BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC ĐẠI NAM

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO BÀI TẬP LỚN

Gửi báo cáo công ty qua Server trung gian

Sinh viên thực hiện : Đỗ Tiến Đại
Đào Đình Chí
Lê Thị Lý

Ngành : Công nghệ thông tin

Giảng viên hướng dẫn : ThS. Lê Thị Thùy Trang

Lời cảm ơn

Trước tiên, em xin gửi lời cảm ơn chân thành và sâu sắc nhất đến Thạc sĩ Lê Thị Thùy Trang – giảng viên phụ trách môn Nhập môn An toàn, Bảo mật Thông tin. Trong suốt quá trình học tập và thực hiện bài tập lớn, cô không chỉ tận tình giảng dạy mà còn truyền cảm hứng, hướng dẫn em từng bước tiếp cận lĩnh vực an toàn thông tin – một lĩnh vực quan trọng và đầy tính ứng dụng trong thời đại số.

Nhờ phương pháp giảng dạy rõ ràng, khoa học và gắn liền với thực tiễn, cô đã giúp em nắm vững các kiến thức nền tảng về các thuật toán mã hóa như AES-GCM, RSA, và SHA-512. Những bài giảng sinh động, dễ hiểu cùng sự khuyến khích nghiên cứu và thực hành đã giúp em phát triển tư duy logic, tư duy phản biện, và khả năng áp dụng lý thuyết vào các dự án thực tế. Đặc biệt, cô đã tạo nên một môi trường học tập tích cực, nơi em không chỉ học kiến thức chuyên môn mà còn rèn luyện tinh thần trách nhiệm và sự cẩn trọng trong công việc liên quan đến bảo mật thông tin.

Trong quá trình thực hiện đề tài "Gửi báo cáo công ty qua Server trung gian" – một hệ thống truyền file an toàn sử dụng Python và giao diện `tkinter`, em đã áp dụng thành công các kiến thức đã học để xây dựng một hệ thống bảo mật với các thành phần Sender, Server, và Receiver. Thông qua việc triển khai các thuật toán AES-GCM (mã hóa file `report.txt`), RSA 1024-bit (ký số metadata và mã hóa khóa phiên), và SHA-512 (kiểm tra toàn vẹn), cùng với giao diện `tkinter` trong `sender_gui.py`, `server_gui.py`, `receiver_gui.py` (sử dụng font `Segoe UI`, `Consolas`, màu nền `#e8f0fe`, log `ScrolledText`), em đã hiểu sâu hơn về cơ chế bảo mật dữ liệu trong các ứng dụng thực tế. Logic xử lý trong `sender_core.py`, `server_core.py`, `receiver_core.py`, và các hàm mã hóa trong `crypto_utils.py` đã giúp em rèn luyện kỹ năng lập trình Python, quản lý socket, và xử lý giao thức truyền dữ liệu.

Em vô cùng biết ơn cô Th.S Lê Thị Thùy Trang vì sự hỗ trợ nhiệt tình, những góp ý chuyên môn quý giá, và sự đồng hành trong suốt quá trình học tập. Sự tận tâm và nghiêm túc của cô đã truyền động lực để em nỗ lực hoàn thiện đề tài một cách tốt nhất. Em cũng xin chân thành cảm ơn khoa và nhà trường đã tổ chức học phần Nhập môn An toàn, Bảo mật Thông tin – một môn học thiết thực, bổ ích, tạo cơ hội cho sinh viên tiếp cận với lĩnh vực an toàn thông tin, vốn là yếu tố cốt lõi trong sự phát triển của công nghệ thông tin hiện đại.

Mặc dù đề tài "Gửi báo cáo công ty qua Server trung gian" vẫn còn một số hạn chế, như quản lý file đầu ra cố định hoặc thiếu cơ chế thử lại khi kết nối thất bại, em đã cố gắng hoàn thiện trong phạm vi thời gian và kinh nghiệm của mình. Với tinh thần cầu tiến, em hy vọng sẽ tiếp tục nhận được sự hướng dẫn và góp ý từ cô để nâng cao trình độ chuyên môn và kỹ năng ứng dụng trong các dự án sắp tới.

Một lần nữa, em xin gửi lời cảm ơn chân thành nhất đến cô Thạc sĩ Lê Thị Thùy Trang. Kính chúc cô thật nhiều sức khỏe, hạnh phúc, và tiếp tục thành công trên con đường giảng dạy và nghiên cứu.

Tóm tắt

Trong bối cảnh chuyển đổi số và nhu cầu bảo mật dữ liệu ngày càng tăng, việc truyền tải các tài liệu nhạy cảm như báo cáo công ty đòi hỏi các giải pháp an toàn để ngăn chặn các mối đe dọa như nghe lén, giả mạo, hay sửa đổi dữ liệu. Đề tài này hướng đến xây dựng một hệ thống truyền file an toàn có tên "Gửi báo cáo công ty qua Server trung gian", nhằm cung cấp một nền tảng thực tiễn để truyền file `report.txt` từ Sender đến Receiver qua Server trung gian, sử dụng các thuật toán mã hóa hiện đại: AES-GCM, RSA 1024-bit, và SHA-512.

