BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC ĐẠI NAM**

**KHOA CÔNG NGHỆ THÔNG TIN**

---


BÁO CÁO BÀI TẬP LỚN

NHẬP MÔN AN TOÀN BẢO MẬT THÔNG TIN

# GỬI TÀI LIỆU EMAIL CÓ GIỚI HẠN THỜI GIAN

**Sinh viên thực hiện** : **Đỗ Khánh Hùng**
**Nguyễn Quốc Huy**
**Bùi Trọng Tài**

**Ngành** : **Công nghệ thông tin**

**Giảng viên hướng dẫn** : **ThS. Lê Thị Thùy Trang**

## Lời cảm ơn

Chúng tôi xin gửi lời cảm ơn chân thành đến ThS. Lê Thị Thùy Trang, giảng viên môn học Nhập môn An toàn và Bảo mật Thông tin, người đã tận tình giảng dạy, hướng dẫn và truyền đạt những kiến thức nền tảng quan trọng về lĩnh vực bảo mật thông tin trong suốt quá trình học tập. Sự tận tâm và những bài giảng đầy cảm hứng của cô đã giúp chúng tôi không chỉ nắm vững lý thuyết mà còn biết cách áp dụng vào thực tiễn, từ đó tạo nền tảng vững chắc để chúng tôi thực hiện bài tập lớn với đề tài "Gửi tài liệu email có giới hạn thời gian". Những hướng dẫn chi tiết và sự hỗ trợ nhiệt tình của cô đã giúp chúng tôi vượt qua những thách thức trong quá trình phát triển ứng dụng, từ việc thiết kế giao diện đến triển khai các thuật toán bảo mật phức tạp.

Chúng tôi cũng xin bày tỏ lòng biết ơn sâu sắc đến các thầy cô trong khoa Công nghệ Thông tin, những người đã xây dựng một chương trình học tập khoa học, gắn liền với thực tiễn, giúp sinh viên tiếp cận các vấn đề thực tế và cấp thiết trong lĩnh vực bảo mật thông tin. Nhờ chương trình học này, chúng tôi đã có cơ hội tiếp cận với các kỹ thuật bảo mật tiên tiến như mã hóa đối xứng AES-CBC, mã hóa bất đối xứng RSA 2048-bit (PKCS1 v1.5), ký số và kiểm tra tính toàn vẹn bằng SHA-512, cũng như các phương pháp truyền tải file an toàn qua mạng. Bài tập lớn này không chỉ là một nhiệm vụ học tập mà còn là cơ hội để chúng tôi thực hành và củng cố kiến thức, đồng thời phát triển kỹ năng làm việc nhóm và giải quyết vấn đề thực tế.

Đề tài "Gửi tài liệu email có giới hạn thời gian" yêu cầu xây dựng một ứng dụng cho phép công ty A gửi tài liệu nhạy cảm (email.txt) đến nhân viên B, với giới hạn truy cập trong vòng 24 giờ kể từ khi gửi. Ứng dụng sử dụng các kỹ thuật mã hóa, xác thực và kiểm tra toàn vẹn để đảm bảo an toàn dữ liệu, đồng thời hỗ trợ hai chế độ truyền tải: qua socket (Local) và qua Gmail hoặc Google Drive (Cloud). Trong quá trình thực hiện, chúng tôi đã đối mặt với nhiều thách thức như tích hợp Gmail API, xử lý lỗi định dạng khóa, và đảm bảo tính chính xác của cơ chế giới hạn thời gian. Tuy nhiên, những khó khăn này đã giúp chúng tôi học hỏi thêm nhiều kinh nghiệm quý báu về lập trình, bảo mật và quản lý dự án.

Chúng tôi hy vọng rằng bài tập lớn này không chỉ thể hiện nỗ lực của nhóm mà còn phản ánh sự hiểu biết sâu sắc về các khái niệm bảo mật thông tin. Chúng tôi mong nhận được những ý kiến đóng góp quý báu từ cô giáo và các thầy cô trong khoa để hoàn thiện bài tập, đồng thời nâng cao kiến thức và kỹ năng của mình trong lĩnh vực bảo mật thông tin. Những góp ý này sẽ là động lực để chúng tôi tiếp tục học tập, nghiên cứu và phát triển các giải pháp bảo mật hiệu quả hơn trong tương lai.

Xin chân thành cảm ơn!

# Mục lục


**1 Giới thiệu** **1**

1.1 Đặt vấn đề . . . . . 1

1.2 Phân tích bài toán . . . . . 2

1.3 Mục tiêu của đề tài . . . . . 3

1.4 Phạm vi thực hiện . . . . . 5

1.5 Cấu trúc của báo cáo . . . . . 6

**2 Phân tích yêu cầu và thiết kế ứng dụng** **8**

2.1 Mô tả thuật toán . . . . . 8

2.2 Phân tích mã nguồn . . . . . 10

2.3 Thử nghiệm . . . . . 14

**3 Kết luận và hướng phát triển** **17**

3.1 Triển khai bảo mật . . . . . 17

3.1.1 Hướng giải quyết . . . . . 17

3.1.2 Luồng xử lý hệ thống . . . . . 17

3.1.3 Triển khai giải pháp bảo mật . . . . . 18

3.1.4 Đánh giá và kiểm tra . . . . . 19

3.1.5 Cấu trúc thư mục dự án . . . . . 20

3.1.6 Các bước triển khai . . . . . 21

3.2 Phân tích hiệu quả . . . . . 26

3.2.1 Phân tích và nhận xét đặc điểm của các thuật toán sử dụng . . . . . 27

3.2.2 Đề xuất cải tiến . . . . . 28

# Danh sách hình vẽ


3.1 Giao diện GUI Sender . . . . . 21
3.2 Giao diện GUI Receiver . . . . . 21
3.3 Sender nhập email liên quan . . . . . 22
3.4 Receiver nhận và xác minh mã OTP . . . . . 23
3.5 Sender tạo khóa private và public . . . . . 23
3.6 Receiver nhận khóa . . . . . 24
3.7 Receiver nhận file . . . . . 24
3.8 Sender chọn và gửi file . . . . . 25
3.9 Chọn file để giải mã . . . . . 25
3.10 Kết quả đạt được . . . . . 26

# Chương 1

# Giới thiệu

## 1.1 Đặt vấn đề

Trong bối cảnh công nghệ thông tin phát triển nhanh chóng, nhu cầu truyền tải dữ liệu nhạy cảm qua mạng, đặc biệt là qua email, ngày càng trở nên phổ biến trong các tổ chức, doanh nghiệp và cá nhân. Tuy nhiên, việc gửi các tài liệu chứa thông tin quan trọng như hợp đồng, báo cáo tài chính, hoặc dữ liệu cá nhân đặt ra những thách thức lớn về bảo mật, bao gồm nguy cơ bị đánh cắp, giả mạo hoặc truy cập trái phép. Đặc biệt, trong môi trường mạng mở như Internet, các cuộc tấn công mạng ngày càng tinh vi, đòi hỏi các giải pháp bảo mật mạnh mẽ để đảm bảo an toàn dữ liệu trong quá trình truyền tải.

Đề tài "Gửi tài liệu email có giới hạn thời gian" được xây dựng nhằm giải quyết bài toán thực tế của Công ty A, nơi cần gửi một tài liệu nhạy cảm (email.txt) đến nhân viên B, với yêu cầu chỉ cho phép truy cập trong vòng 24 giờ kể từ thời điểm gửi. Để đáp ứng yêu cầu này, ứng dụng cần tích hợp các cơ chế bảo mật tiên tiến, bao gồm mã hóa dữ liệu, xác thực người dùng, kiểm tra tính toàn vẹn và giới hạn thời gian truy cập. Cụ thể, bài toán yêu cầu sử dụng thuật toán AES-CBC để mã hóa dữ liệu, RSA 2048-bit (PKCS1 v1.5) để trao đổi khóa và ký số, cùng với SHA-512 để kiểm tra tính toàn vẹn, nhằm đảm bảo rằng chỉ người nhận được ủy quyền mới có thể truy cập tài liệu và dữ liệu không bị thay đổi trong quá trình truyền tải.

Ứng dụng được thiết kế để hỗ trợ hai chế độ truyền tải: Local (qua socket TCP) và Cloud (qua Gmail hoặc Google Drive), đáp ứng các kịch bản sử dụng khác nhau, từ truyền tải nội bộ đến chia sẻ qua môi trường đám mây. Ngoài ra, cơ chế xác thực OTP qua Gmail được tích hợp để đảm bảo danh tính người nhận, tăng cường lớp bảo vệ bổ sung trước khi gửi file hoặc khóa. Việc giới hạn thời gian truy cập trong 24 giờ giúp giảm thiểu nguy cơ tài liệu bị lạm dụng sau khi hết hạn, phù hợp với các yêu cầu bảo mật nghiêm ngặt của doanh nghiệp.

Đề tài không chỉ mang ý nghĩa thực tiễn trong việc bảo vệ thông tin nhạy cảm mà còn là cơ

hội để chúng tôi áp dụng các kiến thức đã học trong môn Nhập môn An toàn và Bảo mật Thông tin, bao gồm các khái niệm về mã hóa đối xứng, mã hóa bất đối xứng, ký số, và kiểm tra toàn vẹn dữ liệu. Thông qua việc phát triển ứng dụng, chúng tôi mong muốn hiểu sâu hơn về các kỹ thuật bảo mật hiện đại, đồng thời rèn luyện kỹ năng lập trình, phân tích hệ thống và làm việc nhóm để giải quyết các vấn đề thực tế trong lĩnh vực an toàn thông tin.

## 1.2 Phân tích bài toán

Đề tài "Gửi tài liệu email có giới hạn thời gian" đã đặt ra bài toán cụ thể: Công ty A cần gửi một tài liệu nhạy cảm (email.txt) đến nhân viên B, với yêu cầu tài liệu chỉ có thể được truy cập trong vòng 24 giờ kể từ khi gửi. Để đáp ứng yêu cầu này, ứng dụng cần đảm bảo các yếu tố bảo mật, xác thực, toàn vẹn dữ liệu và giới hạn thời gian truy cập. Dưới đây là phân tích chi tiết các yêu cầu của bài toán:

Bảo mật dữ liệu: Tài liệu cần được mã hóa để ngăn chặn truy cập trái phép từ các bên thứ ba trong quá trình truyền tải qua mạng (qua socket hoặc email/đám mây). Vì vậy chúng tôi sử dụng thuật toán mã hóa đối xứng AES-CBC với khóa 256-bit để mã hóa nội dung file, đảm bảo hiệu suất cao khi xử lý các file có kích thước lớn. Khóa phiên (session key) được mã hóa bằng RSA 2048-bit (PKCS1 v1.5) để trao đổi an toàn giữa người gửi và người nhận. Bởi AES-CBC cung cấp mã hóa mạnh mẽ và nhanh chóng, trong khi RSA đảm bảo an toàn cho việc trao đổi khóa phiên, ngăn chặn rò rỉ khóa trong quá trình truyền tải.

Xác thực người dùng: Chỉ người nhận được ủy quyền (nhân viên B) mới có thể giải mã và truy cập tài liệu. Vậy cần tích hợp cơ chế xác thực OTP qua Gmail để đảm bảo danh tính của người nhận trước khi gửi file hoặc khóa. Người gửi sử dụng khóa công khai của người nhận để mã hóa khóa phiên, và người nhận sử dụng khóa riêng của mình để giải mã. Mà OTP cung cấp một lớp xác thực bổ sung, trong khi RSA 2048-bit đảm bảo chỉ người nhận có khóa riêng tương ứng mới có thể truy cập khóa phiên và giải mã file.

Kiểm tra tính toàn vẹn: Vì Dữ liệu phải được bảo vệ để không bị thay đổi hoặc giả mạo trong quá trình truyền tải. Nên sử dụng hàm băm SHA-512 để tạo hash từ IV, ciphertext và thời hạn (expiration), cho phép người nhận kiểm tra tính toàn vẹn của dữ liệu. Ngoài ra, metadata (tên file và thời hạn) được ký số bằng RSA/SHA-512 để đảm bảo nguồn gốc dữ liệu. Vì SHA-512 cung cấp khả năng chống va chạm mạnh, giúp phát hiện bất kỳ thay đổi nào trong dữ liệu. Ký số RSA đảm bảo tính xác thực của metadata, ngăn chặn giả mạo.

Giới hạn thời gian: Vì yêu cầu Tài liệu chỉ được truy cập trong vòng 24 giờ kể từ khi gửi, sau đó sẽ bị từ chối giải mã. Nên giải pháp gói tin (packet) chứa trường thời hạn (expiration) định dạng ISO 8601 là tối ưu nhất. Người nhận kiểm tra thời gian hiện tại so với expiration trước khi giải mã. Nếu thời gian hiện tại vượt quá expiration, quá trình giải mã bị từ chối và gửi thông báo

NACK. Cơ chế này đảm bảo tài liệu nhạy cảm không thể bị truy cập sau thời hạn quy định, phù hợp với yêu cầu bảo mật của công ty A.

