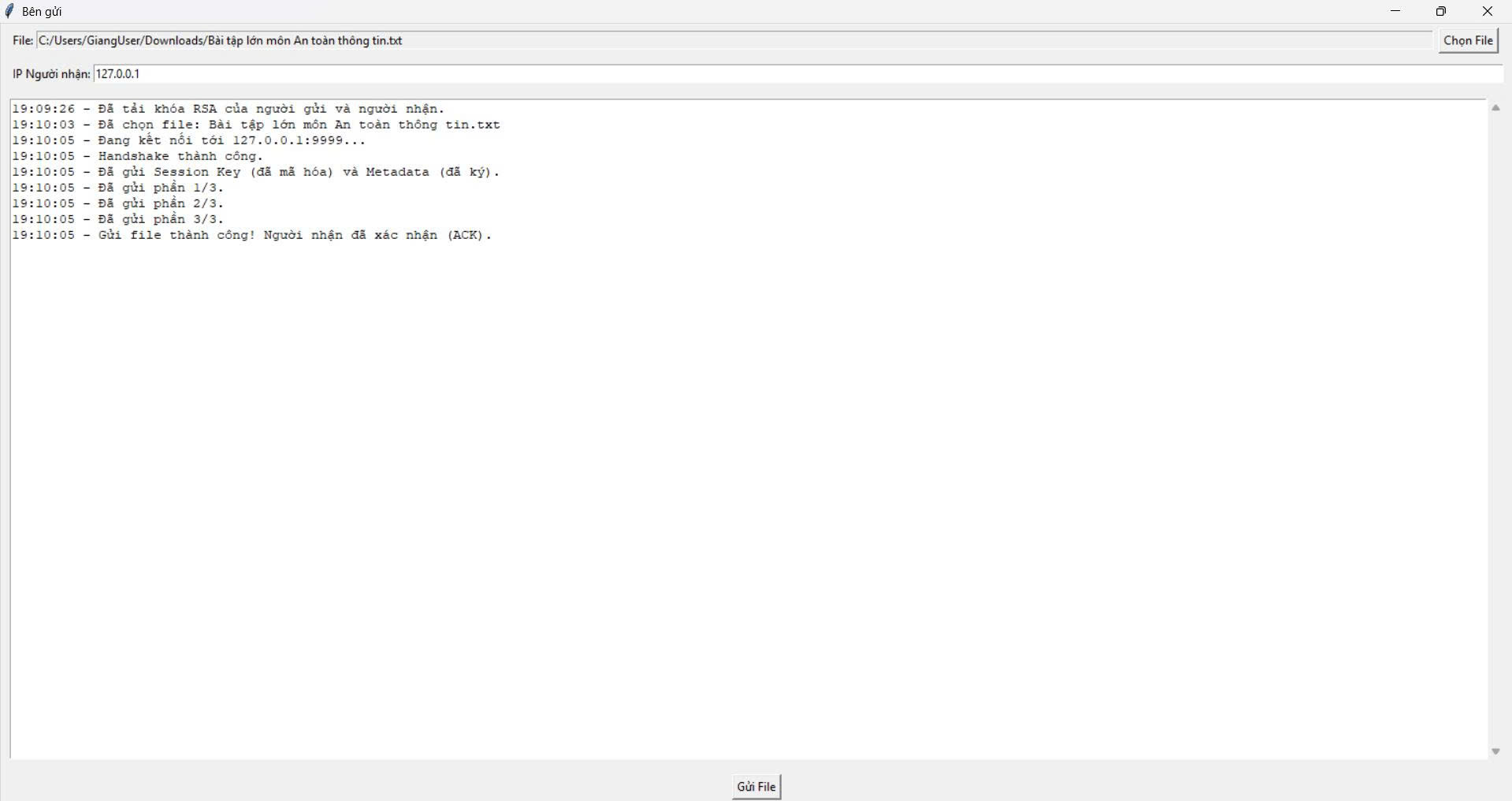
### 3.4. Giao diện người dùng

Để minh họa quá trình gửi và nhận file, nhóm tiến hành chạy ứng dụng và ghi lại kết quả thực tế qua các giao diện sau:



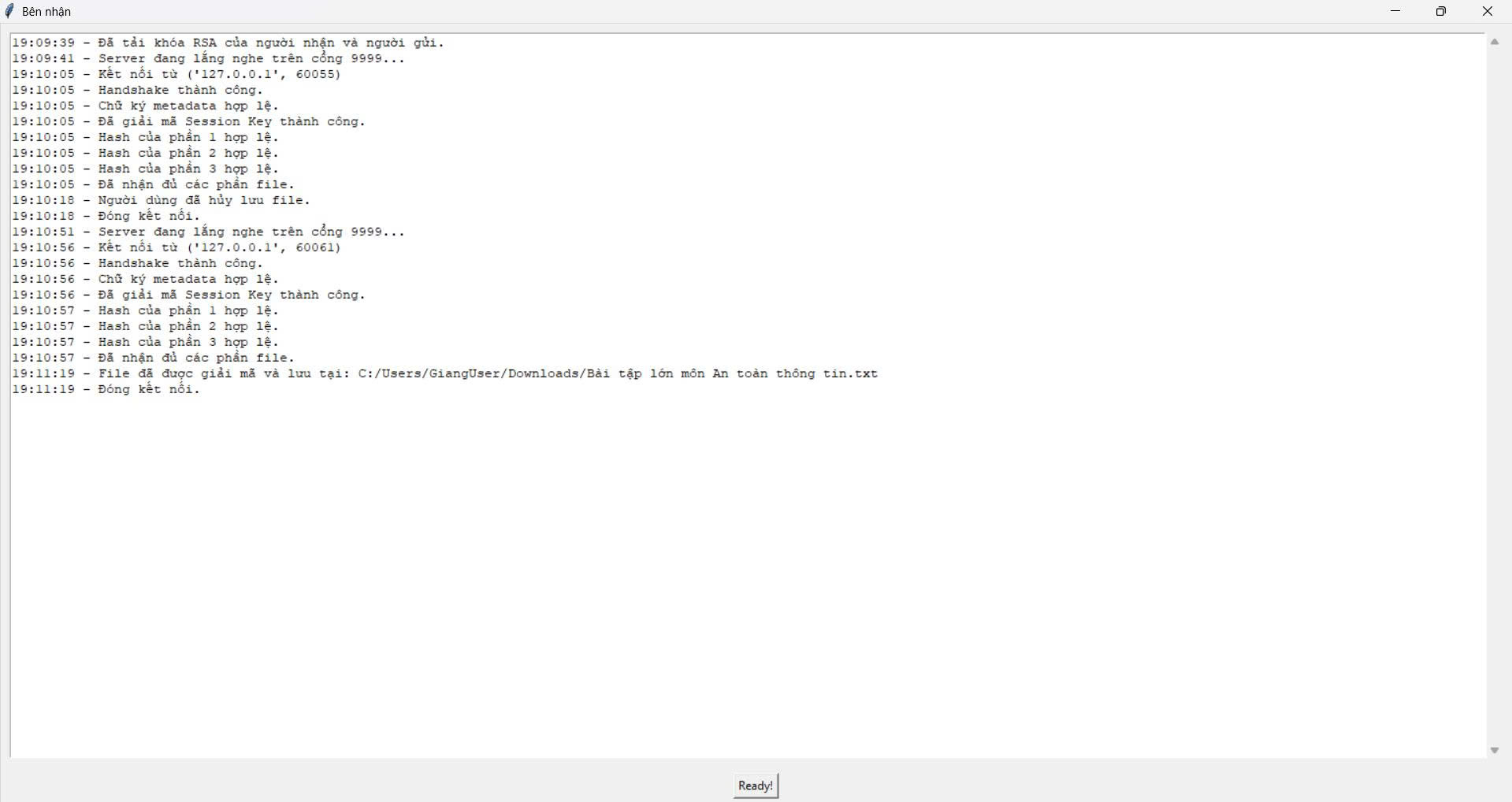
*Hình 5: Giao diện bên nhận (Receiver) đang sẵn sàng lắng nghe cổng 9999 và xác thực file đến.*

