**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

Logo, company name

Description automatically generated

**BÀI TẬP LỚN**

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

**ĐỀ TÀI: “GỬI HỢP ĐỒNG VỚI CHỮ KÝ SỐ RIÊNG”**

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

**Sinh viên thực hiện:**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Mã Sinh Viên** | **Họ và tên** | **Lớp** |
| **1** | **1671020334** | **Nguyễn Anh Tuấn** | **CNTT 16-05** |
| **2** | **1671020335** | **Nguyễn Anh Tuấn** | **CNTT 16-05** |
| **3** | **1671020337** | **Phạm Đình Tuấn** | **CNTT 16-05** |
| **4** | **1671020352** | **Nguyễn Hoàng Việt** | **CNTT 16-05** |

**Hà Nội, Năm 2025**

**TRƯỜNG ĐẠI HỌC ĐẠI NAM**

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

Logo, company name

Description automatically generated

**BÀI TẬP LỚN**

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

**ĐỀ TÀI: “GỬI HỢP ĐỒNG VỚI CHỮ KÝ SỐ RIÊNG”**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **STT** | **Mã Sinh Viên** | **Họ và Tên** | **Ngày Sinh** | **Điểm** | |
| **Bằng Số** | **Bằng Chữ** |
| 1 | 1671020334 | Nguyễn Anh Tuấn | 28/12/2004 |  |  |
| 2 | 1671020335 | Nguyễn Anh Tuấn | 06/07/2004 |  |  |
| 3 | 1671020337 | Phạm Đình Tuấn | 10/11/2004 |  |  |
| 4 | 1671020352 | Nguyễn Hoàng Việt | 16/12/2004 |  |  |

**CÁN BỘ CHẤM THI CÁN BỘ CHẤM THI**

**Hà Nội, Năm 2025**

**TRƯỜNG ĐẠI HỌC ĐẠI NAM**

# **LỜI NÓI ĐẦU**

Trong kỷ nguyên số hóa hiện nay, công nghệ thông tin đang đóng vai trò then chốt trong mọi lĩnh vực của đời sống, đặc biệt là trong việc trao đổi, xử lý và bảo vệ thông tin. Nhận thức được tầm quan trọng của bảo mật dữ liệu trong môi trường số, nhóm chúng em đã lựa chọn học môn An toàn và Bảo mật Thông tin với mong muốn được trang bị những kiến thức nền tảng và chuyên sâu nhằm đảm bảo an toàn cho các hệ thống, dịch vụ và dữ liệu số trong thực tế. Không chỉ dừng lại ở lý thuyết về mã hóa hay các thuật toán bảo mật, môn học này còn mang đến cho nhóm cái nhìn toàn diện về các rủi ro an ninh mạng, cách phòng ngừa và ứng phó với các mối đe dọa từ môi trường Internet. Qua môn học, nhóm đã được tìm hiểu về các công nghệ bảo mật hiện đại như chữ ký số, chứng thực điện tử, mã hóa bất đối xứng, cũng như các cơ chế đảm bảo tính toàn vẹn, xác thực và chống chối bỏ của dữ liệu – những vấn đề có vai trò sống còn trong thời đại số.

Từ đó, nhóm đã quyết định chọn đề tài “Gửi hợp đồng với chữ ký số riêng” nhằm tìm hiểu sâu hơn về cách thức áp dụng chữ ký số trong việc xác thực và bảo mật thông tin trong các giao dịch điện tử. Với đề tài này, nhóm mong muốn mô phỏng được quy trình tạo lập, ký và xác minh hợp đồng số thông qua việc sử dụng cặp khóa riêng – công khai, qua đó minh họa rõ nét cơ chế hoạt động và giá trị thực tiễn của chữ ký số trong việc bảo vệ thông tin.

Trong suốt quá trình thực hiện đề tài, nhóm không chỉ vận dụng kiến thức lý thuyết đã học mà còn rèn luyện kỹ năng làm việc nhóm, tư duy phân tích và giải quyết vấn đề. Thông qua các bước thiết kế, triển khai và kiểm thử mô hình ký số, nhóm càng hiểu rõ hơn những thách thức cũng như tiềm năng của việc ứng dụng các kỹ thuật bảo mật vào thực tế. Những trải nghiệm này chắc chắn sẽ là hành trang quý báu giúp nhóm tự tin hơn khi bước vào thế giới công nghệ đầy biến động và thách thức hiện nay.

Xin chân thành cảm ơn thầy cô và các bạn đã hỗ trợ và đồng hành cùng nhóm trong suốt quá trình học tập.

# **MỤC LỤC**

[**LỜI NÓI ĐẦU** 1](#_Toc202566260)

[**MỤC LỤC** 2](#_Toc202566261)

[**CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI** 5](#_Toc202566262)

[**1.1 Giới thiệu chung về đề tài lựa chọn** 5](#_Toc202566263)

[**1.2 Lí do chọn đề tài** 5](#_Toc202566264)

[**1.3 Mục tiêu** 5](#_Toc202566265)

[**1.4 Phạm vi nghiên cứu** 6](#_Toc202566266)

[**CHƯƠNG 2. CƠ SỞ LÝ THUYẾT** 7](#_Toc202566267)

[**2.1 Đặt vấn đề và phân tích yêu cầu bài toán** 7](#_Toc202566268)

[**2.1.1 Đặt vấn đề** 7](#_Toc202566269)

[**2.1.2 Phân tích yêu cầu** 7](#_Toc202566270)

[**2.1.3 Tính cấp thiết của bảo mật trong giao dịch số** 7](#_Toc202566271)

[**2.2 Mô tả thuật toán và luồng xử lý** 8](#_Toc202566272)

[**2.2.1 Thuật toán sử dụng** 8](#_Toc202566273)

[**2.2.2 Luồng xử lý** 8](#_Toc202566274)

[**2.3 Cơ sở lý thuyết** 10](#_Toc202566275)

[**2.3.1 Chữ ký số** 10](#_Toc202566276)

[**2.3.2. Hợp đồng điện tử** 11](#_Toc202566277)

[**2.3.3. Giao thức bảo mật** 12](#_Toc202566278)

[**2.3.4 Thuật toán mã hóa đối xứng – Triple DES (3DES)** 12](#_Toc202566279)

[**2.3.5 Thuật toán mã hóa bất đối xứng – RSA** 16](#_Toc202566280)

[**2.3.6 Hàm băm SHA-512** 17](#_Toc202566281)

[**2.3.7. Mã hóa Base64** 19](#_Toc202566282)

[**2.3.8 Các cơ sở lý thuyết khác** 20](#_Toc202566283)

[**CHƯƠNG 3: THIẾT KẾ VÀ TRIỂN KHAI HỆ THỐNG** 23](#_Toc202566284)

[**3.1 Kiến trúc tổng thể của hệ thống** 23](#_Toc202566285)

[**3.2 Các thành phần chính của hệ thống** 24](#_Toc202566286)

[**3.3 Quy trình hoạt động của hệ thống** 26](#_Toc202566287)

[**3.4. Môi trường thực hiện** 28](#_Toc202566288)

[**3.5. Giao diện ứng dụng** 29](#_Toc202566289)

[**3.6 Cấu trúc thu mục file nguồn** 30](#_Toc202566290)

[**3.7. Code và giao diện receiver\_app.py** 32](#_Toc202566291)

[**3.8 Code và giao diện sender\_app.py** 36](#_Toc202566292)

[**3.9 Code utils.py** 42](#_Toc202566293)

[**3.10 Code verify\_tool.py** 47](#_Toc202566294)

[**CHƯƠNG 4. KẾT LUẬN** 50](#_Toc202566295)

[**4.1. Đánh giá hiệu quả** 50](#_Toc202566296)

[**4.2. Phân tích, nhận xét đặc điểm của các thuật toán được sử dụng** 51](#_Toc202566297)

[**4.2.1. Triple DES (3DES)** 51](#_Toc202566298)

[**4.2.2. RSA 2048-bit (PKCS#1 v1.5 + SHA-512)** 51](#_Toc202566299)

[**4.2.3. SHA-512** 52](#_Toc202566300)

[**4.3. Đề xuất cải tiến và hướng phát triển** 53](#_Toc202566301)

[**4.4 Kết Quả Đạt Được** 54](#_Toc202566302)

[**LỜI CẢM ƠN** 56](#_Toc202566303)

[**TÀI LIỆU THAM KHẢO** 57](#_Toc202566304)

# **CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI**

## **1.1 Giới thiệu chung về đề tài lựa chọn**

Trong bối cảnh chuyển đổi số đang diễn ra mạnh mẽ, việc trao đổi thông tin và dữ liệu điện tử ngày càng trở nên phổ biến trong các tổ chức, doanh nghiệp. Đặc biệt, các tài liệu quan trọng như hợp đồng kinh tế, hợp đồng lao động, hợp đồng dịch vụ,… đang dần được số hóa và gửi qua mạng Internet thay cho hình thức in ấn truyền thống. Tuy nhiên, việc truyền tải những tài liệu quan trọng này qua mạng tiềm ẩn nhiều rủi ro về bảo mật, tính toàn vẹn và xác thực nguồn gốc của văn bản.

Chữ ký số riêng (private digital signature) được xem là một giải pháp hiệu quả để giải quyết các vấn đề trên. Khi một hợp đồng được ký bằng chữ ký số riêng, người nhận có thể xác thực được nguồn gốc, đảm bảo tính toàn vẹn và chống chối bỏ từ người gửi. Chính vì vậy, nhóm đã lựa chọn đề tài "Gửi hợp đồng với chữ ký số riêng" nhằm tìm hiểu, ứng dụng và phát triển một hệ thống giúp gửi – nhận hợp đồng một cách an toàn và đáng tin cậy.

## **1.2 Lí do chọn đề tài**

* Giao dịch điện tử ngày càng phổ biến, nhưng nếu không có cơ chế đảm bảo an toàn, các hợp đồng gửi qua mạng có thể bị giả mạo, chỉnh sửa hoặc đánh cắp. Điều này dẫn đến nguy cơ tranh chấp pháp lý hoặc mất mát thông tin nghiêm trọng.
* Chữ ký số đóng vai trò then chốt trong việc xác thực danh tính và bảo đảm tính toàn vẹn cho văn bản điện tử. Nhóm mong muốn ứng dụng kiến thức về mã hóa bất đối xứng và xác thực để giải quyết bài toán thực tiễn này.
* Đề tài có tính thực tiễn cao, phù hợp với xu hướng chuyển đổi số trong quản lý, hành chính và kinh doanh. Ngoài ra, nhóm cũng muốn nâng cao kỹ năng triển khai thuật toán bảo mật như RSA, SHA, và kiến thức về trao đổi khóa.
* Đây là cơ hội để nhóm tiếp cận và thực hành xây dựng một ứng dụng truyền tải hợp đồng có chữ ký số riêng – một kỹ năng quan trọng trong ngành bảo mật thông tin.

## **1.3 Mục tiêu**

Mục tiêu nghiên cứu của đề tài:

* Xây dựng một hệ thống gửi hợp đồng điện tử giữa hai bên có áp dụng chữ ký số riêng để xác thực và bảo vệ tài liệu.
* Thực hiện ký số hợp đồng bằng thuật toán bất đối xứng (RSA) kết hợp với hàm băm (SHA-256) để đảm bảo tính xác thực và toàn vẹn.
* Mã hóa hợp đồng trước khi gửi nhằm đảm bảo tính bảo mật trong quá trình truyền tải.
* Kiểm tra tính toàn vẹn của hợp đồng nhận được bằng cách xác minh chữ ký số và nội dung văn bản.
* Ghi nhận thời gian thực hiện ký – gửi – nhận – xác minh và đánh giá hiệu quả thuật toán.
* Đề xuất giải pháp cải tiến nhằm nâng cao tính bảo mật và tối ưu hiệu suất của hệ thống..

## **1.4 Phạm vi nghiên cứu**

Phạm vi nghiên cứu của đề tài:

* Nghiên cứu lý thuyết và triển khai thực nghiệm việc ký số và truyền tải hợp đồng trong môi trường mô phỏng giữa 2 máy (Client – Server).
* Ứng dụng thuật toán mã hóa RSA và hàm băm SHA-256 để ký và xác minh hợp đồng.
* Gửi file hợp đồng (PDF hoặc TXT) dưới dạng mã hóa, có đính kèm chữ ký số riêng
* Không tập trung vào giao diện người dùng thương mại mà ưu tiên triển khai chức năng cốt lõi của hệ thống.
* Thử nghiệm gửi các file hợp đồng có dung lượng khác nhau, ghi nhận độ trễ, đánh giá độ chính xác và toàn vẹn dữ liệu.
* Hạn chế nghiên cứu: Chưa triển khai tích hợp với các hệ thống chữ ký số công cộng (CA – Certificate Authority), tập trung vào mô hình ký số riêng tư.