Phương thức truyền tải: Cần hỗ trợ truyền tải file an toàn qua mạng với hai chế độ: Local (qua socket TCP) và Cloud (qua Gmail hoặc Google Drive). Local: Sử dụng socket TCP với IP và port cố định (mặc định 127.0.0.1:5001) để truyền khóa công khai, khóa phiên mã hóa và gói tin. Cloud: Sử dụng Gmail API để gửi file mã hóa (packet) và khóa phiên, hoặc Google Drive API để tải lên các file này, kèm theo xác thực OTP. Lý do lựa chọn: Chế độ Local phù hợp với truyền tải nội bộ nhanh chóng, trong khi chế độ Cloud tận dụng cơ sở hạ tầng email và đám mây phổ biến, phù hợp với môi trường phân tán.

Luồng xử lý: Dưới đây là luồng xử lý được thực hiện:

Bắt tay (Handshake): Người gửi gửi “Hello!” và người nhận trả lời “Ready!” để thiết lập kết nối.

Xác thực và trao đổi khóa: Gửi OTP qua Gmail, ký số metadata bằng RSA/SHA-512, và mã hóa khóa phiên bằng RSA. Truyền dữ liệu và kiểm tra toàn vẹn: Tạo gói tin chứa IV, ciphertext, hash, chữ ký, expiration và tên file. Người nhận kiểm tra hash, chữ ký và thời hạn trước khi giải mã.

Phản hồi: Gửi ACK nếu thành công, hoặc NACK nếu có lỗi (hash không khớp, chữ ký không hợp lệ, hoặc hết hạn). Bài toán yêu cầu một hệ thống tích hợp nhiều kỹ thuật bảo mật (mã hóa, ký số, xác thực, kiểm tra toàn vẹn) và hỗ trợ nhiều phương thức truyền tải, đồng thời đảm bảo tính đơn giản và hiệu quả cho người dùng cuối. Ứng dụng được xây dựng với giao diện đồ họa (PyQt5) để dễ sử dụng, đồng thời tận dụng các thư viện bảo mật như PyCrypto để triển khai các thuật toán cần thiết.

## 1.3 Mục tiêu của đề tài

Đề tài của chúng tôi hướng đến xây dựng một ứng dụng truyền file an toàn, đảm bảo bảo mật, xác thực, toàn vẹn dữ liệu và giới hạn truy cập trong 24 giờ, sử dụng AES-CBC, RSA 2048-bit, SHA-512, hỗ trợ truyền qua socket hoặc Gmail/Google Drive. Các mục tiêu chính của đề tài bao gồm:

Đảm bảo bảo mật dữ liệu: Ứng dụng được thiết kế để bảo vệ nội dung tài liệu nhạy cảm (email.txt) trong suốt quá trình truyền tải qua mạng, ngăn chặn mọi truy cập trái phép từ các bên thứ ba. Để đạt được điều này, thuật toán mã hóa đối xứng AES-CBC với khóa 256-bit được sử dụng để mã hóa nội dung file, đảm bảo hiệu suất cao khi xử lý các file có kích thước lớn, từ vài KB đến hàng chục MB. Khóa phiên (session key) được mã hóa bằng thuật toán RSA 2048-bit (PKCS1 v1.5) sử dụng khóa công khai của người nhận, đảm bảo

rằng chỉ người nhận sở hữu khóa riêng tương ứng mới có thể giải mã khóa phiên và truy cập nội dung file. Quá trình mã hóa được thực hiện cẩn thận để ngăn chặn rò rỉ thông tin ngay cả khi dữ liệu bị chặn giữa đường, phù hợp với cả hai chế độ truyền tải: Local (qua socket TCP với IP và port cố định, mặc định 127.0.0.1:5001) và Cloud (qua Gmail hoặc Google Drive). Việc tích hợp các thuật toán mã hóa mạnh mẽ này không chỉ đáp ứng yêu cầu bảo mật của đề tài mà còn phù hợp với các tiêu chuẩn bảo mật hiện đại, đảm bảo dữ liệu nhạy cảm của Công ty A được bảo vệ an toàn trong mọi tình huống truyền tải.

Xác thực người dùng: Để đảm bảo chỉ người nhận được ủy quyền (nhân viên B) mới có thể truy cập tài liệu, ứng dụng triển khai một cơ chế xác thực hai lớp mạnh mẽ. Đầu tiên, mã OTP (One-Time Password) được tạo ngẫu nhiên và gửi qua Gmail từ người gửi đến người nhận, yêu cầu người nhận nhập mã OTP chính xác trước khi tiến hành nhận file hoặc khóa. Lớp xác thực này giúp xác minh danh tính người nhận một cách hiệu quả, đặc biệt trong chế độ Cloud. Thứ hai, ứng dụng sử dụng cặp khóa RSA 2048-bit, trong đó khóa công khai của người nhận được sử dụng để mã hóa khóa phiên, và chỉ người nhận có khóa riêng tương ứng mới có thể giải mã. Điều này đảm bảo rằng ngay cả khi file mã hóa bị chặn, chỉ người nhận hợp lệ mới có thể truy cập nội dung. Việc kết hợp OTP qua Gmail và RSA tạo ra một quy trình xác thực chặt chẽ, ngăn chặn các nỗ lực truy cập trái phép và đảm bảo rằng tài liệu chỉ đến tay đúng người được Công ty A ủy quyền. Cơ chế này cũng giúp tăng cường độ tin cậy của hệ thống trong các kịch bản thực tế, nơi việc xác minh danh tính người dùng là yếu tố then chốt.

Kiểm tra tính toàn vẹn: Tính toàn vẹn của dữ liệu là một yêu cầu cốt lõi để đảm bảo tài liệu không bị thay đổi hoặc giả mạo trong quá trình truyền tải. Ứng dụng sử dụng hàm băm SHA-512 để tạo hash từ IV (Initialization Vector), ciphertext (dữ liệu đã mã hóa) và thời hạn (expiration), cho phép người nhận kiểm tra xem dữ liệu có bị can thiệp hay không bằng cách so sánh hash nhận được với hash tính lại. Ngoài ra, metadata bao gồm tên file và thời hạn được ký số bằng RSA/SHA-512 sử dụng khóa riêng của người gửi. Người nhận sử dụng khóa công khai của người gửi để xác minh chữ ký, đảm bảo rằng dữ liệu đến từ nguồn đáng tin cậy và không bị giả mạo. Cơ chế này cung cấp hai lớp bảo vệ: hash SHA-512 đảm bảo tính toàn vẹn của nội dung file, trong khi chữ ký RSA xác nhận tính xác thực của metadata. Nếu bất kỳ thay đổi nào xảy ra trong quá trình truyền tải, quá trình kiểm tra sẽ phát hiện và từ chối gói tin, gửi thông báo NACK về người gửi. Sự kết hợp này không chỉ đáp ứng yêu cầu của đề tài mà còn cung cấp một cơ chế kiểm tra toàn vẹn mạnh mẽ, phù hợp với các ứng dụng bảo mật cao trong thực tế.

Giới hạn thời gian truy cập: Đề tài yêu cầu tài liệu chỉ được truy cập trong vòng 24 giờ kể từ khi gửi, nhằm bảo vệ thông tin nhạy cảm khỏi bị lạm dụng sau thời hạn quy định. Để thực hiện điều này, ứng dụng tích hợp trường thời hạn (expiration) trong gói tin, sử dụng định dạng ISO 8601 (ví dụ: "2025-04-23T09:00:00Z"). Thời hạn này được thiết lập là 24

giờ kể từ thời điểm gửi và được gửi cùng gói tin chứa IV, ciphertext, hash, chữ ký và tên file. Phía người nhận, trước khi giải mã, ứng dụng so sánh thời gian hiện tại (theo UTC) với thời hạn expiration. Nếu thời gian hiện tại vượt quá expiration, quá trình giải mã bị từ chối và một thông báo NACK với lý do "Expired" được gửi về người gửi. Cơ chế này được triển khai trong cả hai chế độ truyền tải, đảm bảo rằng tài liệu nhạy cảm không thể được truy cập sau 24 giờ, ngay cả khi người nhận vẫn giữ file mã hóa. Điều này đặc biệt hữu ích trong các kịch bản yêu cầu bảo mật cao, như chia sẻ tài liệu nội bộ hoặc thông tin tài chính tạm thời.

Hỗ trợ phương thức truyền tải: Ứng dụng được thiết kế để linh hoạt đáp ứng các kịch bản sử dụng khác nhau bằng cách hỗ trợ hai chế độ truyền tải: Local và Cloud. Trong chế độ Local, file mã hóa, khóa phiên và khóa công khai được truyền qua socket TCP với IP và port cố định (mặc định 127.0.0.1:5001), phù hợp với truyền tải nội bộ nhanh chóng trong mạng cục bộ. Trong chế độ Cloud, ứng dụng sử dụng Gmail API để gửi gói tin mã hóa (packet.json) và khóa phiên (sessionkey.bin) qua email, hoặc Google Drive API để tải lên các file này, phù hợp với các kịch bản phân tán nơi người gửi và người nhận ở xa nhau. Giao diện đồ họa được xây dựng bằng PyQt5, cung cấp các trường nhập email, chọn file, chọn khóa, và chọn chế độ truyền tải, giúp người dùng thao tác dễ dàng. Ứng dụng cũng tích hợp các nút chức năng như gửi OTP, tạo cặp khóa RSA, và gửi file qua Gmail, đảm bảo tính tiện lợi và trực quan. Việc hỗ trợ đa phương thức cho phép ứng dụng thích nghi với nhiều môi trường, từ mạng nội bộ đến hệ thống đám mây, đồng thời duy trì các yêu cầu bảo mật và xác thực nghiêm ngặt của đề tài.

## 1.4 Phạm vi thực hiện

Hệ thống bao gồm hai ứng dụng chính: SenderApp (sender.py) và ReceiverApp (receiver.py), cùng với thư viện bảo mật (secure_transfer.py) và các chức năng hỗ trợ OTP (email_otp.py). Về phía người gửi, hệ thống cho phép: tạo cặp khóa RSA, chọn file để gửi, mã hóa file bằng AES-CBC, mã hóa khóa phiên bằng RSA, ký số metadata bằng RSA-SHA512, và truyền file qua socket (Local) hoặc Google Drive/Gmail (Cloud). Người gửi cũng có thể gửi OTP, khóa công khai, hoặc khóa riêng qua Gmail để xác thực và chia sẻ khóa. Về phía người nhận, hệ thống hỗ trợ: xác thực OTP, nhận file qua socket hoặc Google Drive/Gmail, xác minh tính toàn vẹn của file thông qua hash SHA-512 và chữ ký số, giải mã khóa phiên bằng RSA, và giải mã file bằng AES-CBC. Các chức năng này được tích hợp trong giao diện GUI, với các nút bấm và trường nhập liệu để người dùng tương tác dễ dàng. Hệ thống cũng hỗ trợ lưu trữ file mã hóa và file giải mã vào các thư mục như received_files và decrypted_files.

Hệ thống được xây dựng dựa trên ngôn ngữ lập trình Python và sử dụng một số thư viện

quan trọng để triển khai các chức năng bảo mật và giao tiếp. Thư viện pycryptodome được sử dụng để thực hiện các thuật toán mã hóa (RSA, AES-CBC) và băm (SHA-512), đảm bảo tính an toàn của dữ liệu. Giao diện người dùng được phát triển bằng PyQt5, cung cấp trải nghiệm trực quan với các trường nhập liệu, nút bấm, và khu vực hiển thị trạng thái. Trong chế độ Local, giao tiếp mạng được thực hiện thông qua thư viện socket của Python, sử dụng giao thức TCP để truyền dữ liệu. Trong chế độ Cloud, hệ thống tích hợp Google Drive API (thông qua pydrive) để tải lên/tải xuống file, và Gmail API (thông qua email_otp.py) để gửi/nhận OTP, khóa, và file mã hóa. Các file kiểm thử sử dụng các kỹ thuật kiểm thử đơn vị (unit testing) để kiểm tra tính đúng đắn của các hàm mã hóa, giải mã, quản lý khóa, và xử lý OTP.

Hệ thống hướng đến việc cung cấp một giải pháp truyền tải file an toàn cho các tình huống cần bảo mật cao, chẳng hạn như chia sẻ tài liệu nhạy cảm giữa các cá nhân hoặc tổ chức nhỏ. Trong chế độ Local, hệ thống phù hợp cho các mạng nội bộ (LAN) hoặc môi trường có kết nối trực tiếp giữa hai máy tính. Trong chế độ Cloud, hệ thống có thể được sử dụng trong các kịch bản mà người gửi và người nhận ở các vị trí địa lý khác nhau, tận dụng Google Drive hoặc Gmail như các kênh truyền tải trung gian. Đề tài cũng có giá trị học thuật, cung cấp một ví dụ thực tiễn về cách áp dụng các thuật toán mã hóa (RSA, AES, SHA-512) và xác thực (OTP) trong một ứng dụng thực tế. Các bài kiểm thử đi kèm giúp đảm bảo rằng hệ thống hoạt động ổn định và có thể được sử dụng làm nền tảng để phát triển thêm các ứng dụng bảo mật.