Hệ thống ghi lại log chi tiết về quá trình: kiểm tra chữ ký metadata, giải mã Session Key, kiểm tra hash từng phần và xác nhận thành công (ACK).



*Hình 3.2: Giao diện bên gửi (Sender) trong quá trình chọn file và gửi đi.*

Người dùng chọn file, nhập IP người nhận và nhấn "Gửi File". Mỗi bước trong quá trình gửi (handshake, mã hóa, chia phần) đều hiển thị trong nhật ký.



*Hình 3.3: Thông báo bên nhận lưu thành công file giải mã.*

Sau khi nhận đủ và kiểm tra hash thành công, bên nhận cho phép lưu file ra máy người dùng. Đường dẫn được ghi rõ trên giao diện.

* **Giao diện người gửi (Sender):**
  + Cho phép chọn file bằng hộp thoại.
  + Nhập IP người nhận.
  + Hiển thị nhật ký gửi từng bước: kết nối, handshake, mã hóa, gửi từng phần...
* **Giao diện người nhận (Receiver):**
  + Có nút "Ready!" để khởi động lắng nghe.
  + Hiển thị thông tin từng bước: xác thực, nhận phần, kiểm tra hash, giải mã...

→ Cả hai giao diện đều có khung log (*ScrolledText*) để hiển thị trạng thái theo thời gian thực.

### 3.5. Kết quả thử nghiệm

|  |  |
| --- | --- |
| **Tên thử nghiệm** | **Kết quả** |
| Gửi file .txt nhỏ (~5KB) | Thành công trong < 1 giây |
| Gửi file .docx ~1MB | Thành công, đúng định dạng, hash kiểm tra khớp |
| Gửi file bị sửa giữa chừng | Phát hiện hash sai, phản hồi NACK |
| Mất kết nối giữa phiên truyền | Hệ thống log lỗi và không lưu file không đầy đủ |
| File mã hóa thành công | Mỗi phần mã hóa riêng biệt bằng DES, IV riêng |
| File giải mã ra đúng nội dung | Không có sai lệch byte, khớp với file gốc 100% |

### 3.6. Phân tích bảo mật

|  |  |  |
| --- | --- | --- |
| **Tầng bảo vệ** | **Kỹ thuật sử dụng** | **Ý nghĩa** |
| Bảo mật nội dung | Mã hóa DES-CBC với session key 8 bytes | Ngăn người lạ đọc được nội dung file |
| Bảo mật khóa | Session Key được mã hóa bằng RSA | Chỉ người nhận sở hữu private key mới giải mã được |
| Toàn vẹn dữ liệu | SHA-512 Hash + kiểm tra hash | Nếu file bị thay đổi, hash sẽ không khớp, hệ thống từ chối nhận |
| Xác thực nguồn gốc | Chữ ký số bằng RSA/SHA-512 | Đảm bảo file là từ đúng người gửi, không bị giả mạo |

### 3.7. Đánh giá tổng quan

* **Ưu điểm:**
  + Kết hợp đa lớp bảo mật: RSA, DES, SHA-512, Signature
  + Giao diện dễ thao tác, phù hợp cả với người không chuyên
  + Quá trình xử lý có log chi tiết để dễ dàng theo dõi, debug
  + Có kiểm tra toàn vẹn và xác thực nguồn gốc file gửi
* **Hạn chế:**
  + Cố định chia thành 3 phần, không tùy chỉnh theo dung lượng
  + Thiếu xác thực danh tính người gửi/nhận qua tên đăng nhập
  + Giao thức truyền đơn giản, chưa mã hóa toàn bộ luồng socket
  + Không có tính năng tiếp tục gửi nếu mất kết nối giữa chừng