# **CHƯƠNG 2. CƠ SỞ LÝ THUYẾT**

## **2.1 Đặt vấn đề và phân tích yêu cầu bài toán**

### **2.1.1 Đặt vấn đề**

Trong bối cảnh giao dịch điện tử ngày càng phổ biến, việc gửi hợp đồng điện tử đòi hỏi đảm bảo tính bảo mật, xác thực và toàn vẹn dữ liệu. Đề tài tập trung vào việc xây dựng hệ thống gửi file hợp đồng (contract.txt) với chữ ký số riêng, chia file thành ba phần, mỗi phần được mã hóa và ký số để đảm bảo an toàn trong quá trình truyền tải.

### **2.1.2 Phân tích yêu cầu**

* Mã hóa dữ liệu: Sử dụng thuật toán Triple DES để mã hóa từng phần của file hợp đồng, đảm bảo tính bảo mật trong quá trình truyền tải.
* Xác thực người dùng: Sử dụng RSA 2048-bit (PKCS#1 v1.5 + SHA-512) để ký số metadata (tên file, timestamp, kích thước) và trao đổi khóa phiên (SessionKey), đảm bảo chỉ người dùng đúng quyền có thể gửi và nhận dữ liệu.
* Kiểm tra tính toàn vẹn: Sử dụng hàm băm SHA-512 để kiểm tra tính toàn vẹn của từng phần file, tránh tình trạng dữ liệu bị thay đổi hoặc giả mạo trong quá trình truyền tải.

### **2.1.3 Tính cấp thiết của bảo mật trong giao dịch số**

Trong thời đại chuyển đổi số hiện nay, việc trao đổi dữ liệu số trong các hoạt động thương mại, tài chính, y tế và hành chính đã trở thành xu thế tất yếu. Tuy nhiên, đi cùng với lợi ích to lớn về hiệu quả và tốc độ là những rủi ro đáng kể về an ninh thông tin. Các cuộc tấn công mạng nhằm vào dữ liệu hợp đồng, giao dịch ngân hàng, thông tin cá nhân ngày càng tinh vi, có thể gây thiệt hại nghiêm trọng về tài chính và uy tín. Do đó, việc đảm bảo các yêu cầu về **bảo mật, xác thực, và toàn vẹn dữ liệu** trong giao tiếp điện tử không chỉ là nhu cầu kỹ thuật mà còn là đòi hỏi cấp thiết về pháp lý và đạo đức.

Hệ thống đề xuất trong đồ án với cơ chế **mã hóa từng phần, ký số từng gói tin, kiểm tra băm và xác minh khóa phiên** là một hướng tiếp cận phù hợp, giúp mô phỏng gần sát các mô hình đang được sử dụng trong thực tế như hệ thống **ký hợp đồng điện tử ViettelSign, MISA eSign, VNPT eContract.**

## **2.2 Mô tả thuật toán và luồng xử lý**

### **2.2.1 Thuật toán sử dụng**

**Triple DES**:

* Là thuật toán mã hóa đối xứng, sử dụng ba khóa 56-bit (tổng 168-bit) để mã hóa dữ liệu theo ba giai đoạn, tăng cường bảo mật so với DES.
* Đầu vào: Dữ liệu plaintext, khóa 168-bit, vector khởi tạo (IV).
* Đầu ra: Dữ liệu ciphertext.

**RSA 2048-bit (PKCS#1 v1.5 + SHA-512)**:

* RSA là thuật toán mã hóa bất đối xứng, sử dụng cặp khóa công khai và bí mật (2048-bit).
* PKCS#1 v1.5 là chuẩn đệm dữ liệu, kết hợp với SHA-512 để tạo chữ ký số và mã hóa khóa phiên.

**SHA-512**:

* Hàm băm tạo ra giá trị băm 512-bit để kiểm tra tính toàn vẹn của dữ liệu.
* Được sử dụng để băm metadata và từng phần file

### **2.2.2 Luồng xử lý**

**Handshake**

* Người gửi gửi tin nhắn "Hello!" để khởi tạo kết nối.
* Người nhận trả lời "Ready!" để xác nhận sẵn sàng nhận dữ liệu.

**Xác thực và trao khóa**

* Người gửi tạo metadata (tên file: "contract.txt", timestamp, kích thước file) và ký số metadata bằng RSA/SHA-512 sử dụng khóa bí mật.
* Tạo khóa phiên (SessionKey) ngẫu nhiên cho Triple DES.
* Mã hóa SessionKey bằng RSA 2048-bit (PKCS#1 v1.5) với khóa công khai của người nhận và gửi đi.

**Mã hóa và kiểm tra toàn vẹn**

* Tạo vector khởi tạo (IV) ngẫu nhiên cho Triple DES.
* Chia file contract.txt thành ba phần bằng nhau.
* Mã hóa từng phần bằng Triple DES với SessionKey và IV.
* Tính giá trị băm SHA-512(IV || ciphertext) cho từng phần.
* Ký số giá trị băm bằng RSA/SHA-512.
* Gói tin gửi cho mỗi phần:

**Phía người nhận**

* Kiểm tra chữ ký số của metadata bằng khóa công khai của người gửi.
* Giải mã SessionKey bằng RSA 2048-bit (khóa bí mật).
* Đối với mỗi phần:
  + Kiểm tra chữ ký số của giá trị băm (SHA-512) bằng khóa công khai.
  + Tính lại SHA-512(IV || ciphertext) và so sánh với giá trị băm nhận được.
* Nếu tất cả hợp lệ:
  + Giải mã từng phần bằng Triple DES với SessionKey và IV.
  + Ghép các phần để tạo lại file contract.txt.
  + Gửi tin nhắn ACK tới người gửi.
* Nếu chữ ký hoặc băm không hợp lệ:
  + Gửi tin nhắn NACK (lỗi integrity) tới người gửi.

## **2.3 Cơ sở lý thuyết**

### **2.3.1 Chữ ký số**

**a) Khái niệm**

Chữ ký số là một dạng thông tin điện tử được tạo ra bằng cách sử dụng hệ thống mã hóa khóa công khai (Public Key Infrastructure – PKI). Nó cho phép người gửi “ký” vào dữ liệu số (như tài liệu, hợp đồng điện tử) để xác minh danh tính và đảm bảo tính toàn vẹn của dữ liệu.

**b) Vai trò và ứng dụng của chữ ký số**

* Xác thực (Authentication): Giúp xác minh danh tính của người ký.
* Toàn vẹn dữ liệu (Integrity): Đảm bảo nội dung tài liệu không bị thay đổi sau khi ký.
* Không chối bỏ (Non-repudiation): Người ký không thể phủ nhận đã ký vào văn bản.
* Ứng dụng thực tiễn: Giao dịch tài chính, khai báo thuế điện tử, ký kết hợp đồng, văn bản nội bộ doanh nghiệp, văn bản hành chính.

**c) Các thành phần chính của chữ ký số**

* Khóa bí mật (Private key): Dùng để tạo chữ ký số. Chỉ người ký giữ.
* Khóa công khai (Public key): Dùng để xác minh chữ ký số. Có thể chia sẻ công khai.
* Chứng chỉ số (Digital Certificate): Là minh chứng được cấp bởi tổ chức cung cấp chứng thực (CA – Certification Authority), xác nhận tính hợp pháp của cặp khóa.

**d, Chữ ký số nâng cao**

* Tính năng: xác thực, toàn vẹn, không chối bỏ.
* Các yếu tố nâng cao:
  + **Timestamp Authority (TSA):** Bổ sung dấu thời gian tin cậy.
  + **OCSP/CRL:** Kiểm tra hiệu lực chứng chỉ.
  + **Chữ ký lồng:** Ký theo từng phần của file.

### **2.3.2. Hợp đồng điện tử**

**a) Định nghĩa**

Hợp đồng điện tử là hợp đồng được thiết lập dưới dạng thông điệp dữ liệu, có giá trị pháp lý tương đương hợp đồng giấy theo quy định tại Luật Giao dịch điện tử của Việt Nam (2005). Việc ký kết, gửi và lưu trữ được thực hiện thông qua phương tiện điện tử.

**b) Đặc điểm**

* Không cần gặp mặt trực tiếp.
* Nhanh chóng, tiết kiệm chi phí in ấn và vận chuyển.
* Có thể tích hợp kiểm tra và xử lý tự động.
* Yêu cầu đảm bảo tính pháp lý, an toàn, xác thực và toàn vẹn.

**c) Yêu cầu pháp lý và kỹ thuật**

* Pháp lý: Theo quy định của pháp luật, hợp đồng điện tử có hiệu lực khi được sự đồng thuận của các bên và sử dụng chữ ký số hợp lệ.
* Kỹ thuậ**t:** Hợp đồng cần đảm bảo không bị sửa đổi nội dung sau khi ký, khả năng xác minh người ký và tính không thể chối bỏ.

**d, Luật pháp liên quan đến hợp đồng điện tử**

* Theo Luật Giao dịch điện tử 2005, hợp đồng điện tử hợp lệ khi:

Được lưu dưới dạng truy cập được.

* Có đồng thuận của hai bên.
* Ký bằng chữ ký số hợp pháp (Nđ 130/2018/NĐ-CP).

### **2.3.3. Giao thức bảo mật**

**a) Giao thức truyền tải an toàn**

* HTTPS (HTTP Secure): Giao thức truyền tải dữ liệu trên nền HTTP với lớp mã hóa SSL/TLS.
* TLS/SSL (Transport Layer Security / Secure Socket Layer): Giao thức mã hóa bảo vệ dữ liệu trong quá trình truyền qua mạng, đảm bảo tính bảo mật và xác thực giữa máy khách và máy chủ.

**b) Quy trình xác thực và bảo vệ dữ liệu**

1. Băm nội dung hợp đồng bằng thuật toán như SHA-256.
2. Tạo chữ ký số bằng cách mã hóa bản băm bằng khóa bí mật của người ký.
3. Gửi hợp đồng kèm chữ ký số qua kênh bảo mật như HTTPS.
4. Bên nhận kiểm tra chữ ký số bằng khóa công khai, so sánh bản băm để xác minh tính toàn vẹn và người gửi.
5. Lưu trữ và ghi log giao dịch nhằm phục vụ tra cứu, đảm bảo không chối bỏ.

### **2.3.4 Thuật toán mã hóa đối xứng – Triple DES (3DES)**

Triple DES (hay 3DES hoặc TDES) là một thuật toán mã hóa khối, sử dụng thuật toán DES (Tiêu chuẩn mã hóa dữ liệu) ba lần để mã hóa dữ liệu. Thay vì chỉ sử dụng một lần mã hóa DES, Triple DES sử dụng ba lần mã hóa, mỗi lần sử dụng một khóa khác nhau, để tăng cường độ an toàn của quá trình mã hóa.

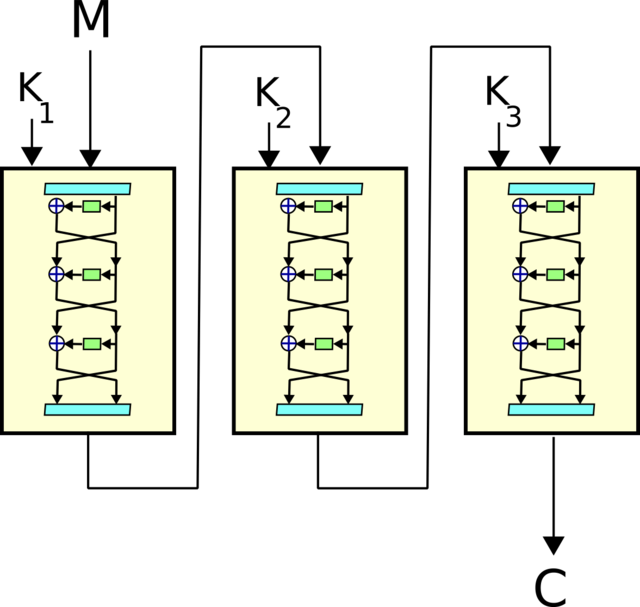
Cụ thể, Triple DES hoạt động bằng cách:

* Mã hóa bằng khóa thứ nhất: Dữ liệu đầu vào được mã hóa bằng khóa đầu tiên (K1) sử dụng thuật toán DES.
* Giải mã bằng khóa thứ hai: Sau đó, dữ liệu đã được mã hóa ở bước trước sẽ được giải mã bằng khóa thứ hai (K2).
* Mã hóa bằng khóa thứ ba: Cuối cùng, dữ liệu đã được giải mã ở bước hai lại được mã hóa bằng khóa thứ ba (K3).