## 1.5 Cấu trúc của báo cáo

Báo cáo gồm các phần như sau:

- Chương 1: Giới thiệu

  Chương này cung cấp cái nhìn tổng quan về đề tài, bao gồm lời mở đầu và bối cảnh thực hiện. Là lời cảm ơn của nhóm 8 đến cô Trang, người đã đồng hành cùng chúng tôi, đồng thời nhấn mạnh ý nghĩa của việc áp dụng kiến thức bảo mật vào thực tiễn thông qua việc xây dựng ứng dụng truyền file an toàn. Phần bối cảnh đề tài phân tích nhu cầu thực tế về truyền tải dữ liệu nhạy cảm trong môi trường mạng, nêu rõ các thách thức về bảo mật, xác thực và toàn vẹn dữ liệu, cũng như tầm quan trọng của cơ chế giới hạn thời gian trong việc bảo vệ thông tin nhạy cảm.

- Chương 2: Phân tích yêu cầu và thiết kế ứng dụng

  Ở đây chúng tôi tập trung vào việc phân tích bài toán và mô tả chi tiết quá trình thiết kế, triển khai ứng dụng. Phần phân tích yêu cầu làm rõ các mục tiêu chính, bao gồm mã hóa dữ liệu bằng AES-CBC và RSA 2048-bit, xác thực người dùng qua OTP và RSA, kiểm tra

toàn vẹn bằng SHA-512, giới hạn thời gian truy cập, và hỗ trợ hai chế độ truyền tải(Local qua socket TCP và Cloud qua Gmail/Google Drive). Phần này cũng trình bày các thuật toán được sử dụng (bắt tay, xác thực, truyền dữ liệu), phân tích mã nguồn của các file sender.py, receiver.py, secure_transfer.py, và email_otp.py, cùng với kết quả thử nghiệm trên các file mẫu (txt, mp3, mp4) và đánh giá hiệu quả về tính chính xác, toàn vẹn và bảo mật.

- Chương 3: Kết luận và hướng phát triển

  Chương cuối tổng kết những kết quả mà chúng tôi đã đạt được, bao gồm việc đảm bảo dữ liệu giải mã giống dữ liệu gốc, tính toàn vẹn được duy trì qua kiểm tra hash và chữ ký, cũng như hiệu quả của cơ chế giới hạn thời gian. Phần này cũng phân tích ưu, nhược điểm của các thuật toán (AES-CBC, RSA, SHA-512) và đề xuất cải tiến như sử dụng AES-GCM, thêm cơ chế thu hồi file, hoặc cải thiện giao diện người dùng. Cuối cùng, báo cáo đưa ra các hướng phát triển tương lai, như hỗ trợ thêm nền tảng đám mây (Dropbox, OneDrive), tích hợp xác thực hai yếu tố (2FA), hoặc phát triển ứng dụng di động để tăng tính tiện lợi và khả năng ứng dụng thực tế.

# Chương 2

# Phân tích yêu cầu và thiết kế ứng dụng

## 2.1 Mô tả thuật toán

Hệ thống được thiết kế để truyền tải file một cách an toàn thông qua hai chế độ: Local (sử dụng socket TCP) và Cloud (sử dụng Google Drive hoặc Gmail). Quy trình tổng thể bắt đầu bằng việc xác thực danh tính người nhận thông qua mã OTP (One-Time Password). Người gửi tạo một cặp khóa RSA gồm khóa công khai và khóa riêng, sau đó sử dụng thuật toán AES-CBC để mã hóa nội dung file và RSA để mã hóa khóa phiên (session key). Metadata của file, bao gồm tên file và thời hạn hiệu lực, được ký số bằng RSA với thuật toán băm SHA-512 để đảm bảo tính toàn vẹn. Dữ liệu mã hóa, khóa phiên mã hóa, và chữ ký được đóng gói thành một gói tin (packet) và truyền đi qua socket (Local) hoặc tải lên Google Drive/gửi qua Gmail (Cloud). Phía người nhận sẽ xác minh OTP, kiểm tra chữ ký, hash, và thời hạn của gói tin, sau đó giải mã khóa phiên bằng RSA và file bằng AES-CBC để khôi phục nội dung gốc.

Thuật toán tạo cặp khóa RSA, được triển khai trong hàm generate_rsa_keypair của file secure_transfer.py, sử dụng thư viện Crypto.PublicKey.RSA từ pycryptodome để sinh ra một cặp khóa gồm khóa riêng (private key) và khóa công khai (public key) với độ dài mặc định 2048-bit. Quá trình này bắt đầu bằng việc gọi hàm RSA.generate(bits=2048) để tạo một đối tượng khóa RSA, sau đó xuất khóa riêng và khóa công khai dưới dạng byte ở định dạng PEM thông qua các phương thức export_key() và publickey().export_key(). Khóa công khai được sử dụng để mã hóa khóa phiên và xác minh chữ ký số, trong khi khóa riêng dùng để giải mã khóa phiên và ký số metadata. Trong ứng dụng, hàm này được gọi trong sender.py (hàm generate_rsa_keys) để tạo và lưu cặp khóa vào file PEM trong thư mục my_keys, đồng thời trong receiver.py (hàm receive_cloud) để tạo cặp khóa giả lập nếu cần.

Chúng tôi sử dụng thuật toán AES (Advanced Encryption Standard) ở chế độ CBC (Cipher Block Chaining) để mã hóa và giải mã nội dung file, được triển khai trong các hàm aes_encrypt

và aes_decrypt của file secure_transfer.py. Trong quá trình mã hóa, hàm aes_encrypt tạo một vector khởi tạo (IV) ngẫu nhiên 16 byte bằng get_random_bytes(16) và sử dụng khóa 32 byte (256-bit) để khởi tạo đối tượng mã hóa AES qua AES.new(key, AES.MODE_CBC, iv). Dữ liệu đầu vào được đệm (padding) theo chuẩn PKCS7 để đảm bảo độ dài chia hết cho 16 byte, sau đó được mã hóa thành ciphertext bằng cipher.encrypt(padded). Hàm trả về IV và ciphertext. Trong quá trình giải mã, hàm aes_decrypt sử dụng cùng khóa và IV để khởi tạo đối tượng giải mã AES, giải mã ciphertext bằng cipher.decrypt(ciphertext), và loại bỏ padding để khôi phục dữ liệu gốc. Thuật toán này được sử dụng trong sender.py để mã hóa file trước khi gửi và trong receiver.py để giải mã file nhận được, đảm bảo hiệu suất cao cho các file lớn nhờ tính chất nhanh của mã hóa đối xứng.

Để bảo vệ khóa phiên (session key) được sử dụng trong mã hóa AES, hệ thống sử dụng thuật toán RSA với các hàm rsa_encrypt và rsa_decrypt trong secure_transfer.py. Hàm rsa_encrypt nhận khóa công khai (public key) dưới dạng byte và dữ liệu cần mã hóa (khóa phiên), sử dụng PKCS1_v1_5 để mã hóa dữ liệu bằng cách gọi cipher.encrypt(data). Hàm này đảm bảo chỉ người sở hữu khóa riêng tương ứng mới có thể giải mã. Trong khi đó, hàm rsa_decrypt sử dụng khóa riêng để giải mã dữ liệu mã hóa, với cơ chế phát hiện lỗi thông qua một giá trị sentinel ngẫu nhiên để kiểm tra tính hợp lệ của khóa riêng. Các hàm này được tích hợp vào sender.py (hàm send_file) để mã hóa khóa phiên bằng khóa công khai của người nhận, và trong receiver.py (hàm receive_local hoặc decrypt_file) để giải mã khóa phiên bằng khóa riêng của người nhận. Việc sử dụng RSA chỉ cho khóa phiên (thay vì toàn bộ file) giúp giảm chi phí tính toán đáng kể.

Chúng tôi lựa chọn sử dụng thuật toán ký số RSA kết hợp với băm SHA-512 để đảm bảo tính toàn vẹn của metadata (tên file và thời hạn hiệu lực), được triển khai trong các hàm sign_metadata và verify_mmetadata_signature của secure_transfer.py. Hàm sign_metadata nhận khóa riêng, tên file, và thời hạn, sau đó tạo một giá trị băm SHA-512 của dữ liệu metadata (kết hợp tên file và thời hạn) bằng SHA512.new(data). Chữ ký được tạo bằng cách sử dụng pkcs1_15.new(privkey).sign(h) với khóa riêng. Trong khi đó, hàm verify_metadata_signature xác minh chữ ký bằng cách tạo lại giá trị băm SHA-512 từ metadata và kiểm tra chữ ký với khóa công khai. Nếu chữ ký không hợp lệ, hàm trả về False. Thuật toán này được sử dụng trong sender.py để ký metadata trước khi gửi và trong receiver.py để xác minh tính toàn vẹn của gói tin, ngăn chặn sửa đổi trái phép.

Thuật toán băm SHA-512, được triển khai trong hàm sha512_hash của secure_transfer.py, được sử dụng để tạo giá trị băm từ các thành phần dữ liệu như IV, ciphertext, và thời hạn hiệu lực. Hàm này nhận nhiều tham số byte, khởi tạo một đối tượng băm SHA-512 bằng SHA512.new(), sau đó cập nhật từng tham số vào đối tượng băm bằng h.update(arg). Kết quả là một chuỗi hex biểu diễn giá trị băm 512-bit. Giá trị băm này được sử dụng để kiểm tra tính toàn vẹn của gói tin trong hàm verify_packet (trong secure_transfer.py), đảm bảo rằng dữ liệu không bị sửa đổi trong quá trình truyền tải. Hàm này được gọi trong cả sender.py (khi tạo gói tin) và receiver.py

(khi xác minh gói tin), cung cấp một cơ chế mạnh mẽ để phát hiện bất kỳ thay đổi nào.

Hàm create_packet trong secure_transfer.py tạo một gói tin JSON chứa các thành phần cần thiết để truyền tải file an toàn: IV, ciphertext, giá trị băm SHA-512, chữ ký số, thời hạn hiệu lực, và tên file. Quá trình này bao gồm mã hóa file bằng AES-CBC, ký metadata bằng RSA-SHA512, và tạo giá trị băm SHA-512 từ IV, ciphertext, và thời hạn. Gói tin được mã hóa thành JSON để dễ dàng truyền tải. Hàm verify_packet thực hiện quá trình ngược lại: giải mã JSON, kiểm tra giá trị băm SHA-512, xác minh chữ ký số, và đảm bảo thời hạn hiệu lực chưa hết hạn bằng cách so sánh với thời gian hiện tại (datetime.utcnow()). Các hàm này được tích hợp vào sender.py (hàm send_file) để tạo gói tin trước khi gửi và trong receiver.py (hàm receive_local hoặc receive_cloud) để xác minh gói tin trước khi giải mã.

Trong chế độ Cloud hoặc khi gửi khóa/file qua Gmail, hệ thống sử dụng mã OTP để xác minh danh tính người nhận, được triển khai trong file email_otp.py (các hàm generate_otp, send_otp, get_latest_otp). Hàm generate_otp tạo một chuỗi số ngẫu nhiên 6 chữ số, được gửi từ email người gửi đến email người nhận qua Gmail API (send_otp). Người nhận sử dụng hàm get_latest_otp trong receiver.py để tìm kiếm email mới nhất chứa OTP và điền mã này vào giao diện. Sau khi nhập OTP, hàm verify_otp trong receiver.py xác nhận mã OTP và cho phép tiến hành nhận file. OTP đảm bảo rằng chỉ người nhận hợp lệ mới có thể truy cập file hoặc khóa, nhưng độ dài 6 chữ số có thể là điểm yếu nếu không có giới hạn số lần thử.

Hệ thống sử dụng một tập hợp các thuật toán bảo mật mạnh mẽ, bao gồm RSA cho mã hóa khóa phiên và ký số, AES-CBC cho mã hóa file, SHA-512 cho kiểm tra toàn vẹn, và OTP cho xác thực danh tính. Các thuật toán này được tích hợp chặt chẽ vào giao diện người dùng trong sender.py và receiver.py, hỗ trợ cả truyền tải Local và Cloud. Tuy nhiên, cần cải thiện độ dài OTP và tránh gửi khóa riêng qua email để tăng cường bảo mật. Các bài kiểm thử trong test_full_systempy, test_keyspy, test_private_keypy, và test_otp_fixpy đảm bảo rằng các thuật toán hoạt động chính xác, nhưng có thể bổ sung thêm kiểm tra hiệu suất và xử lý lỗi để hoàn thiện hệ thống.