**Quản lý khóa RSA**

* import os
* import json
* import base64
* from Crypto.PublicKey import RSA
* from Crypto.Cipher import DES, PKCS1\_v1\_5
* from Crypto.Signature import pkcs1\_15
* from Crypto.Hash import SHA512
* from Crypto.Random import get\_random\_bytes
* from Crypto.Util.Padding import pad, unpad
* BLOCK\_SIZE\_DES = 8 # Kích thước khối của DES là 8 bytes
* # --- Quản lý khóa RSA ---
* def load\_rsa\_key(key\_path):
* """Tải khóa RSA từ file."""
* with open(key\_path, 'rb') as f:
* return RSA.import\_key(f.read())
* # --- Mã hóa/Giải mã RSA (cho Session Key) ---
* def encrypt\_rsa(data, public\_key):
* """Mã hóa dữ liệu bằng Public Key RSA (PKCS#1 v1.5)."""
* cipher\_rsa = PKCS1\_v1\_5.new(public\_key)
* encrypted\_data = cipher\_rsa.encrypt(data)
* return base64.b64encode(encrypted\_data).decode('utf-8')
* def decrypt\_rsa(encrypted\_data\_b64, private\_key):
* """Giải mã dữ liệu bằng Private Key RSA (PKCS#1 v1.5)."""
* encrypted\_data = base64.b64decode(encrypted\_data\_b64)
* cipher\_rsa = PKCS1\_v1\_5.new(private\_key)
* # Cần một sentinel để xử lý lỗi giải mã
* sentinel = get\_random\_bytes(16)
* decrypted\_data = cipher\_rsa.decrypt(encrypted\_data, sentinel)
* return decrypted\_data
* # --- Ký số/Xác thực RSA (cho Metadata) ---
* def sign\_data(data, private\_key):
* """Ký lên dữ liệu bằng Private Key RSA và SHA-512."""
* h = SHA512.new(data)
* signer = pkcs1\_15.new(private\_key)
* signature = signer.sign(h)
* return base64.b64encode(signature).decode('utf-8')
* def verify\_signature(data, signature\_b64, public\_key):
* """Xác thực chữ ký bằng Public Key RSA và SHA-512."""
* h = SHA512.new(data)
* signature = base64.b64decode(signature\_b64)
* verifier = pkcs1\_15.new(public\_key)
* try:
* verifier.verify(h, signature)
* return True
* except (ValueError, TypeError):
* return False
* # --- Mã hóa/Giải mã DES (cho File) ---
* def encrypt\_des(data, key, iv):
* """Mã hóa dữ liệu bằng DES (chế độ CBC)."""
* cipher\_des = DES.new(key, DES.MODE\_CBC, iv)
* padded\_data = pad(data, BLOCK\_SIZE\_DES)
* encrypted\_data = cipher\_des.encrypt(padded\_data)
* return base64.b64encode(encrypted\_data).decode('utf-8')
* def decrypt\_des(encrypted\_data\_b64, key, iv):
* """Giải mã dữ liệu bằng DES (chế độ CBC)."""
* encrypted\_data = base64.b64decode(encrypted\_data\_b64)
* cipher\_des = DES.new(key, DES.MODE\_CBC, iv)
* decrypted\_data = cipher\_des.decrypt(encrypted\_data)
* unpadded\_data = unpad(decrypted\_data, BLOCK\_SIZE\_DES)
* return unpadded\_data
* # --- Hashing ---
* def calculate\_hash(iv\_b64, cipher\_b64):
* """Tính hash SHA-512 cho IV và Ciphertext."""
* iv = base64.b64decode(iv\_b64)
* cipher = base64.b64decode(cipher\_b64)
* h = SHA512.new(iv + cipher)
* return h.hexdigest()
* # --- Tiện ích mạng ---
* def send\_message(sock, data):
* """Gửi một đối tượng JSON qua socket với tiền tố độ dài."""
* message = json.dumps(data).encode('utf-8')
* msg\_len = len(message)
* sock.sendall(msg\_len.to\_bytes(4, 'big'))
* sock.sendall(message)
* def receive\_message(sock):
* """Nhận một đối tượng JSON từ socket."""
* raw\_msg\_len = sock.recv(4)
* if not raw\_msg\_len:
* return None
* msg\_len = int.from\_bytes(raw\_msg\_len, 'big')
* message = sock.recv(msg\_len)
* return json.loads(message.decode('utf-8'))