Có hai phiên bản Triple DES:

* Triple DES với ba khóa: Sử dụng ba khóa độc lập (K1, K2, K3).
* Triple DES với hai khóa: Sử dụng hai khóa độc lập, trong đó khóa thứ ba (K3) giống với khóa thứ nhất (K1).

Triple DES được tạo ra để khắc phục các nhược điểm của DES, đặc biệt là độ dài khóa ngắn (56 bit) của DES, khiến nó dễ bị tấn công bằng phương pháp vét cạn (brute-force). Việc sử dụng ba khóa trong Triple DES giúp tăng độ dài khóa hiệu quả lên 112 bit (với phiên bản hai khóa) hoặc 168 bit (với phiên bản ba khóa), làm cho việc tấn công trở nên khó khăn hơn nhiều.



**a) Nguyên lý hoạt động**

Cách thức hoạt động của Triple DES: DES hoạt động bằng cách mã hóa, giải mã rồi mã hóa lại một khối dữ liệu 64-bit, sử dụng ba khóa 56-bit khác nhau (hoặc hai khóa, trong đó một khóa được sử dụng hai lần).

Cách thức hoạt động chi tiết:

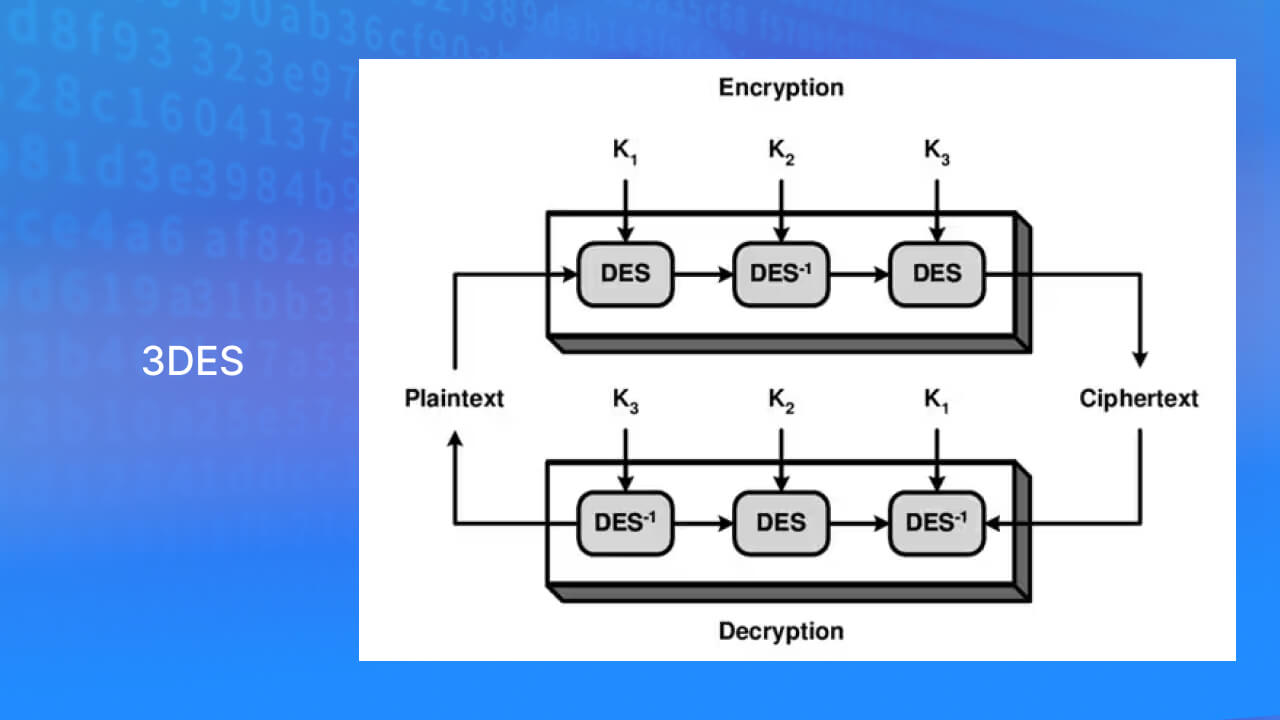
1. Chia dữ liệu thành các khối: Dữ liệu cần mã hóa được chia thành các khối 64 bit.

2. Mã hóa bằng khóa đầu tiên: Khối dữ liệu 64 bit được mã hóa bằng khóa đầu tiên (K1) sử dụng thuật toán DES.

3. Giải mã bằng khóa thứ hai: Dữ liệu đã được mã hóa ở bước trên sau đó được giải mã bằng khóa thứ hai (K2).

4. Mã hóa lại bằng khóa thứ ba: Cuối cùng, dữ liệu đã được giải mã được mã hóa lại bằng khóa thứ ba (K3).

5. Kết quả: Kết quả của quá trình này là một khối dữ liệu 64 bit đã được mã hóa ba lần, mang lại độ an toàn cao hơn so với DES thông thường.



**b) Ưu điểm và nhược điểm**

Ưu điểm của Triple DES:

* Bảo mật tốt hơn DES: 3DES sử dụng 3 lần mã hóa DES, do đó tăng cường đáng kể khả năng chống lại các cuộc tấn công brute-force so với DES một lần mã hóa.
* Dễ triển khai: 3DES khá dễ dàng để triển khai cả phần cứng và phần mềm.
* Tính tương thích rộng rãi: 3DES đã từng được sử dụng rộng rãi trong nhiều hệ thống và giao thức, do đó có tính tương thích cao.

Nhược điểm của Triple DES:

* Tốc độ chậm: Quá trình mã hóa ba lần làm cho 3DES chậm hơn so với các thuật toán mã hóa hiện đại như AES, đặc biệt là khi xử lý lượng dữ liệu lớn.
* Kích thước khối nhỏ: Kích thước khối 64-bit của 3DES có thể là một hạn chế trong một số ứng dụng có tốc độ cao, đặc biệt là trong môi trường thế kỷ 21.
* Bị loại bỏ dần: NIST (Viện Tiêu chuẩn và Công nghệ Quốc gia Hoa Kỳ) đã loại bỏ Triple DES và khuyến nghị chuyển sang sử dụng AES.

c, Ứng dung

Trong thực tế, Triple DES từng được sử dụng rộng rãi trong nhiều hệ thống tài chính và an toàn thông tin, đặc biệt là trong các thiết bị và giao thức cũ. Một số ví dụ về ứng dụng cụ thể của Triple DES gồm:

* Mã hóa dữ liệu trên thẻ ngân hàng (thẻ ATM, thẻ tín dụng).
* Hệ thống POS (Point of Sale) trong các cửa hàng, siêu thị.
* Giao thức bảo mật truyền thông cũ như SSL/TLS phiên bản sớm.
* Mã hóa thông tin cá nhân trong cơ sở dữ liệu, đặc biệt trong các hệ thống cần tính tương thích với DES hoặc không yêu cầu tốc độ xử lý quá cao.

Trong khuôn khổ đề tài này, Triple DES được sử dụng để mã hóa các trường nhạy cảm như số căn cước công dân (CCCD) nhằm đảm bảo rằng thông tin người dùng không thể bị đọc trực tiếp kể cả khi dữ liệu bị đánh cắp. Sự kết hợp Triple DES với AES cho phép hệ thống vừa duy trì khả năng bảo mật mạnh, vừa phân tách linh hoạt dữ liệu theo mức độ nhạy cảm, qua đó tối ưu hiệu quả bảo vệ thông tin trong cơ sở dữ liệu.

**d) Ứng dụng trong đề tài**

* Mỗi phần của file contract.txt sẽ được mã hóa bằng Triple DES sử dụng cùng session key và IV riêng.
* IV sẽ được tạo ngẫu nhiên cho mỗi phần và gửi kèm trong gói tin.

### **2.3.5 Thuật toán mã hóa bất đối xứng – RSA**

**a) Nguyên lý hoạt động**

RSA sử dụng cặp khóa công khai – bí mật:

* Mã hóa: C = M^e mod n (dùng public key)
* Giải mã: M = C^d mod n (dùng private key)

Trong đó:

* e, n: khóa công khai
* d, n: khóa bí mật
* M: bản rõ
* C: bản mã

**b) Ứng dụng**

* Ký số: Người gửi dùng private key để tạo chữ ký cho metadata và từng phần dữ liệu.
* Mã hóa session key**:** Session key dùng cho Triple DES sẽ được mã hóa bằng public key của người nhận → đảm bảo chỉ người nhận giải mã được.

**c) Chuẩn ký số PKCS#1 v1.5 + SHA-512**

* PKCS#1 v1.5: Chuẩn padding cổ điển, đơn giản nhưng dễ triển khai.
* SHA-512: Dùng để băm nội dung trước khi ký.
* Chữ ký số sẽ được tạo bằng công thức: Signature = RSA\_sign(SHA-512(data))

### **2.3.6 Hàm băm SHA-512**

**a, Khái niệm hàm băm (Hash Function)**

Hàm băm mật mã là một thuật toán toán học nhận đầu vào là một chuỗi dữ liệu có độ dài tùy ý và trả về một chuỗi giá trị băm (hash value) có độ dài cố định. Hàm băm có nhiều ứng dụng trong bảo mật như lưu trữ mật khẩu, xác thực dữ liệu, tạo chữ ký số, v.v.

**Thuộc tính cơ bản của một hàm băm mật mã an toàn:**

* Xác định (Deterministic): Một đầu vào luôn cho ra cùng một giá trị băm.
* Không thể đảo ngược (Pre-image resistance): Không thể tìm được đầu vào từ giá trị băm.
* Chống va chạm (Collision resistance): Khó có thể tìm hai đầu vào khác nhau cho cùng một giá trị băm.
* Nhạy cảm dữ liệu (Avalanche effect): Thay đổi nhỏ ở đầu vào dẫn đến thay đổi lớn ở đầu ra.

**b, Thuật toán SHA-512**

**Giới thiệu**

SHA-512 là một thành viên trong họ thuật toán SHA-2 (gồm SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256), được phát triển bởi NSA và tiêu chuẩn hóa bởi NIST vào năm 2001.

* Đầu vào: Chuỗi dữ liệu có độ dài tùy ý.
* Đầu ra: Chuỗi băm dài 512 bit (64 byte).
* Mức độ an toàn: Cao hơn SHA-1 và SHA-256, phù hợp cho các ứng dụng yêu cầu mức độ bảo mật mạnh.

**Cấu trúc hoạt động**

SHA-512 hoạt động theo mô hình Merkle–Damgård, bao gồm các bước sau:

1. Tiền xử lý (Preprocessing):
   * Padding dữ liệu đầu vào để đạt độ dài bội số của 1024 bit.
   * Thêm độ dài gốc của thông điệp vào cuối bản tin.
2. Khởi tạo giá trị băm ban đầu (Initial hash values):
   * Sử dụng 8 hằng số 64-bit được định nghĩa trước.
3. Chia dữ liệu thành các khối 1024 bit và xử lý:
   * Mỗi khối được xử lý qua 80 vòng lặp (round).
   * Dùng các phép toán logic như: AND, OR, XOR, NOT, ROTR (xoay phải), SHR (dịch phải).
4. Tính toán giá trị băm cuối cùng:
   * Sau khi xử lý tất cả các khối, các giá trị trung gian được ghép lại để tạo ra đầu ra cuối cùng dài 512 bit.

**c, Đặc điểm kỹ thuật**

* Kích thước khối: 1024 bit
* Kích thước từ xử lý: 64 bit
* Số vòng lặp: 80
* Kích thước đầu ra: 512 bit

**d, Ưu điểm và nhược điểm của SHA-512**

**Ưu điểm:**

* Mức độ bảo mật cao, chống lại các tấn công sinh va chạm hiệu quả hơn SHA-1 và MD5.
* Phù hợp với kiến trúc máy tính 64-bit, xử lý nhanh trên hệ thống hiện đại.
* Thích hợp cho các ứng dụng yêu cầu độ tin cậy cao như ký số, blockchain, mã hóa file...

**Nhược điểm:**

* Kích thước đầu ra lớn hơn (512 bit) dẫn đến tiêu tốn bộ nhớ và băng thông khi truyền dữ liệu.
* Tốc độ chậm hơn SHA-256 trên hệ thống 32-bit.

**e, Ứng dụng thực tế**

* Lưu trữ mật khẩu (kết hợp với Salt) trong hệ thống đăng nhập.
* Tạo và kiểm tra chữ ký số trong giao tiếp bảo mật.
* Kiểm tra tính toàn vẹn dữ liệu trong truyền tải và lưu trữ.
* Áp dụng trong các hệ thống như Blockchain (Bitcoin dùng SHA-256, nhưng các ứng dụng nâng cao có thể dùng SHA-512).