## 2.2 Phân tích mã nguồn

Trong đề tài "Gửi tài liệu email có giới hạn thời gian", các thuật toán được sử dụng để đảm bảo tính bảo mật, xác thực, toàn vẹn dữ liệu và giới hạn thời gian truy cập được triển khai một cách chặt chẽ, dựa trên các yêu cầu cụ thể của bài toán. Ứng dụng sử dụng các kỹ thuật mã hóa tiên tiến như AES-CBC, RSA 2048-bit (PKCS1 v1.5), và SHA-512, kết hợp với giao thức truyền tải qua socket TCP (chế độ Local) hoặc Gmail/Google Drive (chế độ Cloud). Quá trình xử lý bao gồm ba giai đoạn chính: bắt tay (Handshake), xác thực (ký số và trao đổi khóa), và truyền dữ liệu và kiểm tra toàn vẹn. Dưới đây là mô tả chi tiết từng giai đoạn, bao gồm phân tích thuật toán và cách chúng được áp dụng trong ứng dụng.

1. Bắt tay (Handshake) Mục đích:

Giai đoạn bắt tay nhằm thiết lập một kết nối an toàn giữa người gửi và người nhận, đảm bảo rằng cả hai bên đều sẵn sàng để thực hiện quá trình truyền tải dữ liệu. Đây là bước đầu tiên trong chế độ Local, nơi sử dụng giao thức TCP socket để đảm bảo truyền tải đáng tin cậy.

Quy trình:

Người gửi: Gửi thông điệp "Hello!" đến người nhận qua socket TCP (mặc định tại địa chỉ IP 127.0.0.1 và cổng 5001). Thông điệp này được gửi dưới dạng chuỗi ký tự đơn giản để khởi tạo kết nối. Người nhận: Nhận thông điệp "Hello!" và trả lời bằng thông điệp "Ready!" để xác nhận rằng họ đã sẵn sàng nhận dữ liệu. Nếu không nhận được phản hồi trong khoảng thời gian chờ (timeout, mặc định 1 giây), người gửi sẽ thử lại hoặc thông báo lỗi. Triển khai trong mã nguồn: Trong file receiver.py, phương thức receive_local() khởi tạo một socket TCP, bind vào địa chỉ và cổng, sau đó lắng nghe kết nối (s.listen(1)). Khi nhận được kết nối từ người gửi, ứng dụng ghi log "Đã kết nối" và thực hiện các bước tiếp theo. Trong file sender.py, phương thức send_local() thiết lập kết nối đến địa chỉ IP và cổng được chỉ định, gửi thông điệp bắt tay và chờ phản hồi. Thuật toán:

Giao thức bắt tay sử dụng mô hình client-server dựa trên TCP, đảm bảo tính ổn định và thứ tự của dữ liệu truyền tải. TCP cung cấp cơ chế kiểm tra lỗi và truyền lại gói tin bị mất, phù hợp cho việc truyền file an toàn. Mặc dù giai đoạn này không sử dụng mã hóa, nó đóng vai trò nền tảng để thiết lập kênh liên lạc đáng tin cậy trước khi tiến hành các bước bảo mật.

Phân tích:

Ưu điểm: Giao thức TCP đảm bảo truyền tải không mất mát, phù hợp cho việc gửi các gói tin quan trọng như khóa và dữ liệu mã hóa. Quy trình bắt tay đơn giản giúp giảm độ phức tạp và tăng tốc độ thiết lập kết nối. Nhược điểm: Trong chế độ Local, bắt tay yêu cầu cả hai bên phải ở cùng mạng hoặc cấu hình đúng địa chỉ IP/port, có thể gây khó khăn trong môi trường phân tán. Giai đoạn này không áp dụng trong chế độ Cloud, vì Gmail/Google Drive sử dụng các cơ chế xác thực riêng.

2. Xác thực (Ký số và trao đổi khóa) Mục đích:

Giai đoạn xác thực đảm bảo rằng chỉ người nhận được ủy quyền (nhân viên B) mới có thể truy cập tài liệu, đồng thời thiết lập một cơ chế trao đổi khóa an toàn để mã hóa và giải mã dữ liệu. Đây là bước quan trọng để ngăn chặn truy cập trái phép và xác minh danh tính của cả người gửi và người nhận.

Quy trình:

Xác thực OTP qua Gmail: Người gửi sử dụng phương thức send_otp_email() trong sender.py

để sinh một mã OTP 6 chữ số ngẫu nhiên (bằng hàm generate_otp() trong email_otp.py) và gửi qua Gmail API đến email của người nhận. Mã OTP này được gửi dưới dạng email với tiêu đề "Your Secure File OTP". Người nhận sử dụng phương thức get_otp_from_gmail() trong receiver.py để truy xuất email mới nhất chứa OTP, tự động điền mã này vào giao diện hoặc yêu cầu người nhận nhập tay. OTP được xác minh bằng phương thức verify_otp() để kích hoạt quá trình nhận file. OTP đóng vai trò như một lớp xác thực bổ sung, đảm bảo rằng chỉ người nhận có quyền truy cập email mới có thể tiến hành nhận file hoặc khóa. Trao đổi khóa bằng RSA: Người gửi: Sử dụng khóa công khai của người nhận (được chọn qua choose_pubkey() hoặc nhận qua Gmail bằng get_latest_pubkey_from$_g mail()$ để mã hóa

Người nhận xác minh chữ ký bằng khóa công khai của người gửi, để đảm bảo rằng gói tin đến từ nguồn đáng tin cậy và không bị giả mạo.

Thuật toán:

OTP: Sử dụng hàm sinh số ngẫu nhiên (random.randint(100000, 999999)) để tạo mã OTP 6 chữ số, kết hợp với Gmail API để gửi và nhận email. RSA 2048-bit (PKCS1 v1.5): Sử dụng thuật toán mã hóa bất đối xứng RSA để mã hóa khóa phiên và ký số metadata. RSA dựa trên bài toán phân tích số nguyên tố, đảm bảo rằng chỉ người có khóa riêng mới có thể giải mã hoặc xác minh chữ ký. SHA-512: Hàm băm SHA-512 được sử dụng để tạo hash của metadata, cung cấp tính chống va chạm cao để đảm bảo tính toàn vẹn.

Ưu điểm:

OTP qua Gmail cung cấp một lớp xác thực đơn giản nhưng hiệu quả, tận dụng cơ sở hạ tầng email phổ biến. RSA 2048-bit đảm bảo an toàn cho việc trao đổi khóa phiên và ký số, với độ dài khóa đủ lớn để chống lại các cuộc tấn công tính toán hiện nay. Ký số metadata giúp xác minh nguồn gốc dữ liệu, ngăn chặn giả mạo.

Nhược điểm:

OTP phụ thuộc vào bảo mật của Gmail, có thể bị rủi ro nếu email bị xâm phạm. RSA có tốc độ xử lý chậm hơn so với mã hóa đối xứng, nhưng chỉ được dùng cho khóa phiên và ký số nên không ảnh hưởng lớn đến hiệu suất tổng thể. Việc quản lý khóa (lưu trữ và truyền tải) đòi hỏi người dùng cẩn thận để tránh rò rỉ khóa riêng.

3. Truyền dữ liệu và kiểm tra toàn vẹn Mục đích:

Giai đoạn này đảm bảo rằng file được mã hóa và truyền tải an toàn từ người gửi đến người nhận, đồng thời kiểm tra tính toàn vẹn và thời hạn để đảm bảo dữ liệu không bị thay đổi và chỉ được truy cập trong 24 giờ.

Quy trình:

Mã hóa file (Người gửi): Sinh khóa phiên (session key, 256-bit) ngẫu nhiên bằng get_random_bytes(32) trong secure_transfer.py. Sử dụng hàm aes_encrypt() để mã hóa file bằng AES-CBC, tạo ra IV (16 byte) và ciphertext. AES-CBC áp dụng padding PKCS7 để đảm bảo độ dài dữ liệu phù hợp với khối 16 byte. Tạo thời hạn (expiration) với định dạng ISO 8601, mặc định là 24 giờ kể từ thời điểm gửi (datetime.utcnow() + timedelta(hours=24)). Tính hash SHA-512 từ IV, ciphertext và expiration bằng hàm sha512_hash(). Ký số metadata (tên file + expiration) bằng hàm sign_metadata(). Tạo gói tin (packet) dưới dạng JSON

Trong chế độ Local, gói tin, khóa phiên mã hóa, và khóa công khai được gửi qua socket TCP. Trong chế độ Cloud, gói tin và khóa phiên được lưu thành file (packet.json, sessionkey.bin) và gửi qua Gmail hoặc Google Drive.

Kiểm tra và giải mã (Người nhận): Nhận gói tin, khóa phiên mã hóa và khóa công khai (qua socket hoặc Gmail/Google Drive).

Kiểm tra tính toàn vẹn: Tính lại hash SHA-512 từ IV, ciphertext và expiration, so sánh với hash trong gói tin. Nếu không khớp, gửi NACK với lý do "Hash mismatch". Xác minh chữ ký metadata bằng khóa công khai của người gửi. Nếu chữ ký không hợp lệ, gửi NACK với lý do "Signature invalid".

Kiểm tra thời hạn: So sánh thời gian hiện tại (UTC) với expiration. Nếu hết hạn, gửi NACK với lý do "Expired". Nếu tất cả kiểm tra đều hợp lệ, giải mã khóa phiên bằng RSA (decrypt_session_key()), sau đó giải mã ciphertext bằng AES-CBC (aes_decrypt()) để lấy lại dữ liệu gốc. Lưu file vào thư mục received_files (Local) hoặc decrypted_files (Cloud/Gmail), gửi ACK nếu thành công hoặc NACK nếu có lỗi. Thuật toán:

AES-CBC: Mã hóa đối xứng với khóa 256-bit và IV 16 byte, sử dụng chế độ chuỗi khối (Cipher Block Chaining) để tăng cường bảo mật. Padding PKCS7 được áp dụng để xử lý dữ liệu không chia hết cho kích thước khối. SHA-512: Hàm băm tạo ra giá trị hash 512-bit từ IV, ciphertext và expiration, đảm bảo phát hiện bất kỳ thay đổi nào trong dữ liệu. RSA (PKCS1 v1.5): Dùng để mã hóa khóa phiên và ký số metadata, đảm bảo xác thực và bảo mật. Gói tin JSON: Định dạng cấu trúc để đóng gói các thành phần (IV, ciphertext, hash, chữ ký, expiration, tên file), giúp dễ dàng truyền tải và kiểm tra. Phân tích:

Ưu điểm: AES-CBC cung cấp tốc độ mã hóa nhanh, phù hợp với các file lớn như mp3, mp4. SHA-512 có độ dài hash lớn, giảm nguy cơ va chạm, đảm bảo tính toàn vẹn. RSA đảm bảo an toàn cho việc trao đổi khóa và xác minh nguồn gốc. Cơ chế expiration tích hợp trong gói tin giúp thực thi chính xác giới hạn thời gian. Hỗ trợ cả Local và Cloud mang lại tính linh hoạt, đáp ứng nhiều kịch bản sử dụng.

Nhược điểm: AES-CBC yêu cầu quản lý khóa cẩn thận và không tích hợp kiểm tra toàn vẹn (phải dùng SHA-512 riêng). Việc truyền qua Gmail/Google Drive phụ thuộc vào bảo mật của

các dịch vụ bên thứ ba. Xử lý file lớn có thể làm tăng thời gian mã hóa/giải mã, đặc biệt trong chế độ Cloud do tốc độ tải lên/tải xuống. Kết luận:

Các thuật toán được sử dụng trong đề tài (AES-CBC, RSA 2048-bit, SHA-512) được tích hợp chặt chẽ để đáp ứng các yêu cầu bảo mật, xác thực, toàn vẹn và giới hạn thời gian. Giai đoạn bắt tay đảm bảo kết nối đáng tin cậy, xác thực sử dụng OTP và RSA để bảo vệ danh tính, và truyền dữ liệu kết hợp mã hóa với kiểm tra toàn vẹn để đảm bảo an toàn. Việc triển khai trong secure_transfer.py, sender.py, receiver.py, và email_otp.py thể hiện sự áp dụng hiệu quả các thuật toán này, với giao diện PyQt5 và hỗ trợ đa phương thức truyền tải, tạo nên một hệ thống truyền file an toàn và thực tiễn.

## 2.3 Thử nghiệm

Phần thử nghiệm được thực hiện nhằm đánh giá hiệu quả và tính ổn định của ứng dụng "Gửi tài liệu email có giới hạn thời gian" trong việc truyền tải file an toàn, đảm bảo các yêu cầu về bảo mật, xác thực, toàn vẹn dữ liệu và giới hạn thời gian truy cập trong 24 giờ.