**Tạo và lưu cặp khóa RSA**

from Crypto.PublicKey import RSA

def generate\_and\_save\_keys(name, key\_size=1024):

    """Tạo và lưu cặp khóa RSA."""

    try:

        key = RSA.generate(key\_size)

        # Xuất và lưu private key

        private\_key = key.export\_key()

        with open(f"{name}\_private.pem", "wb") as f\_priv:

            f\_priv.write(private\_key)

        # Xuất và lưu public key

        public\_key = key.publickey().export\_key()

        with open(f"{name}\_public.pem", "wb") as f\_pub:

            f\_pub.write(public\_key)

        print(f"Đã tạo thành công khóa cho '{name}' tại {name}\_private.pem và {name}\_public.pem")

    except Exception as e:

        print(f"Lỗi khi tạo khóa cho {name}: {e}")

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

    print("Bắt đầu tạo khóa RSA (1024-bit)...")

    generate\_and\_save\_keys("sender")

    generate\_and\_save\_keys("receiver")

    print("Hoàn tất.")

**Bên gửi**

import tkinter as tk

from tkinter import filedialog, scrolledtext

import socket

import threading

import time

import os

import json

from cryptography\_utils import \*

class SenderApp:

    def \_\_init\_\_(self, root):

        self.root = root

        self.root.title("Bên gửi")

        self.root.geometry("500x450")

        # Khai báo biến

        self.file\_path = tk.StringVar()

        # Frame cho việc chọn file

        file\_frame = tk.Frame(root)

        file\_frame.pack(padx=10, pady=5, fill=tk.X)

        tk.Label(file\_frame, text="File:").pack(side=tk.LEFT)

        tk.Entry(file\_frame, textvariable=self.file\_path, state='readonly').pack(side=tk.LEFT, fill=tk.X, expand=True)

        tk.Button(file\_frame, text="Chọn File", command=self.select\_file).pack(side=tk.LEFT, padx=5)

        # Frame cho địa chỉ IP

        ip\_frame = tk.Frame(root)

        ip\_frame.pack(padx=10, pady=5, fill=tk.X)

        tk.Label(ip\_frame, text="IP Người nhận:").pack(side=tk.LEFT)

        self.ip\_entry = tk.Entry(ip\_frame)

        self.ip\_entry.insert(0, "127.0.0.1")  # Default IP

        self.ip\_entry.pack(side=tk.LEFT, fill=tk.X, expand=True)

        # Log area

        self.log\_area = scrolledtext.ScrolledText(root, wrap=tk.WORD, state='disabled')

        self.log\_area.pack(padx=10, pady=10, fill=tk.BOTH, expand=True)

        # Nút Gửi

        self.send\_button = tk.Button(root, text="Gửi File", command=self.send\_file)

        self.send\_button.pack(pady=5)

        # Tải khóa

        try:

            self.sender\_private\_key = load\_rsa\_key("sender\_private.pem")

            self.receiver\_public\_key = load\_rsa\_key("receiver\_public.pem")

            self.log\_message("Đã tải khóa RSA của người gửi và người nhận.")

        except FileNotFoundError:

            self.log\_message("LỖI: Không tìm thấy file khóa. Vui lòng chạy generate\_keys.py trước.")