**g, Ứng dụng trong đề tài**

* Tạo hàm băm để kiểm tra tính toàn vẹn: hash = SHA-512(IV || ciphertext)
* Người nhận dùng hàm băm lại để đối chiếu với hash được gửi kèm trong gói tin.

### **2.3.7. Mã hóa Base64**

**a) Khái niệm**

Base64 là kỹ thuật mã hóa dữ liệu nhị phân sang dạng văn bản ASCII để đảm bảo dễ dàng truyền qua mạng hoặc lưu trữ.

**b) Ứng dụng**

* Mã hóa các thành phần: iv, cipher, signature trong gói tin JSON.
* Giúp dữ liệu không bị lỗi khi gửi qua HTTP hoặc các giao thức chỉ hỗ trợ văn bả

**2.3.8 Các cơ sở lý thuyết khác**

**Các kiểu tấn công vào hệ thống truyền tin số**

Việc truyền tải file hợp đồng điện tử qua mạng có thể đối mặt với nhiều kiểu tấn công phổ biến, bao gồm:

* **Tấn công sửa đổi dữ liệu (Data Modification):** Thay đổi nội dung khi truyền.
* **Tấn công giả mạo (Impersonation):** Giả vờ người gửi/nhận.
* **Tấn công lặp lại (Replay Attack):** Gửi lại gói tin cũ.
* **Tấn công xen giữa (Man-in-the-Middle):** Chặn và thay đổi dữ liệu.

Giải pháp: Sử dụng chữ ký số + hash + mã hóa + xác thực giảm thiểu nguy cơ.

**Mã hóa nhiều lớp (Hybrid Encryption)**

* Kết hợp mã hóa bất đối xứng (RSA) và đối xứng (Triple DES).
* RSA dùng để mã hóa SessionKey (khóa phiên).
* Triple DES dùng để mã hóa file.
* Mô hình này giúp đảm bảo tốc độ và bảo mật, được sử dụng trong TLS/SSL, OpenPGP, S/MIME.

**Cơ chế sinh và quản lý khóa**

* SessionKey tạo mới cho mỗi lần truyền.
* IV tạo ngẫu nhiên cho từng phần.
* RSA private key lưu trữ an toàn (HSM, token).
* Cần xác thực khóa public bằng chứng chỉ hoặc fingerprint.

**Quản lý khóa và trao đổi khóa**

* Dùng RSA để mã hóa SessionKey.
* Trao đổi RSA Public/Private an toàn.
* Khóa phải có vòng đời quản lý (thu hồi, thay mới, hết hạn).

**Phòng chống tấn công lặp lại**

* Kẻ xấu ghi lại và gửi lại gói tin hợp lệ.
* Giải pháp: Thêm **timestamp** hoặc **nonce** duy nhất mỗi giao dịch.
* Từ chối gói tin quá cũ hoặc bị trùng lặp.

**Mô hình truyền thông hệ thống**

* Mô hình Client-Server truyền qua TCP/IP.
* Giao thức: HTTPS hoặc gói bảo mật tuỳ chỉnh.
* Dữ liệu: PDF/DOCX/JSON...
* Luồng:
  + Client tạo file → ký số → mã hóa → gửi
  + Server nhận → xác minh → lưu trữ

**Kiểm thử hệ thống bảo mật**

* **Kiểm thử chức năng:** mã hóa, ký số...
* **Kiểm thử an toàn:** gửi gói tin lỗi, thay hash, thay khóa...
* **Hiệu năng:** tốc độ mã hóa file lớn, nhiều người dùng.
* **Tấn công thử nghiệm:** MITM, replay, brute-force.

**Các dạng tấn công phổ biến và cách phòng tránh**

|  |  |  |
| --- | --- | --- |
| Tấn công | Mô tả | Phòng tránh |
| MITM | Xen giữa và thay đổi gói tin | Sử dụng RSA, HTTPS |
| Giả mạo khóa | Gửi khóa độc hại | Xác thực bằng CA |
| Brute-force | Thử mọi khóa | Dùng khóa dài (2048 bit) |
| Replay | Gửi lại gói cũ | Timestamp + nonce |
| Sửa file | Thay đổi nội dung | Dùng hash + chữ ký số |

# **CHƯƠNG 3: THIẾT KẾ VÀ TRIỂN KHAI HỆ THỐNG**

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

Hệ thống được xây dựng theo mô hình kiến trúc ba lớp, đảm bảo tính mô đun hóa và dễ quản lý, bao gồm: giao diện người dùng (client), lớp xử lý nghiệp vụ (mã hóa, giải mã và xử lý chữ ký số), và cơ sở dữ liệu (SQLite).

* Giao diện người dùng (Client): Được xây dựng bằng thư viện PyQt5, giao diện này cung cấp các chức năng tương tác cần thiết cho người dùng, bao gồm các thao tác như chọn hợp đồng để gửi, xem trạng thái gửi/nhận, và thông báo kết quả xử lý. Ngoài ra, có thể bao gồm các chức năng cơ bản như đăng nhập, đăng ký nếu hệ thống yêu cầu quản lý người dùng.
* Lớp xử lý nghiệp vụ: Đây là nơi thực hiện các tác vụ chính của hệ thống liên quan đến bảo mật thông tin. Lớp này chịu trách nhiệm cho các quy trình mã hóa và giải mã dữ liệu hợp đồng, tạo và xác minh chữ ký số, cũng như kiểm tra tính toàn vẹn của dữ liệu. Khi người dùng thực hiện gửi hợp đồng hoặc nhận hợp đồng, thông tin sẽ được xử lý qua lớp này trước khi truyền đi hoặc lưu vào cơ sở dữ liệu/hiển thị.
* Cơ sở dữ liệu (SQLite): Được sử dụng để lưu trữ các thông tin cần thiết của hệ thống, có thể bao gồm metadata của hợp đồng, trạng thái giao dịch, hoặc các khóa công khai/riêng tư (nếu được quản lý tại đây).

**Dòng luồng dữ liệu trong hệ thống:**

Dữ liệu trong hệ thống được triển khai theo quy trình bảo mật như sau:

1. Người dùng chọn hợp đồng: Người dùng chọn file hợp đồng cần gửi qua giao diện.
2. Mã hóa và Ký số dữ liệu: Lớp nghiệp vụ sẽ chia hợp đồng thành các phần, mã hóa từng phần bằng Triple DES, sau đó tạo chữ ký số riêng cho mỗi phần và tính toán hash (SHA-512) để đảm bảo tính toàn vẹn.
3. Trao đổi khóa và gửi dữ liệu: Khóa phiên (SessionKey) được mã hóa bằng RSA 2048-bit và trao đổi an toàn. Các phần hợp đồng đã được mã hóa và ký số sẽ được gửi tới người nhận.
4. Lưu vào cơ sở dữ liệu: Các thông tin liên quan đến giao dịch (ví dụ: metadata hợp đồng, trạng thái gửi) có thể được lưu trữ vào cơ sở dữ liệu.
5. Giải mã và xác minh (phía người nhận): Khi cần hiển thị hoặc lưu trữ hợp đồng đã nhận, lớp nghiệp vụ sẽ tiến hành kiểm tra hash và chữ ký số của từng phần. Nếu tất cả đều hợp lệ, các phần sẽ được giải mã bằng Triple DES và ghép lại để trình bày hoặc lưu trữ.

**Thuật toán mã hóa và bảo mật được sử dụng:**

* Triple DES: Sử dụng cho việc mã hóa nội dung các phần của hợp đồng.
* RSA 2048-bit (PKCS#1 v1.5 + SHA-512): Sử dụng cho việc trao đổi khóa phiên và tạo chữ ký số riêng cho từng phần của hợp đồng, đảm bảo tính xác thực và chống chối bỏ.
* SHA-512: Sử dụng để kiểm tra tính toàn vẹn của dữ liệu từng phần hợp đồng sau khi mã hóa

## **3.2 Các thành phần chính của hệ thống**

Hệ thống được tổ chức thành các thành phần độc lập, mỗi thành phần đảm nhận một nhiệm vụ cụ thể, đảm bảo tính mô đun hóa và dễ dàng phát triển:

**Giao diện người dùng (UI):**

Được xây dựng bằng thư viện **tkinter**, cung cấp các cửa sổ tương tác với người dùng.

Các chức năng chính bao gồm:

* Phía Người gửi (sender.py): Lựa chọn file hợp đồng để gửi, thiết lập kết nối, quản lý và hiển thị trạng thái gửi, cho phép tạo/tải các cặp khóa RSA của người gửi và khóa công khai của người nhận.
* Phía Người nhận (receiver.py): Khởi động máy chủ, lắng nghe kết nối, hiển thị danh sách các file đang nhận, quản lý và hiển thị trạng thái nhận, cho phép tạo/tải các cặp khóa RSA của người nhận và khóa công khai của người gửi.
* Giao diện log: Cả hai ứng dụng đều tích hợp khu vực hiển thị log để theo dõi các hoạt động, thông báo lỗi và trạng thái xử lý.

**Các module xử lý chính:**

sender.py: Đây là script chính của ứng dụng Người gửi (Client). Nó quản lý giao diện người dùng để chọn file, khởi tạo kết nối mạng, điều phối quá trình chia nhỏ file, mã hóa (Triple DES), ký số (RSA/SHA-512) và gửi từng phần của hợp đồng qua mạng.

receiver.py: Đây là script chính của ứng dụng Người nhận (Server). Nó thiết lập một máy chủ lắng nghe kết nối, nhận các gói dữ liệu từ Người gửi, thực hiện xác minh chữ ký, kiểm tra tính toàn vẹn (SHA-512), giải mã (Triple DES) các phần dữ liệu và ghép nối chúng để khôi phục hợp đồng gốc.

utils.py: Là module chứa các hàm tiện ích dùng chung cho cả sender.py, receiver.py và verify\_tool.py. Các hàm này bao gồm:

* Tạo và tải các cặp khóa RSA.
* Thực hiện mã hóa và giải mã bằng thuật toán Triple DES (CBC mode).
* Mã hóa và giải mã dữ liệu bằng RSA (PKCS#1 v1.5) (dùng cho trao đổi khóa phiên).
* Tính toán giá trị băm SHA-512.
* Ký số và xác minh chữ ký số sử dụng RSA (PKCS#1 v1.5 với SHA-512).
* Các hàm hỗ trợ truyền nhận dữ liệu qua socket.

verify\_tool.py: Một công cụ độc lập, được phát triển để hỗ trợ xác minh chữ ký và tính toàn vẹn của một file hoặc một hash cho mục đích kiểm tra và gỡ lỗi offline. Nó sử dụng các hàm mật mã từ utils.py

## **3.3 Quy trình hoạt động của hệ thống**

Quy trình hoạt động của hệ thống gửi hợp đồng với chữ ký số riêng được thiết kế để đảm bảo tính bảo mật, toàn vẹn và xác thực của tài liệu contract.txt trong quá trình truyền tải giữa Người gửi và Người nhận. Hệ thống này tập trung vào các bước giao tiếp mật mã chính, được hỗ trợ bởi các hàm trong file utils.py và triển khai trong sender.py và receiver.py, đồng thời ghi lại các hành động quan trọng để phục vụ mục đích kiểm tra và truy vết.

**Quy trình được chia thành các giai đoạn chính như sau:**

**Giai đoạn Bắt tay (Handshake):**

* Người gửi (ứng dụng sender.py) khởi tạo kết nối TCP với Người nhận. Đây là bước đầu tiên để thiết lập kênh giao tiếp an toàn.
* Người nhận (ứng dụng receiver.py) lắng nghe các kết nối đến. Khi nhận được yêu cầu kết nối, Người nhận chấp nhận và thiết lập một phiên giao tiếp riêng biệt cho client đó.
* *Trong triển khai:* Việc thiết lập kết nối socket trong sender.py (trong hàm connect\_to\_receiver hoặc tương tự) và receiver.py (trong hàm start\_server và handle\_client) đại diện cho giai đoạn này. Các thông điệp "Hello!" và "Ready!" sẽ được trao đổi thông qua các gói dữ liệu sử dụng hàm send\_data\_packet và receive\_data\_packet từ utils.py.