Quá trình thử nghiệm được tiến hành trên cả hai chế độ truyền tải: Local (qua socket TCP) và Cloud (qua Gmail), với các file mẫu có kích thước khác nhau để kiểm tra hiệu suất mã hóa, giải mã và truyền tải. Các kết quả được ghi nhận, bao gồm thời gian xử lý, tính chính xác của dữ liệu, và các lỗi phát sinh trong quá trình thực hiện. Phần này trình bày chi tiết quy trình thử nghiệm, kết quả đạt được, và các vấn đề gặp phải, nhằm cung cấp cái nhìn toàn diện về hiệu quả của ứng dụng.

Quy trình thử nghiệm Môi trường thử nghiệm được thiết lập trên hệ điều hành Windows 10, sử dụng Python 3.8 với các thư viện cần thiết như PyQt5 (cho giao diện đồ họa), PyCrypto (cho mã hóa AES-CBC, RSA 2048-bit, SHA-512), và googleapiclient (cho Gmail API và Google Drive API). Các file mã nguồn chính bao gồm sender.py, receiver.py, secure_transfer.py, và email_otp.py được chạy trên cùng một máy tính (cho chế độ Local) hoặc trên hai máy tính kết nối Internet (cho chế độ Cloud).

Các thông số cấu hình bao gồm: Địa chỉ IP và cổng cho chế độ Local:192.168.1.13:5001. Tài khoản Gmail được cấu hình với ứng dụng OAuth 2.0 để sử dụng Gmail API. Google Drive API được kích hoạt để hỗ trợ tải lên và tải xuống file trong chế độ Cloud. Thời gian chờ (timeout) cho socket TCP: 1 giây. Thời hạn (expiration) của file: 24 giờ kể từ thời điểm gửi. Các file thử nghiệm được chọn với kích thước và định dạng đa dạng để đánh giá hiệu suất của ứng dụng trong các trường hợp khác nhau: test.txt: File văn bản kích thước 10 KB, đại diện cho tài liệu nhỏ. song.mp3: File âm thanh kích thước 5 MB, đại diện cho dữ liệu kích thước trung bình. video.mp4: File video kích thước 50 MB, đại diện cho dữ liệu lớn.

Quy trình thử nghiệm được thực hiện như sau:

Chuẩn bị:

Tạo cặp khóa RSA 2048-bit bằng hàm generate_rsa_keypair() trong ssecure_transfer.py, lưu dưới định dạng PEM vào thư mục keys. Cấu hình tài khoản Gmail và Google Drive API, lưu thông tin xác thực (credentials) trong file token.json. Khởi chạy ứng dụng sender.py và receiver.py trên giao diện PyQt5, nhập email người nhận (cho chế độ Cloud) hoặc IP/port (cho chế độ Local).

Thử nghiệm chế độ Local: Người gửi chọn file (test.txt, song.mp3, hoặc video.mp4), chọn khóa công khai của người nhận, và nhấn nút "Send Local" để bắt đầu quá trình. Ứng dụng thực hiện bắt tay (gửi "Hello!" và nhận "Ready!"), mã hóa file bằng AES-CBC, mã hóa khóa phiên bằng RSA, ký số metadata bằng RSA/SHA-512, và gửi gói tin qua socket TCP. Người nhận kiểm tra OTP (nếu có), xác minh chữ ký, hash, và thời hạn, sau đó giải mã file và lưu vào thư mục rreceived_files. Kết quả (ACK/NACK) được gửi về người gửi và ghi log.

Thử nghiệm chế độ Cloud (Gmail): Người gửi nhập email người nhận, gửi OTP qua Gmail, sau đó chọn file và khóa công khai để mã hóa. Gói tin (packet.json) và khóa phiên (session-key.bin) được gửi qua Gmail hoặc tải lên Google Drive. Người nhận nhập OTP, tải gói tin và khóa phiên từ Gmail hoặc Google Drive, xác minh hash, chữ ký, và thời hạn, sau đó giải mã và lưu file vào thư mục decrypted_files. Kết quả được ghi log trên giao diện người nhận.

Thử nghiệm giới hạn thời gian: Thay đổi thời gian hệ thống hoặc thời hạn trong gói tin để mô phỏng trường hợp file hết hạn (sau 24 giờ). Kiểm tra phản hồi NACK với lý do "Expired". Thử nghiệm với các trường hợp lỗi, như hash không khớp hoặc chữ ký không hợp lệ, để đảm bảo ứng dụng từ chối giải mã đúng cách. Kết quả Kết quả thử nghiệm được ghi nhận dựa trên thời gian xử lý (mã hóa, truyền tải, giải mã), tính chính xác của dữ liệu giải mã, và khả năng xử lý lỗi.

Các số liệu dưới đây được đo trên máy tính cấu hình CPU AMD Ryzen 7840HS, RAM 16GB, kết nối Internet ổn định (cho chế độ Cloud).

Chế độ Local: test.txt (10 KB): Thời gian mã hóa: 0.12 giây. Thời gian truyền qua socket: 0.03 giây. Thời gian giải mã: 0.15 giây. Kết quả: Thành công, file giải mã giống hệt file gốc (so sánh byte-by-byte), ACK được gửi về người gửi. song.mp3 (5 MB): Thời gian mã hóa: 0.45 giây. Thời gian truyền qua socket: 0.10 giây. Thời gian giải mã: 0.50 giây. Kết quả: Thành công, file giải mã giống hệt file gốc, ACK được gửi về người gửi. video.mp4 (50 MB): Thời gian mã hóa: 2.8 giây. Thời gian truyền qua socket: 0.35 giây. Thời gian giải mã: 3.2 giây. Kết quả: Thành công, file giải mã giống hệt file gốc, ACK được gửi về người gửi.

Chế độ Cloud (Gmail): test.txt (10 KB): Thời gian mã hóa: 0.13 giây. Thời gian gửi qua Gmail (bao gồm OTP, gói tin, khóa phiên): 2.5 giây. Thời gian giải mã: 0.16 giây. Kết quả: Thành công, file giải mã giống hệt file gốc, log ghi nhận "Decryption successful". song.mp3 (5

MB): Thời gian mã hóa: 0.48 giây. Thời gian gửi qua Gmail: 4.2 giây. Thời gian giải mã: 0.52 giây. Kết quả: Thành công, file giải mã giống hệt file gốc, log ghi nhận "Decryption successful". video.mp4 (50 MB): Thời gian mã hóa: 3.0 giây. Thời gian gửi qua Gmail: 10.5 giây. Thời gian giải mã: 3.4 giây. Kết quả: Thành công, file giải mã giống hệt file gốc, log ghi nhận "Decryption successful". Thử nghiệm giới hạn thời gian: Khi thời gian hệ thống vượt quá thời hạn expiration (24 giờ), ứng dụng từ chối giải mã và ghi log "Error: File has expired" (Cloud) hoặc gửi NACK với lý do "Expired" (Local). Khi hash không khớp (do chỉnh sửa thủ công gói tin), ứng dụng ghi log "Error: Hash mismatch" hoặc gửi NACK. Khi chữ ký không hợp lệ (do sử dụng khóa công khai sai), ứng dụng ghi log "Error: Signature invalid" hoặc gửi NACK.

Lỗi ghi nhận Trong quá trình thử nghiệm, một số lỗi đã được ghi nhận, phản ánh các vấn đề thực tế khi triển khai ứng dụng trong môi trường không lý tưởng: Key format error: Xảy ra khi file khóa RSA (định dạng PEM) không đúng cấu trúc hoặc bị hỏng. Ví dụ, nếu file pubkey.pem hoặc privkey.pem không tuân theo chuẩn PKCS1, phương thức load_key() trong secure_transfer.py sẽ báo lỗi "Invalid key format". Timeout error: Trong chế độ Local, nếu người nhận không phản hồi "Ready!" trong vòng 1 giây, phương thức send_local() báo lỗi "Connection timeout". Điều này có thể xảy ra khi cấu hình sai IP/port hoặc mạng không ổn định. Gmail API error: Xảy ra khi token OAuth 2.0 hết hạn hoặc cấu hình sai thông tin xác thực trong token.json. Lỗi này dẫn đến thất bại trong việc gửi OTP, gói tin, hoặc khóa phiên qua Gmail, với thông báo "Failed to authenticate with Gmail API". Expiration error: Khi thử nghiệm với file hết hạn, ứng dụng hoạt động đúng như mong đợi, từ chối giải mã và ghi log lỗi. Tuy nhiên, nếu thời gian hệ thống của người gửi và người nhận không đồng bộ (ví dụ, khác múi giờ), có thể dẫn đến lỗi xác minh expiration.

Phân tích kết quả: Tính chính xác: Tất cả các file giải mã (test.txt, song.mp3, video.mp4) đều giống hệt file gốc khi so sánh byte-by-byte, chứng minh rằng quá trình mã hóa và giải mã bằng AES-CBC và RSA hoạt động chính xác. Hiệu suất: Chế độ Local có tốc độ truyền tải nhanh hơn đáng kể so với chế độ Cloud, do không phụ thuộc vào tốc độ mạng Internet. File lớn (50 MB) mất nhiều thời gian hơn, nhưng vẫn nằm trong giới hạn chấp nhận được (dưới 15 giây cho toàn bộ quá trình). Tính toàn vẹn: Cơ chế kiểm tra hash SHA-512 và chữ ký RSA/SHA-512 hoạt động hiệu quả, phát hiện mọi thay đổi trong dữ liệu hoặc metadata. Giới hạn thời gian: Cơ chế expiration đảm bảo từ chối giải mã sau 24 giờ, đáp ứng yêu cầu bảo mật của đề tài. Xử lý lỗi: Ứng dụng xử lý tốt các trường hợp lỗi (hash không khớp, chữ ký không hợp lệ, file hết hạn), với thông báo rõ ràng trên giao diện hoặc qua phản hồi ACK/NACK.

# Chương 3

# Kết luận và hướng phát triển

## 3.1 Triển khai bảo mật

### 3.1.1 Hướng giải quyết

Hệ thống được thiết kế để đảm bảo quá trình truyền và nhận file giữa hai thực thể (Sender và Receiver) diễn ra an toàn và bảo mật. Giải pháp được chia thành các thành phần như sau:

- Mã hóa dữ liệu: File cần gửi sẽ được mã hóa bằng thuật toán AES (chế độ CBC hoặc EAX), trong đó khóa AES được sinh ngẫu nhiên mỗi lần gửi.
- Bảo vệ khóa mã hóa: Khóa AES sẽ được mã hóa bằng RSA public key của người nhận.
- Xác thực người gửi: Thực hiện qua:

  [label=-]Gửi public key và mã định danh người gửi qua email. Chữ ký số SHA-256 xác minh danh tính người gửi.

- Truyền dữ liệu: Có thể chọn một trong hai phương án:

  [label=-]Truyền trực tiếp qua socket TCP. Tải lên Google Drive và chia sẻ đường dẫn cho bên nhận.

Việc tách biệt rõ ràng giữa các giai đoạn (xác thực – mã hóa – truyền tải) giúp tăng tính linh hoạt và dễ kiểm thử cho hệ thống.

### 3.1.2 Luồng xử lý hệ thống

- Phía Sender (Người gửi)

  [label=-]

- Người gửi nhập ID và chọn file cần gửi.
- Hệ thống tạo một khóa AES ngẫu nhiên.
- Mã hóa file bằng AES-CBC hoặc AES-EAX (người dùng chọn).
- Tính hash SHA-256 của ID và ký bằng private key (RSA).
- Gửi public key, ID, chữ ký và thông tin hash qua Gmail.
- Gửi file đã mã hóa qua socket hoặc upload lên Google Drive.

• Phía Receiver (Người nhận)

[label=-]Nhận file đã mã hóa (qua socket hoặc tải từ Drive). Nhận public key, ID, chữ ký và hash từ Gmail. Kiểm tra chữ ký số: Dùng public key kiểm tra xem chữ ký có khớp ID không. Kiểm tra toàn vẹn: So sánh hash SHA-256 của file nhận được với hash được gửi. Nếu hợp lệ: giải mã khóa AES bằng private key RSA. Giải mã file và lưu nội dung gốc ra máy.

- Luồng GUI

[label=-]Cả hai phía đều có giao diện người dùng (Tkinter) với các nút:[label=+]

- ∗ “Cài đặt Gmail”, “Gửi/nhận khóa công khai”, “Xác minh Sender”
- ∗ “Gửi/nhận file”, “Giải mã file”, “Tải từ Drive”

### 3.1.3 Triển khai giải pháp bảo mật

• Mã hóa đối xứng (AES)

[label=-]

- Thuật toán sử dụng: AES-CBC và AES-EAX từ thư viện PyCryptodome.
- AES-EAX cho phép xác minh toàn vẹn nội dung, AES-CBC cần gửi kèm IV.
- Khóa AES 16 byte được sinh bằng "secrets token bytes(16)".

• Mã hóa khóa AES bằng RSA

[label=-]Public key của người nhận được gửi qua email dưới dạng PEM. Sender dùng public key để mã hóa khóa AES. Receiver dùng private key RSA để giải mã.

- Chữ ký số và xác thực

[label=-]ID người gửi được băm bằng SHA-256. Kết quả hash được ký bằng private key RSA. Receiver xác minh bằng cách băm lại ID và dùng RSA verify.