            self.send\_button.config(state='disabled')

    def log\_message(self, message):

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

        self.log\_area.insert(tk.END, f"{time.strftime('%H:%M:%S')} - {message}\n")

        self.log\_area.yview(tk.END)

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

    def select\_file(self):

        path = filedialog.askopenfilename()

        if path:

            self.file\_path.set(path)

            self.log\_message(f"Đã chọn file: {os.path.basename(path)}")

    def send\_file(self):

        if not self.file\_path.get():

            self.log\_message("Lỗi: Vui lòng chọn file trước khi gửi.")

            return

        self.send\_button.config(state='disabled')

        thread = threading.Thread(target=self.send\_thread, daemon=True)

        thread.start()

    def send\_thread(self):

        host = self.ip\_entry.get()

        port = 9999

        filepath = self.file\_path.get()

        try:

            with socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) as s:

                self.log\_message(f"Đang kết nối tới {host}:{port}...")

                s.connect((host, port))

                # 1. Handshake

                s.sendall(b"Hello!")

                ready = s.recv(1024).decode()

                if ready != "Ready!":

                    raise ConnectionAbortedError("Handshake thất bại.")

                self.log\_message("Handshake thành công.")

                # 2. Chuẩn bị dữ liệu và gửi control message

                session\_key = get\_random\_bytes(8)  # DES key is 8 bytes

                encrypted\_key\_b64 = encrypt\_rsa(session\_key, self.receiver\_public\_key)

                metadata = {

                    'filename': os.path.basename(filepath),

                    'timestamp': time.time(),

                    'parts': 3

                }

                metadata\_bytes = json.dumps(metadata, sort\_keys=True).encode('utf-8')

                signature\_b64 = sign\_data(metadata\_bytes, self.sender\_private\_key)

                control\_message = {

                    'encrypted\_key': encrypted\_key\_b64,

                    'metadata': metadata,

                    'signature': signature\_b64

                }

                send\_message(s, control\_message)

                self.log\_message("Đã gửi Session Key (đã mã hóa) và Metadata (đã ký).")

                # 3. Đọc, chia file và gửi từng phần

                with open(filepath, 'rb') as f:

                    file\_content = f.read()

                file\_size = len(file\_content)

                part\_size = (file\_size + 2) // 3  # Chia làm 3 phần gần bằng nhau

                for i in range(3):

                    start = i \* part\_size

                    end = start + part\_size

                    part\_data = file\_content[start:end]

                    if not part\_data: continue

                    iv = get\_random\_bytes(8)  # DES IV is 8 bytes

                    iv\_b64 = base64.b64encode(iv).decode('utf-8')

                    cipher\_b64 = encrypt\_des(part\_data, session\_key, iv)

                    hash\_val = calculate\_hash(iv\_b64, cipher\_b64)

                    part\_message = {

                        'part\_index': i + 1,

                        'iv': iv\_b64,

                        'cipher': cipher\_b64,

                        'hash': hash\_val

                    }

                    send\_message(s, part\_message)

                    self.log\_message(f"Đã gửi phần {i + 1}/3.")

                    time.sleep(0.1)  # Thêm độ trễ nhỏ

                # 4. Nhận phản hồi

                response = s.recv(1024).decode()

                if response == "ACK":

                    self.log\_message("Gửi file thành công! Người nhận đã xác nhận (ACK).")

                else:

                    self.log\_message("Gửi file thất bại! Người nhận phản hồi (NACK).")

        except Exception as e:

            self.log\_message(f"LỖI: {e}")

        finally:

            self.send\_button.config(state='normal')

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

    root = tk.Tk()

    app = SenderApp(root)

    root.mainloop()