**Giai đoạn Xác thực & Trao khóa phiên:**

* Người gửi chuẩn bị metadata của file hợp đồng (bao gồm tên file, timestamp và kích thước file). Để đảm bảo tính xác thực và không thể chối bỏ, Người gửi sử dụng khóa riêng tư RSA của mình để ký số metadata này, với hàm sign\_data từ utils.py sử dụng SHA-512 (PKCS#1 v1.5).
* Một khóa phiên đối xứng (SessionKey) cho thuật toán Triple DES và một Initialization Vector (IV) cho phiên sẽ được tạo ngẫu nhiên bởi Người gửi (sử dụng get\_random\_bytes từ utils.py). SessionKey này được mã hóa bằng khóa công khai RSA của Người nhận (sử dụng rsa\_encrypt từ utils.py) để đảm bảo bí mật trong quá trình truyền tải.
* Người gửi sau đó gửi metadata đã ký, SessionKey đã mã hóa và IV đến Người nhận thông qua các gói dữ liệu.
* Người nhận tiếp nhận các thông tin này. Người nhận sẽ giải mã SessionKey bằng khóa riêng tư RSA của mình (sử dụng rsa\_decrypt từ utils.py) và xác minh chữ ký số của metadata bằng khóa công khai RSA của Người gửi (sử dụng verify\_signature từ utils.py). Quá trình này xác thực danh tính của Người gửi và thiết lập khóa đối xứng bí mật để mã hóa/giải mã dữ liệu hợp đồng.

**Giai đoạn Mã hóa & Kiểm tra Toàn vẹn dữ liệu hợp đồng:**

* Người gửi đọc nội dung của file contract.txt. File này được chia thành 3 phần riêng biệt.
* Mỗi phần sẽ được mã hóa độc lập bằng thuật toán Triple DES sử dụng SessionKey và IV đã thiết lập (sử dụng encrypt\_triple\_des từ utils.py).
* Để đảm bảo tính toàn vẹn của từng phần dữ liệu đã mã hóa, Người gửi tính toán một giá trị băm (hash) SHA-512 cho từng cặp (IV || ciphertext) của mỗi phần (sử dụng compute\_sha512 từ utils.py). Giá trị hash này sau đó được ký số bằng khóa riêng tư RSA của Người gửi (sử dụng sign\_data).
* Mỗi phần dữ liệu được đóng gói thành một gói tin JSON bao gồm: iv, cipher (dữ liệu đã mã hóa), hash của (IV || ciphertext), và sig (chữ ký số của hash đó). Các gói tin này được gửi tuần tự tới Người nhận thông qua hàm send\_data\_packet.
* *Trong triển khai:* Các logic chia file, mã hóa, tính hash và ký số cho từng phần sẽ được thực hiện trong sender.py, sử dụng các hàm từ utils.py.

**Giai đoạn Xử lý tại Người nhận:**

* Người nhận (ứng dụng receiver.py) tiếp nhận từng gói tin dữ liệu từ Người gửi. Đối với mỗi gói tin, Người nhận thực hiện các bước kiểm tra quan trọng:

Tính toán lại hash SHA-512 của (IV || ciphertext) nhận được và so sánh với giá trị hash có trong gói tin (sử dụng compute\_sha512).

Xác minh chữ ký số (sig) của hash bằng cách sử dụng khóa công khai RSA của Người gửi (sử dụng verify\_signature).

* **Nếu tất cả các kiểm tra hash và chữ ký số của *mỗi phần* đều hợp lệ:**

Người nhận sẽ tiến hành giải mã từng phần dữ liệu bằng thuật toán Triple DES, sử dụng SessionKey đã trao đổi và IV tương ứng (sử dụng decrypt\_triple\_des từ utils.py).

Các phần dữ liệu đã giải mã sau đó được ghép lại theo đúng thứ tự để khôi phục file contract.txt gốc và lưu trữ vào thư mục received\_files (như trong receiver.py).

Sau khi hoàn tất quá trình nhận và xác minh, Người nhận gửi một thông báo xác nhận (ACK) tới Người gửi (sử dụng send\_data\_packet với trạng thái "OK").

* **Ngược lại, nếu bất kỳ kiểm tra hash hoặc chữ ký số nào không hợp lệ:**

Người nhận sẽ từ chối quá trình truyền tải cho phần đó và gửi một thông báo từ chối (NACK) tới Người gửi (sử dụng send\_data\_packet với trạng thái "ERROR"), chỉ rõ lỗi về tính toàn vẹn hoặc xác thực.

* *Trong triển khai:* Các hàm verify\_signature, compute\_sha512, decrypt\_triple\_des trong receiver.py (thông qua import từ utils.py) đóng vai trò trung tâm trong giai đoạn này.

## **3.4. Môi trường thực hiện**

Hệ thống truyền file an toàn được xây dựng bằng ngôn ngữ lập trình **Python phiên bản 3.13.5**, với sự hỗ trợ của nhiều thư viện mã nguồn mở giúp hiện thực hóa các chức năng bảo mật và giao diện người dùng. Cụ thể:

* **PyCryptodome**: Thư viện mã hóa mạnh mẽ, cung cấp các thuật toán **Triple DES**, **RSA 2048-bit** và **SHA-512**. Đây là nền tảng chính để đảm bảo các yêu cầu về mã hóa, giải mã, ký và xác minh chữ ký số cho dữ liệu và các khóa phiên.
* **Flask**: Được sử dụng để xây dựng **giao diện người dùng web (Web GUI)** cho cả ứng dụng gửi (Sender) và nhận (Receiver). Flask cung cấp một framework nhẹ và linh hoạt, cho phép tương tác dễ dàng thông qua trình duyệt web.
* **socket**: Thư viện tiêu chuẩn của Python, được sử dụng để thiết lập **kết nối mạng (TCP sockets)** giữa ứng dụng gửi và nhận, phục vụ cho việc truyền tải dữ liệu.
* **base64, json, os, datetime, threading, math**: Các thư viện tiêu chuẩn khác của Python được sử dụng để xử lý dữ liệu (chuỗi Base64, định dạng JSON), quản lý tệp và thư mục (os), định dạng thời gian (datetime), xử lý đa luồng (threading) cho server nhận và các phép toán số học (math).

Ứng dụng được phát triển và thử nghiệm trên máy tính cá nhân sử dụng hệ điều hành **Windows**. Do tính chất là ứng dụng web (Flask), nó có khả năng tương thích cao và có thể triển khai trên nhiều môi trường khác nhau có hỗ trợ Python và Flask

## **3.5. Giao diện ứng dụng**

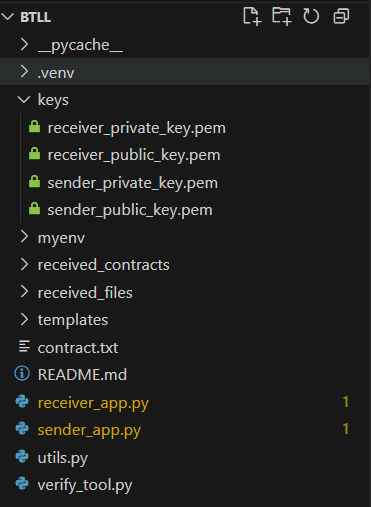
Hệ thống được chia thành hai ứng dụng độc lập: **SenderApp** (phía gửi) và **ReceiverApp** (phía nhận và xác thực). Mỗi ứng dụng đều có giao diện người dùng riêng biệt, được thiết kế để trực quan và dễ sử dụng thông qua trình duyệt web.

Giao diện được xây dựng bằng **Flask**, một micro-framework cho Python, cho phép tạo các ứng dụng web nhẹ và linh hoạt. Người dùng có thể tương tác với các chức năng của ứng dụng (như tạo khóa, tải khóa, kết nối, gửi/nhận file) thông qua các trang web được hiển thị.

Các tính năng chính của giao diện bao gồm:

* **Tải/Tạo khóa RSA**: Cho phép người dùng dễ dàng quản lý các cặp khóa công khai và riêng tư.
* **Kết nối**: Thiết lập kết nối giữa SenderApp và ReceiverApp.
* **Gửi/Nhận file**: Cung cấp các nút và trường để chọn file, gửi đi và hiển thị trạng thái nhận file.
* **Nhật ký hoạt động**: Hiển thị các thông báo chi tiết về quá trình hoạt động của ứng dụng, bao gồm trạng thái kết nối, quá trình trao đổi khóa, và thông tin mã hóa/giải mã của từng phần file.

## **3.6 Cấu trúc thu mục file nguồn**

****

Dự án được tổ chức một cách rõ ràng, phân chia các vai trò và chức năng chính như sau:

**receiver\_app.py:**

Đảm nhiệm vai trò Giao diện và xử lý phía người nhận (Receiver).

Chứa logic khởi động máy chủ Flask, quản lý các kết nối đến, xử lý bắt tay (handshake), trao đổi khóa phiên, nhận và giải mã các phần file, xác minh chữ ký số, và lưu trữ file đã nhận.

Cũng tích hợp chức năng sinh cặp khóa RSA cho người nhận.

**sender\_app.py:**

Đảm nhiệm vai trò Giao diện và xử lý phía người gửi (Sender).

Chứa logic khởi động máy chủ Flask, kết nối đến người nhận, thực hiện bắt tay, trao đổi khóa phiên, chia file thành các phần, mã hóa và ký từng phần, sau đó gửi chúng đi.

Cũng tích hợp chức năng sinh cặp khóa RSA cho người gửi.

**utils.py:**

Đóng vai trò là module chứa các hàm xử lý dùng chung (common functions).

Bao gồm tất cả các hàm mật mã (RSA, Triple DES, SHA512, ký/xác minh chữ ký) và các hàm tiện ích mạng (gửi/nhận gói dữ liệu qua socket) được sử dụng bởi cả sender\_app.py và receiver\_app.py.

**verify\_tool.py:**

Là một công cụ xác minh chữ ký file offline độc lập.

Cho phép người dùng xác minh tính toàn vẹn và xác thực của một file hoặc một hash cụ thể bằng cách sử dụng khóa công khai của người gửi và thông tin hash/chữ ký từ log.

**Thư mục keys:**

Nơi lưu trữ tất cả các tệp khóa RSA (.pem) được tạo ra và sử dụng trong hệ thống:

receiver\_private\_key.pem: Khóa riêng tư của người nhận.

receiver\_public\_key.pem: Khóa công khai của người nhận.

sender\_private\_key.pem: Khóa riêng tư của người gửi.

sender\_public\_key.pem: Khóa công khai của người gửi.

**Thư mục myenv (hoặc .venv):** Chứa môi trường ảo của dự án, đảm bảo các thư viện Python được cài đặt độc lập và không gây xung đột với các dự án khác.

**Thư mục received\_files (hoặc received\_contracts):** Thư mục đích nơi các file đã được người nhận giải mã và xác minh sẽ được lưu trữ.

**Thư mục templates:** Chứa các tệp mẫu HTML cho giao diện web của cả ứng dụng gửi và nhận (ví dụ: sender.html, receiver.html).

**contract.txt:** Một tệp ví dụ hoặc tệp dữ liệu mẫu được sử dụng để kiểm tra chức năng gửi file.

**README.md:** Tệp tài liệu dự án, cung cấp thông tin tổng quan, hướng dẫn sử dụng và các chi tiết khác

## **3.7. Code và giao diện receiver\_app.py**

**Tiêu đề giao diện:** "ỨNG DỤNG NGƯỜI NHẬN HỢP ĐỒNG"

Giao diện ReceiverApp được thiết kế dưới dạng ứng dụng web, cung cấp các chức năng cần thiết để người nhận quản lý khóa, khởi động máy chủ để lắng nghe kết nối, và theo dõi quá trình nhận file hợp đồng một cách bảo mật. Các thành phần chính được tổ chức thành các khối chức năng rõ ràng:

**Thành phần giao diện gồm:**

**Quản Lý Khóa RSA:**

* Nút "Tạo Khóa Mới": Cho phép người dùng sinh ra một cặp khóa RSA mới (khóa riêng tư và khóa công khai) cho người nhận.
* Ô chọn file "Khóa Riêng Tư Người Nhận": Cho phép tải lên tệp khóa riêng tư của người nhận (ví dụ: receiver\_private\_key.pem).
* Ô chọn file "Khóa Công Khai Người Gửi": Cho phép tải lên tệp khóa công khai của người gửi (ví dụ: sender\_public\_key.pem).
* Nút "Tải Khóa" (cho khóa riêng tư): Kích hoạt việc tải khóa riêng tư đã chọn.
* Nút "Tải Khóa" (cho khóa công khai): Kích hoạt việc tải khóa công khai của người gửi đã chọn.

**Khởi Động Server:**

* **Ô nhập "Cổng Server"**: Cho phép nhập số cổng mà server sẽ lắng nghe các kết nối đến (ví dụ: 12345 hoặc 5001).
* **Nút "Bắt Đầu Server"**: Khởi động máy chủ ReceiverApp, cho phép nó lắng nghe và chấp nhận các kết nối từ SenderApp.
* **Nút "Dừng Server"**: Dừng hoạt động của máy chủ ReceiverApp.