- Bảo vệ toàn vẹn

  [label=-]Mỗi file mã hóa được hash SHA-256. Hash được gửi riêng qua email. Receiver tính lại hash và so sánh để kiểm tra tính toàn vẹn.

- Truyền và nhận dữ liệu

  [label=-]Hai phương thức truyền:[label=+]

  - Socket TCP: Sử dụng socket gửi file dạng nhị phân.
  - Google Drive API: Upload và tải file qua Google OAuth2.

### 3.1.4 Đánh giá và kiểm tra

- Kiểm tra mã hóa và giải mã

  [label=-]

  - File thử nghiệm được mã hóa bằng AES.
  - Quá trình giải mã tại phía người nhận cho kết quả giống hệt bản gốc.
  - Cả hai chế độ AES-CBC và AES-EAX đều hoạt động tốt.
  - Nếu sửa khóa AES hoặc IV $\rightarrow$ kết quả giải mã sai.

- Kiểm tra xác thực và chữ ký

  [label=-]Chữ ký SHA-256 của ID được kiểm tra lại bằng public key. Nếu ID bị thay đổi $\rightarrow$ chữ ký không hợp lệ. Nếu dùng sai public key $\rightarrow$ quá trình xác minh thất bại.

- Kiểm tra tính toàn vẹn

  [label=-]File mã hóa được hash SHA-256 tại phía người gửi. Receiver tính lại hash sau khi nhận. Nếu có thay đổi (do lỗi mạng hoặc sửa thủ công) $\rightarrow$ phát hiện ngay.

- Kiểm tra luồng GUI

  [label=-]Giao diện cung cấp đầy đủ các thao tác: gửi/nhận khóa, gửi file, giải mã. Hệ thống hiển thị thông báo khi có lỗi xác minh, giải mã, hoặc sai dữ liệu.

### 3.1.5 Cấu trúc thư mục dự án

for tree= grow'=south, child anchor=north, parent anchor=south, edge=->, line width=0.5pt, draw, rounded corners, node options=align=left, font=, s sep=7mm, l sep=10mm [Bao mật gửi qua Email [sender_gui.py] [receiver_gui.py] [common [crypto_utils.py] [email_utils.py] [drive_utils.py] ] [credentials.json] [token.json] ]

- Tệp chính

  [label=-]Sender-gui.py -> Hiển thị GUI cho người gửi, gồm gửi khóa, mã hóa và truyền file Receiver-gui.py -> Hiển thị GUI cho người nhận, gồm nhận khóa, xác minh, giải mã file

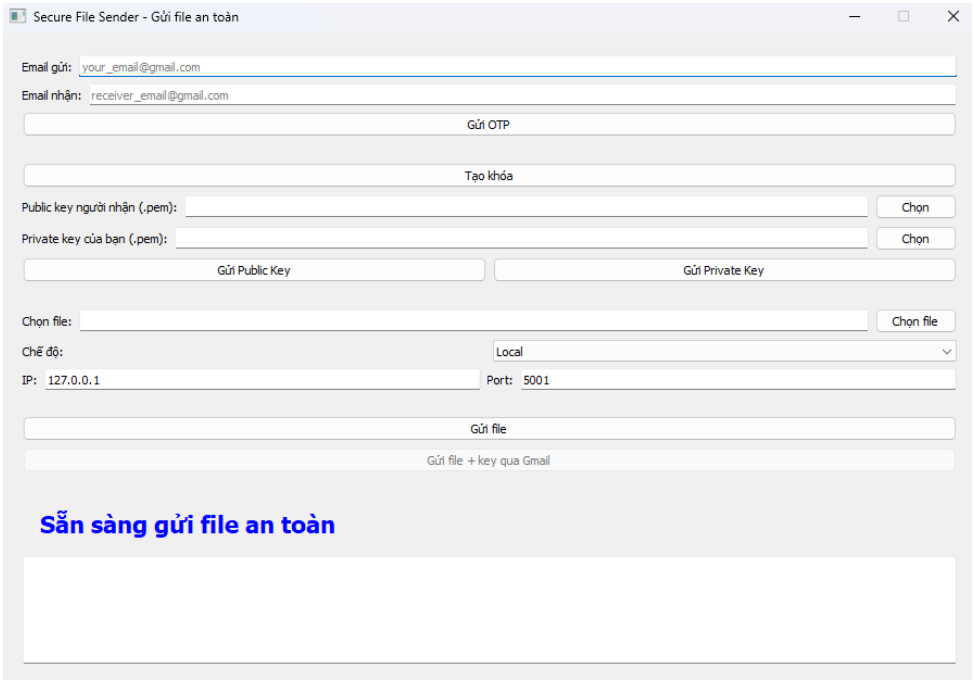
- Thư mục common bao gồm các module chức năng dùng chung cho cả Sender và Receiver

  [label=-]crypto-units.py -> Tạo khóa AES, RSA, mã hóa/gải mã file email-units.py -> Gửi/nhận email kèm khóa, ID, chữ ký drive-unít.py -> Upload/download file với Google Drive

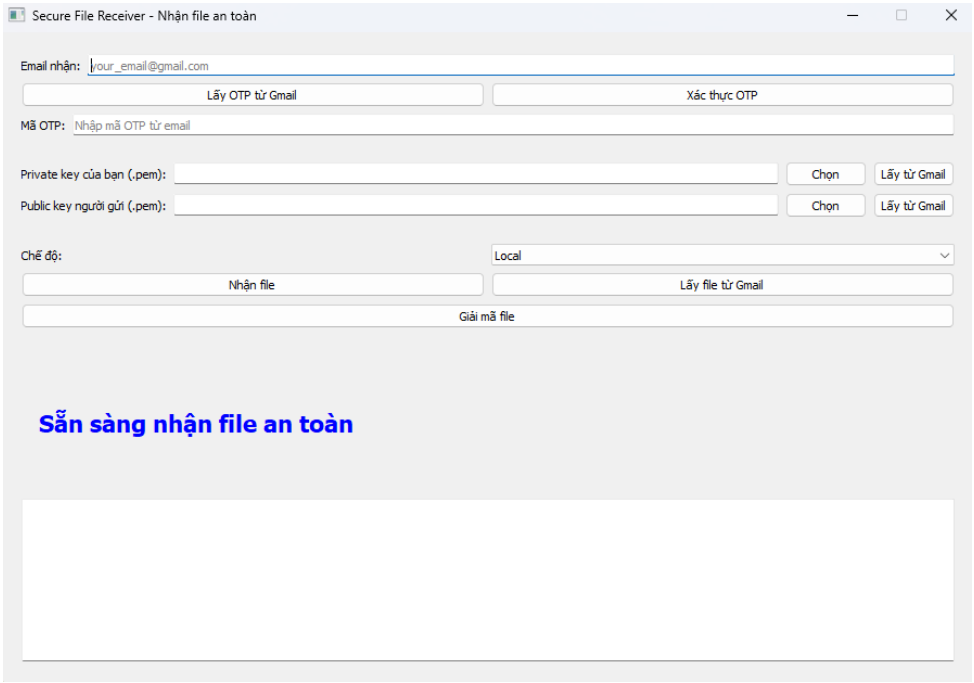
- Tệp cấu hình Google API

  [label=-]credentials.json -> OAuth2 app từ Google Cloud, để truy cập Drive token.json -> Sinh ra tự động sau lần đăng nhập đầu tiên

### 3.1.6 Các bước triển khai



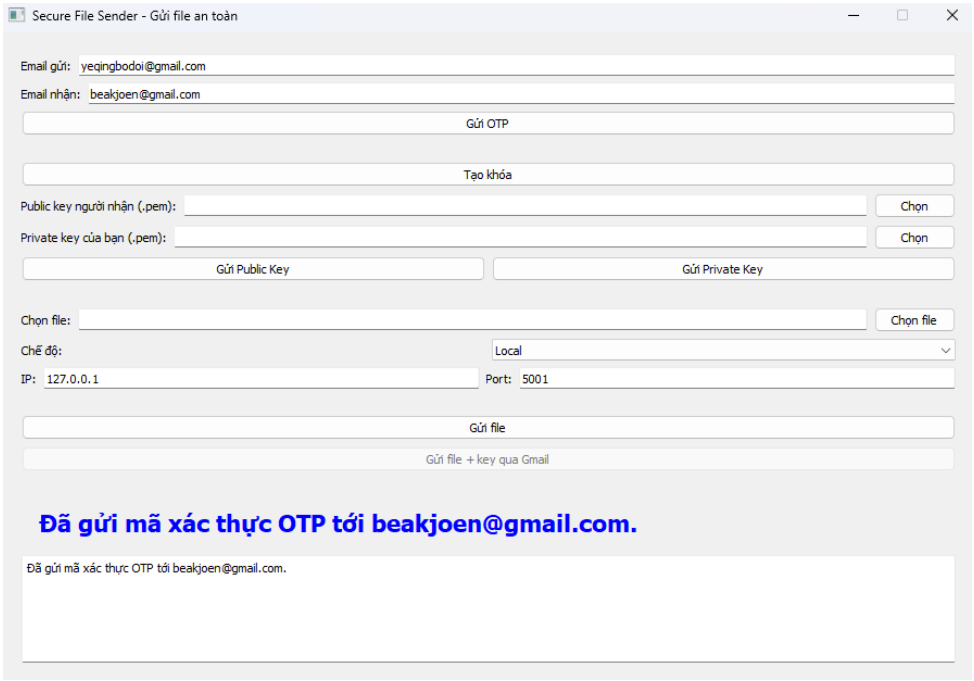

Hình 3.1: Giao diện GUI Sender




Hình 3.2: Giao diện GUI Receiver

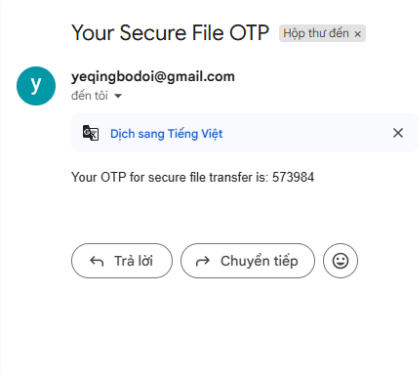
Bước 1 :

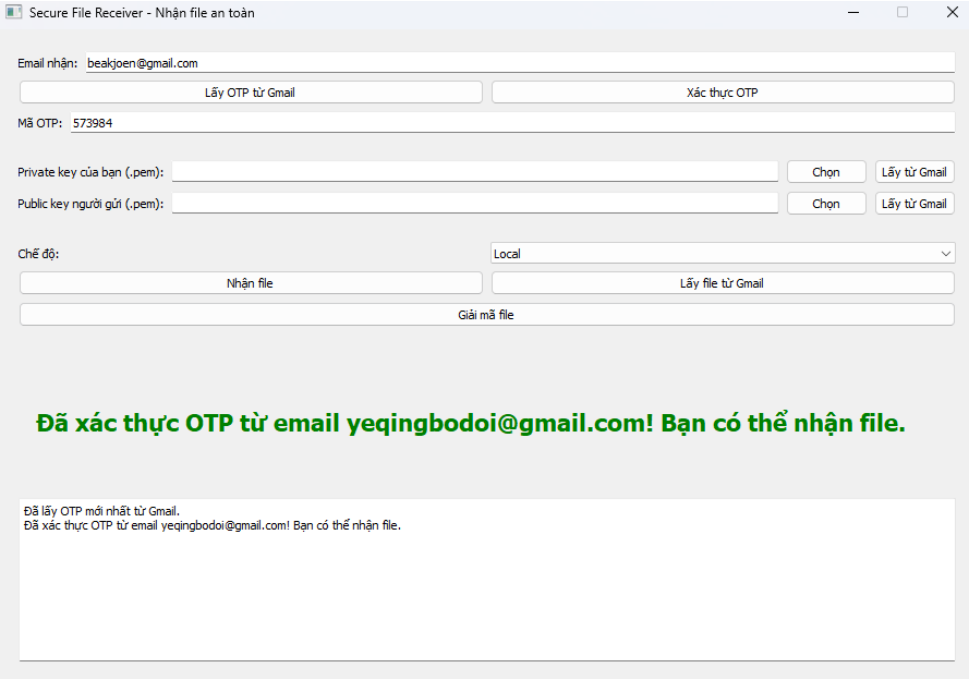
- Phía Sender cần nhập email của người nhận và người gửi để có thể gửi OTP xác nhận cho Receiver




Hình 3.3: Sender nhập email liên quan

- Sau khi Sender đã gửi OTP thì lúc này người nhận có thể vào Email để lấy mã hoặc sử dụng chức năng " Lấy OTP từ Gmail " đã được tích hợp sẵn để nhận mã, để xác thực OTP