**Bên nhận**

import tkinter as tk

from tkinter import scrolledtext, simpledialog, filedialog

import socket

import threading

import time

import os

import json

from cryptography\_utils import \*

class ReceiverApp:

    def \_\_init\_\_(self, root):

        self.root = root

        self.root.title("Bên nhận")

        self.root.geometry("500x400")

        self.log\_area = scrolledtext.ScrolledText(root, wrap=tk.WORD, state='disabled')

        self.log\_area.pack(padx=10, pady=10, fill=tk.BOTH, expand=True)

        self.start\_button = tk.Button(root, text="Ready!", command=self.start\_listening)

        self.start\_button.pack(pady=5)

        # Tải khóa private của người nhận

        try:

            self.receiver\_private\_key = load\_rsa\_key("receiver\_private.pem")

            self.sender\_public\_key = load\_rsa\_key("sender\_public.pem")

            self.log\_message("Đã tải khóa RSA của người nhận và người gửi.")

        except FileNotFoundError:

            self.log\_message("LỖI: Không tìm thấy file khóa. Vui lòng chạy generate\_keys.py trước.")

            self.start\_button.config(state='disabled')

    def log\_message(self, message):

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

        self.log\_area.insert(tk.END, f"{time.strftime('%H:%M:%S')} - {message}\n")

        self.log\_area.yview(tk.END)

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

    def start\_listening(self):

        self.start\_button.config(state='disabled')

        self.log\_message("Server đang lắng nghe trên cổng 9999...")

        thread = threading.Thread(target=self.listen\_thread, daemon=True)

        thread.start()

    def listen\_thread(self):

        host = '0.0.0.0'

        port = 9999

        with socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) as s:

            s.bind((host, port))

            s.listen()

            conn, addr = s.accept()

            with conn:

                self.log\_message(f"Kết nối từ {addr}")

                try:

                    # 1. Handshake

                    hello = conn.recv(1024).decode()

                    if hello != "Hello!":

                        raise ValueError("Handshake thất bại.")

                    conn.sendall(b"Ready!")

                    self.log\_message("Handshake thành công.")

                    # 2. Nhận SessionKey và Metadata

                    control\_message = receive\_message(conn)

                    encrypted\_key\_b64 = control\_message['encrypted\_key']

                    metadata = control\_message['metadata']

                    signature\_b64 = control\_message['signature']

                    # Xác thực chữ ký của metadata

                    metadata\_bytes = json.dumps(metadata, sort\_keys=True).encode('utf-8')

                    if not verify\_signature(metadata\_bytes, signature\_b64, self.sender\_public\_key):

                        raise ValueError("Chữ ký metadata không hợp lệ!")

                    self.log\_message("Chữ ký metadata hợp lệ.")

                    # Giải mã session key

                    session\_key = decrypt\_rsa(encrypted\_key\_b64, self.receiver\_private\_key)

                    self.log\_message("Đã giải mã Session Key thành công.")

                    # 3. Nhận các phần file

                    file\_parts\_data = {}

                    for \_ in range(metadata['parts']):

                        part\_message = receive\_message(conn)

                        part\_index = part\_message['part\_index']

                        # Kiểm tra hash

                        iv\_b64 = part\_message['iv']

                        cipher\_b64 = part\_message['cipher']

                        hash\_received = part\_message['hash']

                        hash\_calculated = calculate\_hash(iv\_b64, cipher\_b64)

                        if hash\_received != hash\_calculated:

                            raise ValueError(f"Hash của phần {part\_index} không khớp!")

                        self.log\_message(f"Hash của phần {part\_index} hợp lệ.")

                        file\_parts\_data[part\_index] = (iv\_b64, cipher\_b64)

                    self.log\_message("Đã nhận đủ các phần file.")