**Nhật Ký Hoạt Động:**

* Một khung hiển thị lớn, nơi tất cả các sự kiện và thông báo quan trọng của ứng dụng được ghi lại theo thời gian thực.
* Hiển thị các bước như tải khóa, trạng thái server, chấp nhận kết nối, quá trình bắt tay (handshake), trao đổi khóa phiên, và các thông tin liên quan đến việc nhận và xác minh file.

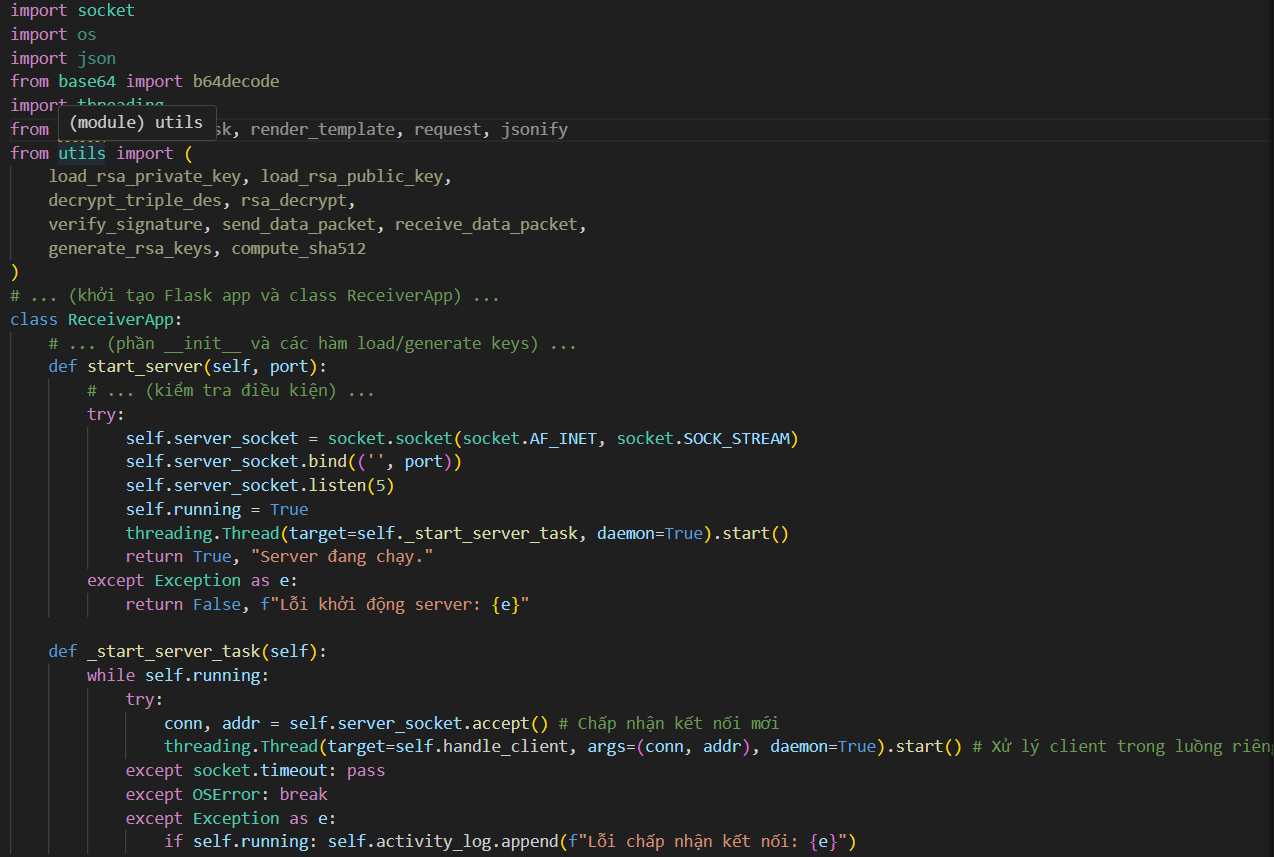
**Các nút chức năng:**

Các nút chức năng được tích hợp trực tiếp vào các khối chức năng tương ứng trên giao diện web:

* "Tạo Khóa Mới": Nút này thực hiện việc sinh khóa.
* "Tải Khóa": Các nút này thực hiện việc tải các tệp khóa đã chọn.
* "Bắt Đầu Server": Nút này khởi động máy chủ.
* "Dừng Server": Nút này dừng máy chủ.

**A screenshot of a computer

AI-generated content may be incorrect.**

****

**Ý nghĩa:**

start\_server: Khởi động máy chủ, cho phép nó lắng nghe và chấp nhận các kết nối đến từ người gửi.

start\_server\_task: Chạy nền để liên tục đón nhận các kết nối mới, mỗi kết nối sẽ được xử lý độc lập trong một luồng riêng

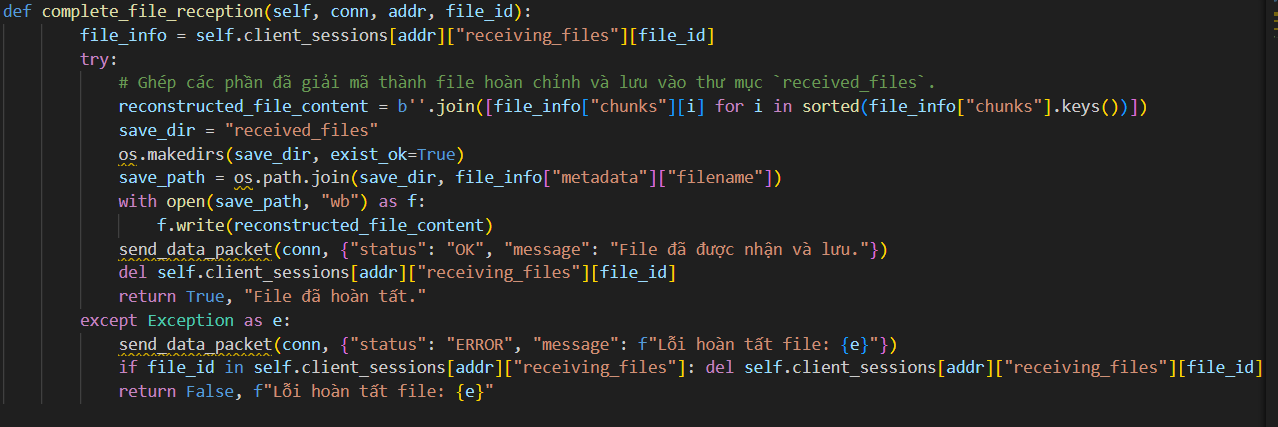
****

**Ý nghĩa:**

handle\_client: Xử lý toàn bộ giao tiếp với một người gửi cụ thể.

key\_exchange: Xác thực thông tin trao đổi khóa bằng chữ ký số và giải mã khóa phiên (session key) bằng RSA.

file\_chunk: Với mỗi phần file nhận được, nó kiểm tra tính toàn vẹn (hash) và xác thực (chữ ký số), sau đó giải mã phần file đó bằng khóa phiên đã thiết lập

****

**Ý nghĩa:**

complete\_file\_reception: Được gọi khi tất cả các phần của một file đã được nhận. Nó có nhiệm vụ sắp xếp, ghép các phần đã giải mã lại và lưu thành file gốc hoàn chỉnh vào hệ thống.

## **3.8 Code và giao diện sender\_app.py**

Tiêu đề giao diện: "ỨNG DỤNG NGƯỜI GỬI HỢP ĐỒNG"

Giao diện SenderApp được thiết kế dưới dạng ứng dụng web, trực quan và dễ sử dụng, cho phép người dùng thực hiện các thao tác liên quan đến việc gửi file hợp đồng một cách bảo mật. Các thành phần chính được tổ chức thành các khối chức năng rõ ràng:

**Thành phần giao diện gồm:**

**Quản Lý Khóa RSA:**

* Nút "Tạo Khóa Mới": Cho phép người dùng sinh ra một cặp khóa RSA mới (khóa riêng tư và khóa công khai) cho người gửi.
* Ô chọn file "Khóa Riêng Tư Người Gửi": Cho phép tải lên tệp khóa riêng tư của người gửi (ví dụ: sender\_private\_key.pem).
* Ô chọn file "Khóa Công Khai Người Nhận": Cho phép tải lên tệp khóa công khai của người nhận (ví dụ: receiver\_public\_key.pem).
* Nút "Tải Khóa" (cho khóa riêng tư): Kích hoạt việc tải khóa riêng tư đã chọn.
* Nút "Tải Khóa" (cho khóa công khai): Kích hoạt việc tải khóa công khai của người nhận đã chọn.

**Kết Nối Với Người Nhận:**

* Ô nhập "Địa Chỉ Máy Nhận": Cho phép nhập địa chỉ IP hoặc hostname của máy chủ người nhận (ví dụ: 127.0.0.1).
* Ô nhập "Cổng": Cho phép nhập số cổng mà máy chủ người nhận đang lắng nghe (ví dụ: 12345 hoặc 5001).
* Nút "Kết Nối": Thực hiện việc thiết lập kết nối TCP và quá trình bắt tay (handshake) cùng trao đổi khóa phiên với người nhận.

**Gửi Hợp Đồng:**

* Ô chọn file "Choose File": Cho phép người dùng chọn tệp hợp đồng (.txt hoặc bất kỳ loại file nào) từ máy tính để gửi đi.
* Nút "Gửi Hợp Đồng": Khi được nhấn, ứng dụng sẽ xử lý file đã chọn (chia phần, mã hóa bằng Triple DES, ký số từng phần bằng RSA/SHA-512) và gửi đến người nhận.

**Nhật Ký Hoạt Động:**

* Một khung hiển thị lớn, nơi tất cả các sự kiện và thông báo quan trọng của ứng dụng được ghi lại theo thời gian thực.
* Hiển thị các bước như tải khóa, kết nối thành công, trạng thái trao đổi khóa, và đặc biệt là Hash (Base64) và Chữ ký (Base64) của từng phần file sau khi gửi (sau khi đã sửa đổi code như hướng dẫn trước đó).

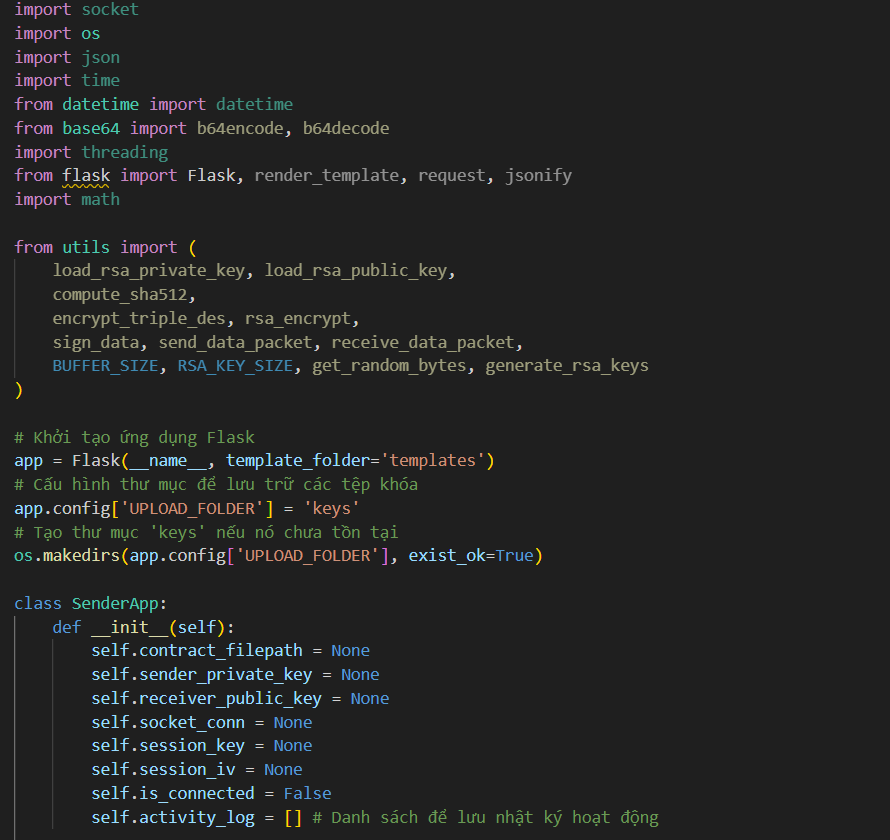
**Các nút chức năng:**

Các nút chức năng được tích hợp trực tiếp vào các khối chức năng tương ứng trên giao diện web, thay vì là các nút riêng biệt như trong mô tả ban đầu của bạn.