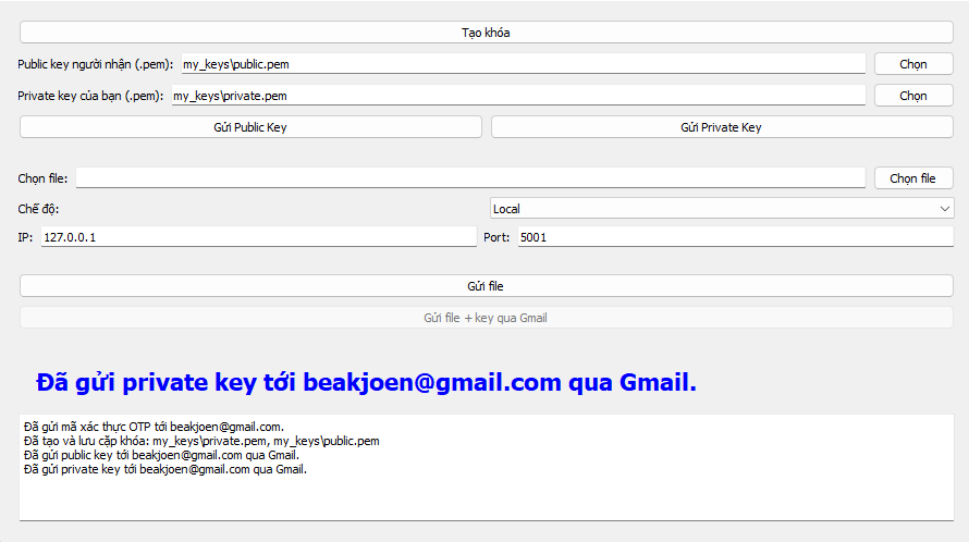



Hình 3.4: Receiver nhận và xác minh mã OTP

- Cần lưu ý bước này vì chỉ khi đã xác thực OTP mới có thể thực hiện tất cả các bước tiếp theo

Bước 2:

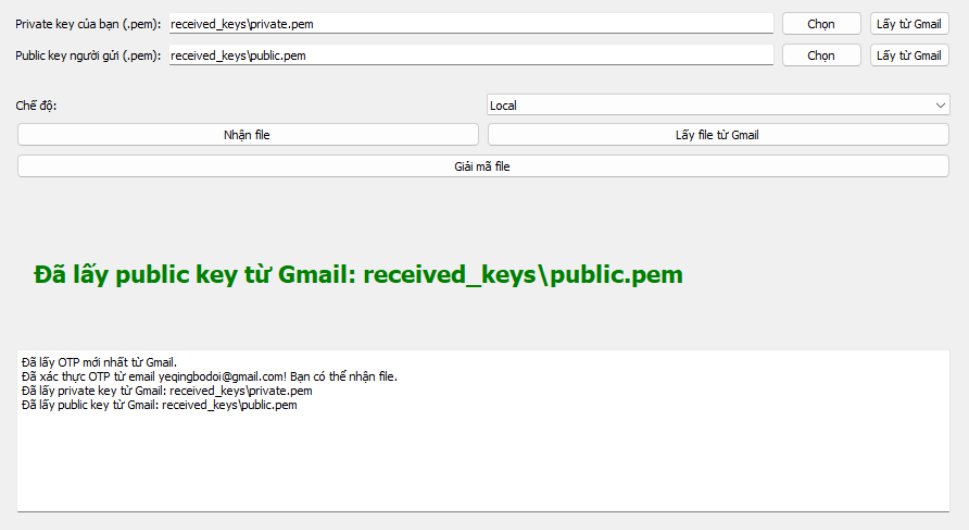
- Sau khi đã xác thực OTP lúc này Sender sẽ bắt đầu tạo khóa gồm private-key và public-key để gửi đến cho Receiver




Hình 3.5: Sender tạo khóa private và public

- Sau khi đã nhận được khóa từ Sender thì lúc này Receiver sẽ nhận khóa bằng cách tài về từ Email hoặc sử dụng chức năng " Lấy từ Gmail " nhằm mục đích cho việc giải mã file

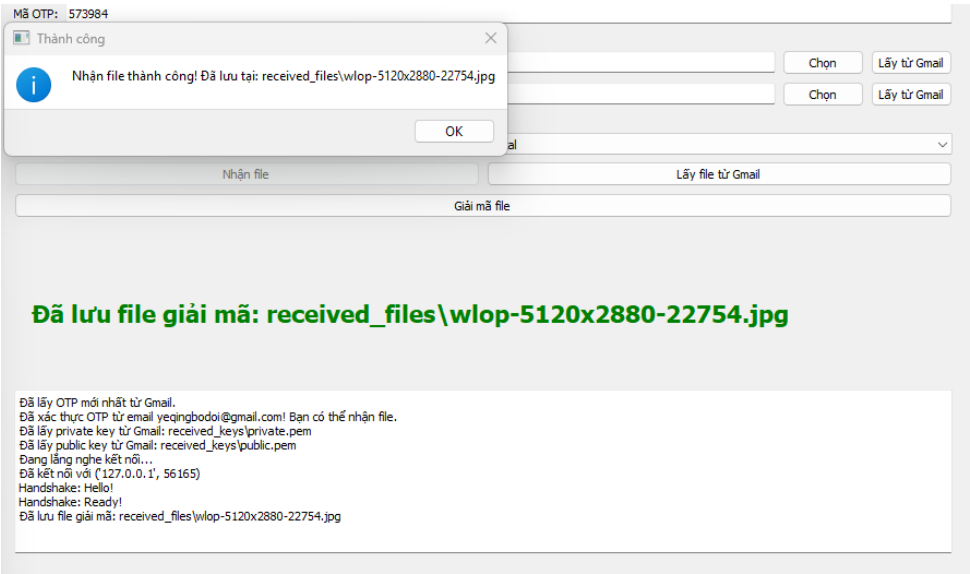
sau này.




Hình 3.6: Receiver nhận khóa

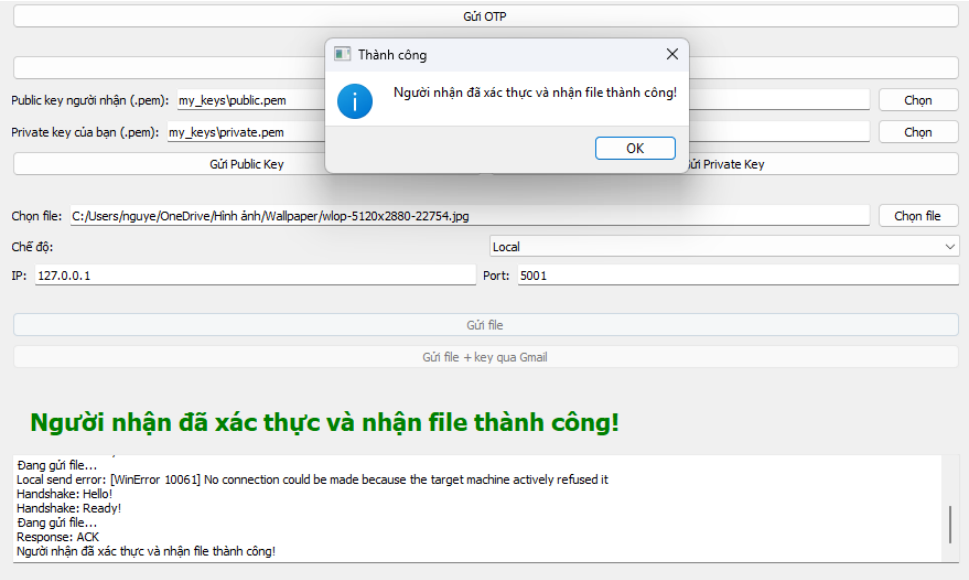
Bước 3:

- Đên bước nhận file lúc này bên Receiver sẽ là người lắng nghe kết nối từ Sender nên cần thực hiện trước.




Hình 3.7: Receiver nhận file

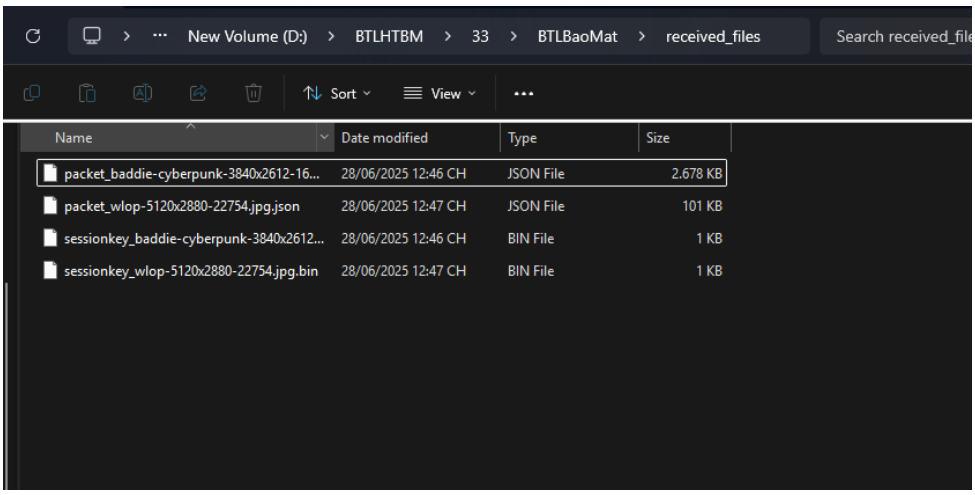
- Lúc này bên phía Sender sẽ chọn file và chọn phương thức gửi cũng như nhập địa chỉ IP và Port của Receiver và gửi file.




Hình 3.8: Sender chọn và gửi file

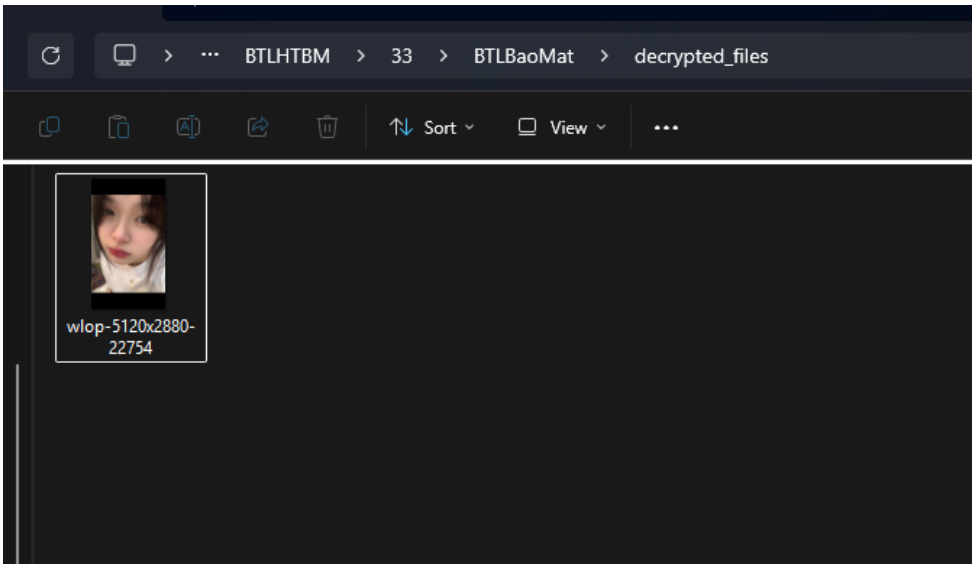
Bước 4:

- Sau khi đã có file Receiver sẽ dùng chức năng " Giải mã File " và được đưa đến folder received-files để chọn file đã được nhận trước đó ("Cần lưu ý : phải chọn đủ 2 file package và sessionkey")




Hình 3.9: Chọn file để giải mã

- Kết quả đạt được sau khi giải mã sẽ được lưu trong decrypted-files




Hình 3.10: Kết quả đạt được

## 3.2 Phân tích hiệu quả

Hệ thống chuyển file an toàn được phát triển nhằm giải quyết bài toán bảo mật trong quá trình truyền tải dữ liệu nhạy cảm – một trong những vấn đề sống còn đối với các tổ chức hoạt động trong môi trường số hiện đại. Bằng cách tích hợp các thuật toán mật mã học tiên tiến cùng cơ chế xác thực hai yếu tố (2FA), hệ thống không chỉ cung cấp một kênh truyền tải dữ liệu đáng tin cậy, mà còn đảm bảo ba thuộc tính cốt lõi của an ninh thông tin là: tính bảo mật (confidentiality), toàn vẹn (integrity) và khả dụng (availability).

Về hiệu quả bảo mật, việc ứng dụng các thuật toán mã hóa hiện đại như AES-256, RSA hoặc Elliptic Curve Cryptography (ECC) cho phép mã hóa nội dung tệp trước khi truyền đi, đảm bảo dữ liệu không thể bị đọc hoặc can thiệp trái phép trong suốt quá trình chuyển đổi. Khi được kết hợp với xác thực hai yếu tố – như mã OTP gửi về email hoặc điện thoại – hệ thống tăng cường đáng kể mức độ xác minh người dùng, từ đó ngăn chặn các hành vi truy cập trái phép, kể cả khi thông tin tài khoản bị lộ. Điều này đặc biệt hiệu quả trong các môi trường có yêu cầu cao về bảo mật như tổ chức tài chính, y tế, pháp luật, hoặc chính phủ.