                    # Gửi ACK

                    conn.sendall(b"ACK")

                    # Ghép và lưu file

                    save\_path = filedialog.asksaveasfilename(

                        initialfile=metadata['filename'],

                        title="Lưu file",

                        filetypes=(("All files", "\*.\*"),)

                    )

                    if save\_path:

                        full\_content = b''

                        for i in sorted(file\_parts\_data.keys()):

                            iv\_b64, cipher\_b64 = file\_parts\_data[i]

                            iv = base64.b64decode(iv\_b64)

                            decrypted\_part = decrypt\_des(cipher\_b64, session\_key, iv)

                            full\_content += decrypted\_part

                        with open(save\_path, 'wb') as f:

                            f.write(full\_content)

                        self.log\_message(f"File đã được giải mã và lưu tại: {save\_path}")

                    else:

                        self.log\_message("Người dùng đã hủy lưu file.")

                except Exception as e:

                    self.log\_message(f"LỖI: {e}")

                    try:

                        conn.sendall(b"NACK")

                    except:

                        pass  # Kết nối có thể đã đóng

                finally:

                    self.log\_message("Đóng kết nối.")

                    self.start\_button.config(state='normal')

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

    root = tk.Tk()

    app = ReceiverApp(root)

    root.mainloop()

**KẾT LUẬN**

Sau một thời gian nghiên cứu và triển khai, đề tài "Gửi bài tập chia thành nhiều phần" đã được hoàn thành, đạt được đầy đủ các mục tiêu và yêu cầu đã đề ra ban đầu. Dự án đã thành công trong việc xây dựng một chương trình ứng dụng mô phỏng quá trình truyền file an toàn qua mạng dựa trên mô hình Client-Server.

Thông qua việc vận dụng các kiến thức đã học trong học phần Nhập môn An toàn, bảo mật thông tin, nhóm đã thiết kế và hiện thực hóa một giao thức bảo mật hoàn chỉnh, kết hợp sức mạnh của các thuật toán mật mã kinh điển. Cụ thể, hệ thống đã đáp ứng thành công ba yêu cầu bảo mật cốt lõi:

* Tính bảo mật (Confidentiality): Nội dung file đã được bảo vệ hiệu quả bằng thuật toán mã hóa đối xứng DES, đảm bảo rằng dữ liệu không thể bị đọc trộm trên đường truyền.
* Tính xác thực (Authentication): Danh tính của người gửi đã được xác minh một cách tin cậy thông qua cơ chế chữ ký số sử dụng thuật toán bất đối xứng RSA, giúp chống lại các hành vi giả mạo.
* Tính toàn vẹn (Integrity): Sự nguyên vẹn của dữ liệu đã được đảm bảo bởi hàm băm an toàn SHA-512, giúp phát hiện bất kỳ sự thay đổi trái phép nào đối với nội dung gói tin.

Các kết quả thử nghiệm trong những kịch bản khác nhau đã chứng minh rằng chương trình hoạt động ổn định, đúng đắn và xử lý lỗi hiệu quả, file nhận được ở phía Server hoàn toàn trùng khớp với file gốc.

Việc hoàn thành bài tập lớn này không chỉ giúp nhóm củng cố vững chắc kiến thức lý thuyết đã học mà còn mang lại những kinh nghiệm thực tiễn quý báu trong việc phân tích, thiết kế và triển khai một hệ thống có yêu cầu về an ninh. Đây là một nền tảng quan trọng để tiếp tục nghiên cứu và phát triển các giải pháp bảo mật phức tạp hơn trong tương lai.

**DANH MỤC TÀI LIỆU THAM KHẢO**

[1]. Stallings, William (2017). Cryptography and Network Security: Principles and Practice (7th ed.). Pearson.

[2]. Khoa An toàn thông tin, Học viện Kỹ thuật Mật mã (2020). Giáo trình Mật mã học. Nhà xuất bản Khoa học và Kỹ thuật.

[3]. https://www.tutorialspoint.com/cryptography/data\_encryption\_standard.htm

[4]. https://vi.wikipedia.org/wiki/DES\_(m%C3%A3\_h%C3%B3a)

[5]. <https://vi.wikipedia.org/wiki/RSA_(m%C3%A3_h%C3%B3a)>

[6]. <https://vi.wikipedia.org/wiki/H%C3%A0m_b%C4%83m>