* "Tạo Khóa Mới": Nút này thực hiện việc sinh khóa.
* "Tải Khóa": Các nút này thực hiện việc tải các tệp khóa đã chọn.
* "Kết Nối": Nút này khởi tạo kết nối.
* "Gửi Hợp Đồng": Nút này bắt đầu quá trình mã hóa, ký và gửi file**.**

A screenshot of a computer

AI-generated content may be incorrect.

****

**Ý nghĩa:**

* app = Flask(...): Khởi tạo ứng dụng web Flask cho người gửi.
* SenderApp class: Lớp chính quản lý trạng thái của ứng dụng gửi, bao gồm các khóa RSA, kết nối socket, khóa phiên (session key), IV, và nhật ký hoạt động.

A screen shot of a computer program

AI-generated content may be incorrect.

**Ý nghĩa:**

* **connect\_to\_receiver**: Hàm này thiết lập kết nối mạng (TCP) từ ứng dụng gửi đến ứng dụng nhận. Sau khi kết nối thành công, nó sẽ gọi hàm \_perform\_handshake\_and\_key\_exchange để khởi tạo kênh truyền thông bảo mật

A screen shot of a computer code

AI-generated content may be incorrect.

**Ý nghĩa:**

* **\_perform\_handshake\_and\_key\_exchange**: Đây là trái tim của quá trình thiết lập bảo mật. Nó thực hiện bắt tay ban đầu, sau đó tạo ra khóa phiên (session key) và IV ngẫu nhiên. Các khóa này được mã hóa bằng khóa công khai RSA của người nhận và được ký số bằng khóa riêng tư của người gửi, đảm bảo rằng chỉ người nhận mới có thể giải mã khóa phiên và người nhận có thể xác minh rằng khóa này đến từ người gửi hợp lệ.

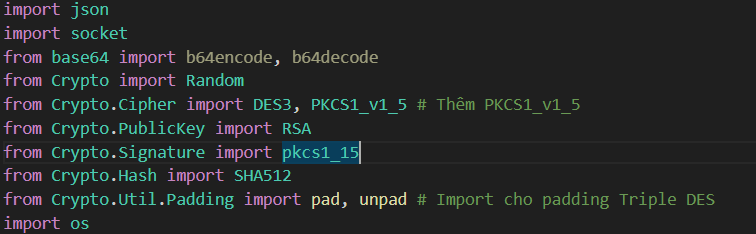
A screen shot of a computer program

AI-generated content may be incorrect.

**Ý nghĩa:**

* **send\_contract\_file**: Hàm này quản lý toàn bộ quá trình gửi file. Nó chia file thành 3 phần, với mỗi phần: tạo IV ngẫu nhiên, mã hóa bằng Triple DES, tính toán hash của (IV || Ciphertext) và ký hash đó bằng khóa riêng tư của người gửi. Cuối cùng, nó gửi từng phần kèm theo IV, ciphertext, hash và chữ ký đến người nhận, đảm bảo tính toàn vẹn và xác thực của từng phần dữ liệu.

## **3.9 Code utils.py**

****

**Các hàm cần thiết**

A computer screen shot of text

AI-generated content may be incorrect.

**Ý nghĩa:** **Quản lý khóa RSA**

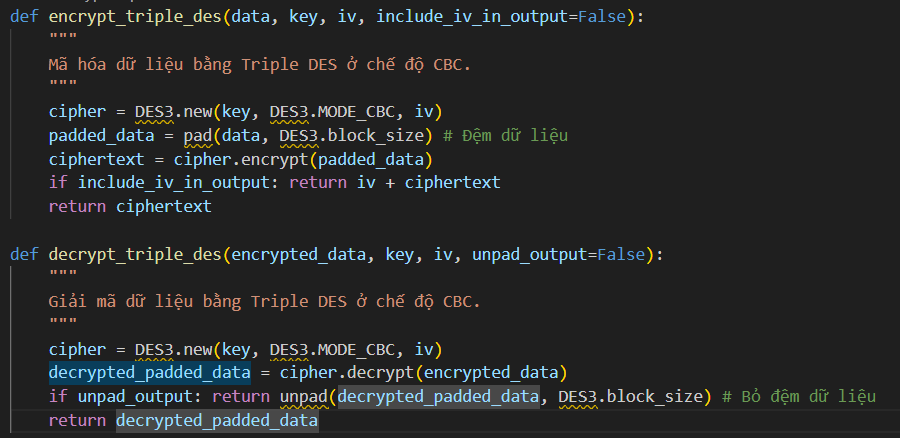
* **generate\_rsa\_keys**: Tạo một cặp khóa RSA (khóa riêng tư và khóa công khai) với kích thước 2048-bit và lưu chúng vào các tệp .pem. Đây là nền tảng cho mã hóa bất đối xứng và chữ ký số.
* **load\_rsa\_private\_key / load\_rsa\_public\_key**: Tải các khóa RSA từ tệp .pem để ứng dụng có thể sử dụng chúng cho các hoạt động mã hóa/giải mã hoặc ký/xác minh.

A screen shot of a computer code

AI-generated content may be incorrect.

**Ý nghĩa:** **Mã hóa và Giải mã RSA**

* **rsa\_encrypt**: Mã hóa dữ liệu (thường là các khóa phiên nhỏ) bằng khóa công khai RSA của người nhận. Chỉ người sở hữu khóa riêng tư tương ứng mới có thể giải mã.
* **rsa\_decrypt**: Giải mã dữ liệu đã được mã hóa bằng RSA bằng khóa riêng tư tương ứng. Các hàm này sử dụng chế độ đệm PKCS#1 v1.5 để tăng cường bảo mật



**Ý nghĩa: Mã hóa và Giải mã Triple DES (3DES)**

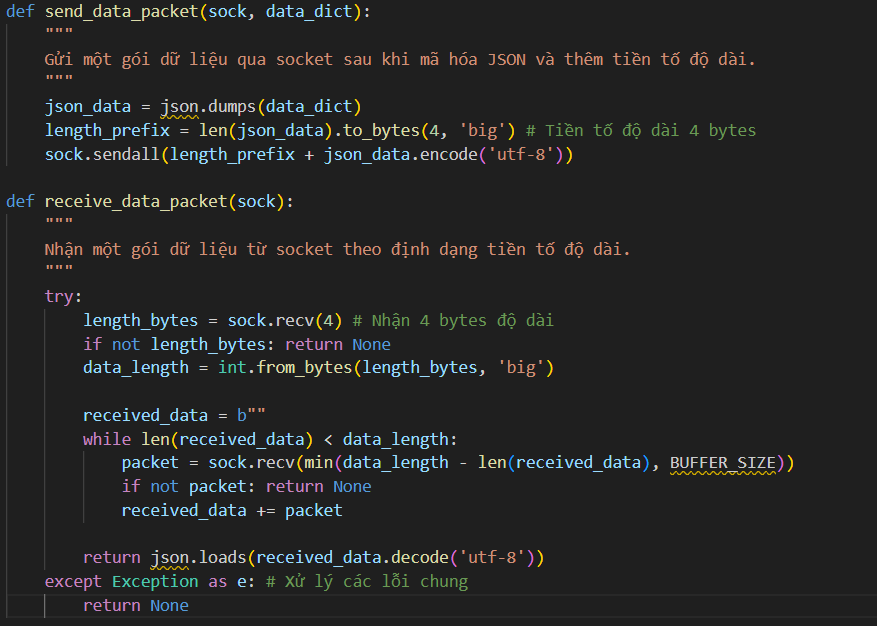
* **encrypt\_triple\_des**: Mã hóa dữ liệu (các phần của file) bằng thuật toán Triple DES ở chế độ CBC. Nó sử dụng một khóa phiên và một IV duy nhất cho mỗi lần mã hóa, đồng thời thêm đệm vào dữ liệu để đảm bảo độ dài phù hợp với kích thước khối của thuật toán.
* **decrypt\_triple\_des**: Giải mã dữ liệu đã được mã hóa bằng Triple DES. Nó sử dụng cùng khóa phiên và IV đã dùng để mã hóa, sau đó loại bỏ phần đệm để khôi phục dữ liệu gốc.

A screen shot of a computer program

AI-generated content may be incorrect.

**Ý nghĩa:** **Băm và Chữ ký số**

* **compute\_sha512**: Tính toán giá trị băm SHA-512 của bất kỳ dữ liệu đầu vào nào. Hash này được dùng để kiểm tra tính toàn vẹn và là đầu vào cho chữ ký số.
* **sign\_data**: Tạo chữ ký số cho một giá trị băm (hash) bằng khóa riêng tư RSA của người ký. Chữ ký này đảm bảo tính xác thực (ai đã ký) và không chối bỏ.
* **verify\_signature**: Xác minh chữ ký số bằng cách sử dụng hash gốc, chữ ký và khóa công khai RSA của người ký. Nếu chữ ký hợp lệ, nó xác nhận rằng dữ liệu không bị thay đổi và đúng là do người sở hữu khóa riêng tư tương ứng đã ký.



**Ý nghĩa:** **Giao tiếp Socket**

* **send\_data\_packet**: Đóng gói dữ liệu Python dictionary thành chuỗi JSON, thêm một tiền tố 4-byte chỉ định độ dài của dữ liệu JSON, sau đó gửi toàn bộ qua socket. Điều này đảm bảo người nhận biết được kích thước gói tin cần nhận.
* **receive\_data\_packet**: Nhận dữ liệu từ socket. Đầu tiên, nó đọc 4 byte đầu tiên để xác định độ dài của gói tin, sau đó đọc đủ dữ liệu theo độ dài đó, và cuối cùng giải mã dữ liệu JSON thành Python dictionary.

## **3.10 Code verify\_tool.py**

A screenshot of a computer

AI-generated content may be incorrect.

**A black screen with white text

AI-generated content may be incorrect.**

**Các dòng này nhập các thư viện cần thiết.**

* os: Để tương tác với hệ điều hành (ví dụ: kiểm tra file).
* base64.b64decode: Để giải mã chuỗi Base64 thành byte.
* load\_rsa\_public\_key, verify\_signature, compute\_sha512: Các hàm mật mã quan trọng từ tệp utils.py

**A black screen with text

AI-generated content may be incorrect.**

Đoạn này yêu cầu người dùng nhập đường dẫn đến khóa công khai của người gửi. Hàm load\_rsa\_public\_key được gọi để tải khóa này. Khóa công khai này là bắt buộc để xác minh chữ ký số

A screen shot of a computer program

AI-generated content may be incorrect.

Người dùng nhập chuỗi hash gốc (dạng Base64) từ log của người gửi. Hàm b64decode chuyển đổi chuỗi Base64 này thành dạng byte gốc, sẵn sàng cho việc xác minh. Khối try-except xử lý lỗi nếu chuỗi Base64 không hợp lệ

**A computer screen with text

AI-generated content may be incorrect.**

Người dùng nhập chuỗi chữ ký số (dạng Base64). Hàm b64decode giải mã chuỗi này thành dạng byte gốc của chữ ký

**A screen shot of a computer code

AI-generated content may be incorrect.**

Hàm verify\_signature được gọi với hash đã giải mã, chữ ký đã giải mã và khóa công khai của người gửi. Nếu hàm trả về True, chữ ký hợp lệ, xác nhận dữ liệu nguyên vẹn và đúng người gửi. Nếu trả về False, chữ ký không hợp lệ, cho thấy dữ liệu có thể đã bị thay đổi hoặc khóa không khớp.

# **CHƯƠNG 4. KẾT LUẬN**

## **4.1. Đánh giá hiệu quả**

Dựa trên kết quả thử nghiệm, chúng ta sẽ đánh giá hiệu quả của hệ thống theo các tiêu chí sau:

**Tính đúng đắn của dữ liệu:** Sau khi giải mã, file contract.txt nhận được phải giống hệt với file contract.txt gốc. Điều này được xác nhận bằng cách so sánh giá trị băm SHA-256 của cả hai file. Nếu các giá trị băm khớp nhau, chứng tỏ quá trình mã hóa, truyền tải và giải mã đã diễn ra chính xác mà không làm mất mát hoặc thay đổi dữ liệu.

**Kiểm tra tính toàn vẹn và xác thực:**

* Hệ thống có khả năng **phát hiện thành công các thay đổi nhỏ** trong dữ liệu hoặc chữ ký số. Trong trường hợp cố ý giả mạo dữ liệu trên đường truyền (ví dụ: thay đổi một byte trong ciphertext hoặc IV của một phần), Người nhận phải từ chối hợp đồng bằng cách gửi NACK.
* Cơ chế ký số từng phần và kiểm tra hash cho phép **xác định chính xác phần nào của hợp đồng bị lỗi** hoặc giả mạo, giúp dễ dàng hơn trong việc xử lý lỗi.