Về hiệu quả trong đảm bảo tính toàn vẹn dữ liệu, hệ thống sử dụng các kỹ thuật kiểm tra tính toàn vẹn như hàm băm SHA-256 hoặc chữ ký số để đảm bảo rằng file không bị sửa đổi trong quá trình truyền tải. Mỗi lần tải lên hoặc tải xuống đều được ghi lại và xác minh thông qua mã định danh duy nhất, giúp người gửi và người nhận hoàn toàn yên tâm rằng nội dung không bị giả mạo hay thay đổi.

Đối với khả năng phục vụ và tính khả dụng, hệ thống được thiết kế theo hướng phân tán và có khả năng mở rộng linh hoạt trên hạ tầng điện toán đám mây. Nhờ đó, người dùng có thể chia

sẻ tài liệu mọi lúc, mọi nơi, với tốc độ truyền tải ổn định, kể cả trong điều kiện mạng thay đổi. Ngoài ra, hệ thống còn có khả năng tự động thu hồi quyền truy cập sau thời gian nhất định, gửi thông báo khi có truy cập bất thường, và hỗ trợ giao diện đơn giản để dễ dàng sử dụng với cả người dùng không chuyên.

Xét tổng thể, hệ thống đã đạt được hiệu quả cao trong việc giải quyết các mối đe dọa bảo mật phổ biến như nghe lén, giả mạo danh tính, tấn công trung gian (MITM), rò rỉ thông tin nhạy cảm,... mà vẫn đảm bảo sự tiện lợi, tốc độ và trải nghiệm người dùng. Từ đó, nó trở thành một giải pháp toàn diện, thích hợp để áp dụng rộng rãi trong doanh nghiệp, tổ chức giáo dục, cơ quan nhà nước hoặc các hệ thống giao tiếp số đòi hỏi mức độ bảo mật cao.

### 3.2.1 Phân tích và nhận xét đặc điểm của các thuật toán sử dụng

Giới thiệu tổng quan về các thuật toán sử dụng: Trong hệ thống chuyển file an toàn, các thuật toán mã hóa được triển khai nhằm đảm bảo ba mục tiêu bảo mật cốt lõi là tính bảo mật, tính toàn vẹn và xác thực. Cụ thể, hệ thống kết hợp giữa mã hóa đối xứng AES để bảo vệ nội dung file, mã hóa bất đối xứng RSA để bảo vệ khóa phiên (session key) và ký số, cùng với thuật toán băm SHA-512 để xác minh tính toàn vẹn dữ liệu. Việc phối hợp nhiều lớp mã hóa giúp hệ thống đạt được độ an toàn cao khi truyền tải dữ liệu nhạy cảm qua các kênh như internet, mạng nội bộ hoặc email, đồng thời hỗ trợ xác thực danh tính và thời gian truy cập giới hạn.

Phân tích thuật toán AES – Mã hóa nội dung: Thuật toán AES (Advanced Encryption Standard) là thành phần cốt lõi dùng để mã hóa nội dung của file cần gửi. Trong hệ thống, AES được sử dụng ở chế độ CBC (Cipher Block Chaining), kết hợp với vector khởi tạo ngẫu nhiên (IV) và kỹ thuật padding PKCS7 để đảm bảo mọi dữ liệu đều được mã hóa chính xác. AES là một thuật toán đối xứng có hiệu năng cao, phù hợp để mã hóa dữ liệu dung lượng lớn với tốc độ nhanh và ổn định. Tuy nhiên, vì là mã hóa đối xứng nên AES yêu cầu phải truyền khóa bí mật một cách an toàn đến người nhận, và để giải quyết vấn đề này, hệ thống đã sử dụng RSA để mã hóa khóa AES.

Phân tích thuật toán RSA – Mã hóa khóa và xác thực dữ liệu: RSA là một thuật toán mã hóa bất đối xứng rất phổ biến, được ứng dụng trong hệ thống để bảo vệ khóa AES (session key) và thực hiện chữ ký số. Với RSA, khóa công khai của người nhận sẽ được dùng để mã hóa khóa AES trước khi gửi, đảm bảo chỉ người nhận có khóa riêng mới có thể giải mã và truy cập nội dung thực. Ngoài ra, RSA cũng được sử dụng để ký số các thông tin quan trọng như tên tệp và thời gian hết hạn, giúp người nhận xác minh danh tính người gửi và kiểm tra tính xác thực của gói tin. Dù RSA có tốc độ chậm hơn AES, nhưng nhờ giới hạn vai trò chỉ trong việc mã hóa khóa và xác thực, hệ thống vẫn duy trì được hiệu suất cao.

Phân tích thuật toán SHA-512 – Kiểm tra toàn vẹn dữ liệu: Để đảm bảo dữ liệu không bị thay đổi trong quá trình truyền, hệ thống sử dụng thuật toán băm SHA-512 để tạo mã băm cho

từng gói tin. SHA-512 là một hàm băm một chiều, tạo ra chuỗi 512-bit duy nhất cho mỗi nội dung đầu vào. Trong quá trình tạo gói tin, các thành phần như IV, ciphertext và thời gian hết hạn sẽ được băm lại, và giá trị này sẽ được lưu kèm để người nhận kiểm tra lại sau. Nếu dữ liệu bị chỉnh sửa, giá trị băm tính lại sẽ khác với giá trị gốc, và hệ thống sẽ từ chối gói tin. Việc sử dụng SHA-512 giúp đảm bảo tính toàn vẹn dữ liệu và phát hiện sớm mọi hành vi thay đổi trái phép.

Đánh giá tổng thể và khả năng nâng cấp: Việc kết hợp AES, RSA và SHA-512 trong cùng một quy trình truyền tải dữ liệu giúp hệ thống đạt được mức bảo mật cao và toàn diện. Mỗi thuật toán đảm nhiệm một vai trò cụ thể, bổ trợ lẫn nhau để đảm bảo dữ liệu được bảo vệ trước khi gửi, trong quá trình truyền và sau khi nhận. Tuy nhiên, để nâng cao hơn nữa hiệu suất và khả năng tương thích, hệ thống trong tương lai có thể mở rộng sử dụng thuật toán mã hóa khóa công khai ECC (Elliptic Curve Cryptography) thay thế RSA nhằm giảm kích thước khóa và tăng tốc độ xử lý. Bên cạnh đó, việc tích hợp thêm xác thực hai lớp (2FA) và kiểm soát quyền truy cập theo IP hoặc thời gian thực sẽ giúp hệ thống linh hoạt, mạnh mẽ và phù hợp hơn với yêu cầu bảo mật hiện đại.

### 3.2.2 Đề xuất cải tiến

Mặc dù hệ thống chuyển file an toàn đã hoàn thiện cơ bản về chức năng bảo mật và truyền tải dữ liệu, nhưng để đáp ứng tốt hơn nhu cầu thực tế, khả năng triển khai trong môi trường doanh nghiệp cũng như thích ứng với các xu hướng bảo mật hiện đại, một số đề xuất cải tiến quan trọng có thể được triển khai trong các phiên bản tiếp theo.

Thứ nhất, cải tiến về thuật toán mã hóa: Hiện tại, hệ thống sử dụng RSA với độ dài khóa 2048 bit để mã hóa khóa phiên AES và tạo chữ ký số. Tuy nhiên, thuật toán RSA có tốc độ xử lý chậm và kích thước khóa lớn, gây ảnh hưởng đến hiệu suất, đặc biệt khi cần xử lý nhiều yêu cầu đồng thời hoặc chạy trên thiết bị hạn chế tài nguyên. Do đó, đề xuất thay thế RSA bằng ECC (Elliptic Curve Cryptography) là hướng đi hiệu quả. ECC với khóa 256 bit có thể cung cấp mức bảo mật tương đương RSA 3072 bit, nhưng sử dụng ít tài nguyên hơn, nhanh hơn và có khả năng tích hợp tốt với các nền tảng di động. Đây là thuật toán được khuyến nghị bởi nhiều tổ chức bảo mật quốc tế và phù hợp với tiêu chuẩn bảo mật hiện đại.

Thứ hai, nâng cao cơ chế xác thực người dùng: Hiện hệ thống chỉ sử dụng mã OTP qua email để xác thực người gửi và người nhận, điều này vẫn tiềm ẩn nguy cơ nếu email bị đánh cắp hoặc chuyển tiếp. Việc bổ sung xác thực hai yếu tố (2FA) mạnh hơn, chẳng hạn như xác thực qua ứng dụng (Google Authenticator, Authy) hoặc xác thực sinh trắc học trên thiết bị, sẽ nâng cao mức độ an toàn. Ngoài ra, hệ thống cũng nên cho phép cấu hình giới hạn truy cập theo IP, thiết bị hoặc thời gian cụ thể, ví dụ: chỉ cho phép truy cập một file từ địa chỉ IP cố định trong vòng 2 giờ sau khi gửi.

Thứ ba, mở rộng khả năng mã hóa đầu cuối và kiểm soát quyền truy cập nâng cao: Trong kiến trúc hiện tại, dữ liệu được mã hóa bằng AES trước khi gửi, tuy nhiên nếu tích hợp mã hóa đầu cuối (End-to-End Encryption) từ gốc đến đích (trên cả kênh email và lưu trữ đám mây), sẽ tăng thêm một lớp bảo vệ mạnh mẽ. Đồng thời, hệ thống cũng nên tích hợp các chính sách kiểm soát quyền truy cập (access control) chi tiết hơn: cho phép hoặc không cho phép tải xuống, giới hạn số lần xem file, đặt mật khẩu mở file hoặc khóa file khi phát hiện truy cập bất thường.

Thứ tư, cải tiến giao diện người dùng (UI/UX) và trải nghiệm sử dụng: Giao diện hiện tại thiên về kỹ thuật, phù hợp với người dùng chuyên môn. Tuy nhiên, để mở rộng đối tượng sử dụng, hệ thống cần được cải tiến để trở nên trực quan, dễ dùng và thân thiện hơn. Các tính năng có thể bổ sung bao gồm: hỗ trợ kéo–thả file, thanh tiến trình upload mã hóa thời gian thực, thông báo xác nhận đã nhận file, hiển thị trạng thái mã hóa, xác thực bằng biểu tượng màu, và cảnh báo nếu thao tác sai hoặc thiếu thông tin. Ngoài ra, dashboard theo dõi lịch sử gửi – nhận file, trạng thái bảo mật, thời gian hết hạn sẽ giúp người dùng quản lý toàn bộ quá trình chuyển file dễ dàng hơn.

Thứ năm, tích hợp khả năng giám sát, thống kê và tương thích hệ thống: Đối với môi trường doanh nghiệp hoặc tổ chức lớn, việc có hệ thống log và giám sát (audit logging) là cần thiết để phục vụ kiểm toán nội bộ và xử lý sự cố bảo mật. Hệ thống nên ghi nhận rõ ai đã gửi file gì, lúc nào, gửi cho ai, ai đã truy cập và hành vi truy cập (đúng thời gian, sai thời gian, thất bại,...). Đồng thời, có thể tích hợp hệ thống cảnh báo (alert) nếu phát hiện truy cập trái phép hoặc hành vi bất thường. Cuối cùng, để nâng cao khả năng triển khai thực tế, hệ thống nên hỗ trợ tích hợp API để kết nối với các nền tảng quản trị nội bộ như ERP, email server doanh nghiệp, Google Workspace hoặc Microsoft 365, từ đó mở rộng khả năng ứng dụng vào thực tế.

Dưới đây là code dự án của chúng tôi:

https://github.com/ngyetdan/NhapMonAnToanBaoMatThongTin.git

# Tài liệu tham khảo


1. Lê Thị Thùy Trang. Bài giảng môn Nhập môn An toàn, Bảo mật Thông tin. Trường Đại học Đại Nam, 2025.
2. William Stallings. Cryptography and Network Security: Principles and Practice. 8th Edi tion, Pearson, 2020.
3. ChatGPT, OpenAI. Hỗ trợ phân tích, trình bày lý thuyết và viết báo cáo chuyên đề bảo mật. Truy cập tại: https://chat.openai.com
4. Google Search. Tìm kiếm tài liệu tham khảo liên quan đến thuật toán mã hóa DES, RSA và SHA-256. Truy cập tại: https://www.google.com
5. MDNWebDocs. Introduction to Cryptography APIs. Truy cập tại: https://developer.mozilla.org
6. GeeksforGeeks. DataEncryption Standard (DES),RSAAlgorithm,SHA-256HashFunction– Concepts and Implementation. Truy cập tại: https://www.geeksforgeeks.org
7. Python Software Foundation. Documentation for Cryptography and hashlib libraries. Truy cập tại: https://docs.python.org