**Hiệu năng:**

* **Triple DES:** Hiệu năng của Triple DES tương đối chậm so với các thuật toán mã hóa đối xứng hiện đại như AES. Với các file kích thước lớn, thời gian mã hóa và giải mã sẽ đáng kể.
* **RSA 2048-bit (Ký số/Xác minh):** Các phép toán ký số và xác minh chữ ký RSA là rất tốn kém về mặt tính toán. Việc ký số riêng cho từng trong 3 phần của hợp đồng sẽ làm tăng đáng kể thời gian xử lý tổng thể, đặc biệt là ở phía Người gửi (ký) và Người nhận (xác minh).
* **SHA-512:** Hàm băm SHA-512 hoạt động rất nhanh và hiệu quả, ít ảnh hưởng đến hiệu năng tổng thể.

Tổng thể, hệ thống đáp ứng tốt các yêu cầu về bảo mật, xác thực và toàn vẹn. Tuy nhiên, hiệu năng có thể là một vấn đề đối với các file hợp đồng có dung lượng rất lớn do đặc tính của Triple DES và số lượng phép toán RSA phải thực hiện.

## **4.2. Phân tích, nhận xét đặc điểm của các thuật toán được sử dụng**

### **4.2.1. Triple DES (3DES)**

**Đặc điểm:** Là một thuật toán mã hóa khối đối xứng, sử dụng ba lần DES để tăng cường bảo mật. Kích thước khóa hiệu quả là 112 hoặc 168 bit (tùy thuộc vào số khóa độc lập).

**Ưu điểm:**

* Đã được kiểm chứng và tin cậy trong nhiều năm.
* Cung cấp mức độ bảo mật đủ mạnh cho nhiều ứng dụng.

**Nhược điểm:**

* **Tốc độ chậm:** Là nhược điểm lớn nhất so với các thuật toán hiện đại như AES. Mỗi khối dữ liệu phải trải qua ba lần mã hóa/giải mã, dẫn đến hiệu suất thấp hơn.
* **Kích thước khối nhỏ (64-bit):** Kích thước khối nhỏ hơn làm cho nó kém hiệu quả hơn trong việc xử lý các luồng dữ liệu lớn so với AES (128-bit).
* **Đang dần bị thay thế:** Đã có những khuyến nghị chuyển sang các thuật toán mạnh mẽ và hiệu quả hơn như AES.

**Phù hợp với bài toán:** Triple DES đáp ứng yêu cầu mã hóa, nhưng sẽ là nút cổ chai về hiệu năng nếu contract.txt có kích thước rất lớn.

### **4.2.2. RSA 2048-bit (PKCS#1 v1.5 + SHA-512)**

**Đặc điểm:** Thuật toán mã hóa bất đối xứng được sử dụng cho cả mã hóa khóa và chữ ký số. Kích thước khóa 2048-bit cung cấp độ an toàn cao. PKCS#1 v1.5 là một tiêu chuẩn đệm được sử dụng để chuẩn bị dữ liệu cho các phép toán RSA. SHA-512 được dùng làm hàm băm trước khi ký số.

**Ưu điểm:**

* **Trao đổi khóa an toàn:** Cho phép trao đổi Session Key (khóa đối xứng) một cách bí mật và an toàn.
* **Ký số mạnh mẽ:** Cung cấp khả năng xác minh nguồn gốc và tính toàn vẹn của dữ liệu một cách không thể chối cãi. Việc sử dụng SHA-512 đảm bảo tính toàn vẹn cao của dữ liệu được ký.
* **Bảo mật cao:** Khóa 2048-bit được coi là rất an toàn trước các tấn công dựa trên việc phân tích thừa số nguyên lớn.

**Nhược điểm:**

* **Tốc độ chậm:** Các phép toán RSA (đặc biệt là ký số và giải mã) cực kỳ chậm so với các thuật toán đối xứng. Do đó, nó không phù hợp để mã hóa trực tiếp toàn bộ dữ liệu file lớn.
* **Yêu cầu tài nguyên:** Thực hiện các phép toán với số nguyên lớn đòi hỏi tài nguyên tính toán đáng kể.

**Phù hợp với bài toán:** RSA là lựa chọn xuất sắc cho trao đổi Session Key và ký số metadata cũng như từng phần của hợp đồng, nơi mà bảo mật và xác thực là tối quan trọng hơn hiệu năng. Việc ký số riêng từng phần làm tăng độ tin cậy nhưng cũng làm tăng thời gian xử lý.

### **4.2.3. SHA-512**

**Đặc điểm:** Hàm băm mật mã tạo ra đầu ra có độ dài cố định 512 bit (64 byte) từ bất kỳ đầu vào nào.

**Ưu điểm:**

* **Tính toàn vẹn cao:** Bất kỳ sự thay đổi nào dù nhỏ trong dữ liệu đầu vào cũng tạo ra một giá trị băm hoàn toàn khác, giúp phát hiện giả mạo hiệu quả.
* **Chống va chạm mạnh:** Khó khăn về mặt tính toán để tìm ra hai đầu vào khác nhau tạo ra cùng một giá trị băm.
* **Tốc độ nhanh:** So với các thuật toán mã hóa, tính toán băm rất nhanh.

**Nhược điểm:**

* **Không mã hóa:** Bản thân SHA-512 không cung cấp bảo mật dữ liệu (tính bí mật). Nó phải được kết hợp với chữ ký số (như RSA) để đảm bảo cả tính toàn vẹn và xác thực.

**Phù hợp với bài toán:** SHA-512 là lựa chọn lý tưởng để tính toán giá trị băm cho việc kiểm tra toàn vẹn và là đầu vào cho quá trình ký số RS

## **4.3. Đề xuất cải tiến và hướng phát triển**

**Cải tiến**:

* Thay thế Triple DES bằng AES-256 để tăng tốc độ mã hóa và nâng cao bảo mật, đặc biệt với các file kích thước lớn.
* Sử dụng RSA với đệm OAEP thay cho PKCS#1 v1.5 để giảm thiểu các lỗ hổng lý thuyết và tăng cường an toàn trong trao đổi khóa.
* Tối ưu hóa xử lý file lớn bằng cách áp dụng truyền tải luồng (streaming) thay vì chia file tĩnh, giảm thời gian xử lý và sử dụng bộ nhớ hiệu quả hơn.

**Hướng phát triển**

**Tích hợp thực tế:**

* Phát triển nền tảng ký hợp đồng điện tử với giao diện web/di động, hỗ trợ quản lý hợp đồng và phân quyền người dùng (admin, người ký).
* Tích hợp gửi hợp đồng qua email (SMTP/TLS) hoặc ứng dụng nhắn tin an toàn như Telegram, Signal.

**Tăng cường bảo mật:**

* Sử dụng HSM hoặc Smartcard để lưu trữ khóa riêng an toàn.
* Áp dụng chữ ký số theo chuẩn quốc gia (VD: Bộ TT&TT) hoặc liên kết với VNPT-CA, Viettel-CA.
* Triển khai kiểm tra dữ liệu đầu vào và chống tấn công MITM, DDoS.

**Tự động hóa và tối ưu:**

* Áp dụng kiểm thử tự động bằng Pytest và kiểm thử API với Postman.
* Sử dụng JMeter để đo tải và pipeline CI/CD cho kiểm thử hồi quy.
* Tối ưu xử lý file lớn bằng truyền tải luồng (streaming).

**Hướng dài hạn:**

* Hỗ trợ đa định dạng file (PDF, DOCX) và tích hợp blockchain để lưu trữ lịch sử hợp đồng minh bạch.
* Áp dụng TLS 1.3 để tăng cường bảo mật truyền tải.

## **4.4 Kết Quả Đạt Được**

**Về mặt kiến thức**

* Nắm vững kiến thức về mã hóa, chữ ký số và các thuật toán mật mã cơ bản như RSA, SHA-256, Diffie-Hellman.
* Hiểu được vai trò và nguyên lý hoạt động của chữ ký số trong việc đảm bảo tính toàn vẹn, tính xác thực và tính không thể chối bỏ của dữ liệu.
* Vận dụng lý thuyết về an toàn và bảo mật thông tin để xây dựng hệ thống gửi hợp đồng tích hợp chữ ký số riêng đảm bảo độ tin cậy cao.
* Hiểu rõ các bước trong quy trình ký số: tạo khóa, ký tài liệu, xác minh chữ ký, và trao đổi khóa an toàn.

**Về mặt kỹ năng**

* Phát triển kỹ năng lập trình ứng dụng sử dụng công nghệ mã hóa khóa công khai để bảo vệ dữ liệu trong quá trình truyền tải.
* Có khả năng thiết kế luồng xử lý bảo mật (ký số, kiểm tra, xác minh) trong các ứng dụng thực tế.
* Tăng cường kỹ năng kiểm thử và phân tích dữ liệu đầu ra để đảm bảo tính chính xác, toàn vẹn của thông tin.
* Nâng cao khả năng làm việc nhóm, phân chia công việc hiệu quả và phối hợp giữa các thành viên trong nhóm khi triển khai hệ thống gửi và xác minh hợp đồng điện tử.

**Hạn chế**

* Chưa tích hợp đầy đủ các lớp bảo mật nâng cao như xác thực hai yếu tố (2FA), lưu trữ khóa riêng an toàn bằng phần cứng (HSM).
* Việc triển khai hệ thống vẫn ở mức độ mô hình, chưa áp dụng trong môi trường thực tế doanh nghiệp để đánh giá hiệu suất thực tế.
* Một số bước trong quy trình xác thực còn được thực hiện thủ công, chưa được tự động hóa hoàn toàn.
* Khả năng xử lý các tình huống ngoại lệ hoặc tấn công giả mạo chữ ký số chưa được khai thác sâu.

# **LỜI CẢM ƠN**

Trong suốt quá trình thực hiện đề tài “Gửi hợp đồng với chữ ký số riêng”, nhóm chúng tôi đã nhận được sự hỗ trợ quý báu từ thầy cô và bạn bè, những người đã tạo điều kiện và động viên chúng tôi trong suốt hành trình nghiên cứu và hoàn thành đề tài.

Đầu tiên, nhóm xin chân thành cảm ơn Trần Đức Thắng giảng viên môn An Toàn Bảo Mật Thông Tin, người đã tận tình hướng dẫn, cung cấp kiến thức và chỉ bảo chúng tôi trong suốt quá trình nghiên cứu và thực hiện đề tài. Những lời khuyên và sự chỉ bảo của thầy cô không chỉ giúp chúng tôi giải quyết các vấn đề kỹ thuật mà còn là động lực lớn giúp nhóm vượt qua những khó khăn trong quá trình phát triển phần mềm.

Chúng tôi cũng xin cảm ơn các bạn trong nhóm, những người đã cùng nhau làm việc, chia sẻ ý tưởng và nỗ lực hết mình để hoàn thành đề tài. Mỗi thành viên trong nhóm đều đóng góp công sức, trí tuệ và thời gian để cùng nhau đạt được mục tiêu chung. Sự hợp tác, tinh thần làm việc nhóm và sự sáng tạo của các bạn là yếu tố quan trọng giúp chúng tôi hoàn thành công trình này.

Cuối cùng, nhóm xin cảm ơn tất cả những ai đã tạo điều kiện và giúp đỡ chúng tôi trong suốt thời gian thực hiện đề tài này. Chúng tôi hy vọng rằng kết quả của đề tài sẽ không chỉ đáp ứng được yêu cầu và mong đợi của thầy cô mà còn đóng góp một phần nhỏ vào việc cải thiện công tác quản lý sinh viên tại các trường học.

Chúng tôi xin chân thành cảm ơn và mong tiếp tục nhận được sự hỗ trợ từ thầy cô và các bạn trong những dự án học thuật và nghiên cứu sau này.

# **TÀI LIỆU THAM KHẢO**

[1] T. Đ. Hoàng, *An toàn và bảo mật thông tin*, TP.HCM: NXB Đại học Quốc gia TP.HCM, 2019.

[2] W. Stallings, *Cryptography and Network Security: Principles and Practice*, 7th ed., Pearson, 2017.

[3] B. Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, 2nd ed., Wiley, 2015.

[4] Giáo trình an toàn bảo mật thông tin của khoa CNTT

[5] PyCryptodome Documentation: <https://pycryptodome.readthedocs.io/>

[6] Python Socket Programming – <https://docs.python.org/3/library/socket.html>

[7] Tkinter GUI Programming – <https://docs.python.org/3/library/tkinter.html>

[8] William Stallings, Cryptography and Network Security: Principles and Practice, 7th Edition, Pearson Education, 2017