Hệ thống được triển khai bằng Python, sử dụng giao diện `tkinter` và thư viện `pycryptodome`, với các thành phần chính: Sender (mã hóa và gửi file), Server trung gian (chuyển tiếp và ghi log), và Receiver (xác minh và giải mã). Các thuật toán được tích hợp như sau:

- **AES-GCM**: Mã hóa nội dung file `report.txt` với khóa phiên 256-bit, tạo `nonce`, `ciphertext`, và `tag` để đảm bảo bí mật và xác thực, được triển khai trong `aes_encrypt` và `aes_decrypt` (`crypto_utils.py`).
- **RSA 1024-bit**: Mã hóa khóa phiên bằng OAEP và ký số metadata (tên file, timestamp, ID) bằng PSS, sử dụng SHA-512, được triển khai trong `rsa_encrypt`, `rsa_sign`, `rsa_decrypt`, `rsa_verify` (`crypto_utils.py`).
- **SHA-512**: Kiểm tra tính toàn vẹn của dữ liệu bằng hash trên `nonce + ciphertext + tag + timestamp`, được triển khai trong `sha512_hash` (`crypto_utils.py`).

Giao diện người dùng, được xây dựng trong `sender_gui.py`, `server_gui.py`, và `receiver_gui.py`, sử dụng `tkinter` với màu nền `#e8f0fe`, font `Segoe UI` cho tiêu đề, `Consolas` cho log, và khu vực `ScrolledText` (nền `#1e1e1e`, chữ `#00ff00`) để hiển thị trạng thái thời gian thực. Logic hệ thống, được triển khai trong `sender_core.py`, `server_core.py`, và `receiver_core.py`, hỗ trợ handshake (`Hello!`, `Ready!`), truyền gói tin JSON (buffer 65536 bytes), xác minh chữ ký, hash, tag, và lưu file. Server trung gian ghi log giao dịch vào `server_log.txt` với định dạng thời gian `[Sun Jun 29 22:10:00 2025]`.

Hệ thống không chỉ giúp người dùng hiểu rõ cách áp dụng các thuật toán mã hóa trong kịch bản doanh nghiệp mà còn rèn luyện kỹ năng lập trình Python, quản lý socket, và thiết kế giao diện. Tuy vẫn còn hạn chế như buffer cố định hoặc thiếu cơ chế thử lại, hệ thống là một công cụ thực tiễn hiệu quả, mở ra tiềm năng ứng dụng trong các môi trường doanh nghiệp hoặc giáo dục về an toàn thông tin.

Mục lục

Danh sách bảng

Chương 1

Giới thiệu đề tài

1.1 Đặt vấn đề

Trong thời đại công nghệ 4.0, thông tin trở thành tài sản quan trọng nhưng cũng dễ bị tổn thương trước các mối đe dọa mạng. Theo thống kê từ Kaspersky, năm 2023 ghi nhận hơn 1,5 tỷ vụ tấn công mạng trên toàn cầu, trong đó Việt Nam nằm trong top 10 quốc gia chịu nhiều nguy cơ nhất, với hàng triệu vụ tấn công nhắm vào dữ liệu doanh nghiệp và cá nhân. Các mối đe dọa như nghe lén, giả mạo danh tính, hay đánh cắp dữ liệu không chỉ gây thiệt hại kinh tế mà còn đe dọa an ninh quốc gia. Điều này nhấn mạnh vai trò của bảo mật thông tin trong việc đảm bảo an toàn dữ liệu trong môi trường số.

Hệ thống truyền file an toàn qua server trung gian được phát triển để đáp ứng nhu cầu bảo vệ dữ liệu trong quá trình truyền tải. Đề tài tập trung vào việc gửi báo cáo công ty (file `report.txt`) qua một server trung gian đến đối tác, sử dụng các thuật toán mã hóa hiện đại như AES-GCM để mã hóa nội dung, RSA 1024-bit để ký số và trao đổi khóa, cùng với SHA-512 để kiểm tra tính toàn vẹn. Quy trình này không chỉ đảm bảo tính bí mật mà còn xác thực nguồn gốc và bảo vệ dữ liệu khỏi bị sửa đổi. Tuy nhiên, việc triển khai một hệ thống như vậy đòi hỏi sự chính xác trong xử lý mã hóa, ký số, và truyền dữ liệu, đồng thời cần giao diện thân thiện để người dùng dễ dàng thao tác.

Đề tài "Gửi báo cáo công ty qua Server trung gian" ra đời nhằm cung cấp một giải pháp thực tiễn, tích hợp các thuật toán bảo mật vào một hệ thống truyền file an toàn. Hệ thống bao gồm ba thành phần: Sender (người gửi), Server trung gian, và Receiver (người nhận), được phát triển bằng Python với giao diện đồ họa sử dụng `tkinter`. Mục tiêu là đảm bảo file báo cáo được truyền an toàn, với log thời gian giao dịch được lưu trữ tại server trung gian, đồng thời cung cấp trải nghiệm người dùng trực quan.

1.2 Bối cảnh đề tài

Sự phát triển của công nghệ số mang lại nhiều lợi ích nhưng cũng làm gia tăng các rủi ro bảo mật. Theo báo cáo của Cục An toàn thông tin (Bộ Thông tin và Truyền thông) năm 2023, Việt Nam ghi nhận hơn 13.000 vụ tấn công mạng nghiêm trọng, tăng 15% so với năm trước, với các hình thức như phishing, ransomware, và DDoS. Những con số này cho thấy sự cấp bách trong việc phát triển các giải pháp bảo mật dữ liệu, đặc biệt trong các giao dịch nhạy cảm như truyền báo cáo công ty.

Trong môi trường doanh nghiệp, việc gửi báo cáo giữa các phòng ban hoặc đối tác thường xuyên diễn ra. Tuy nhiên, nếu không được bảo vệ, dữ liệu có thể bị chặn hoặc sửa đổi, dẫn đến rủi ro về bảo mật và uy tín. Các thuật toán như AES-GCM và RSA đã được ứng dụng rộng rãi trong các giao thức bảo mật như HTTPS hay VPN, nhưng việc tích hợp chúng vào một hệ thống truyền file độc lập vẫn là một thách thức. Đề tài này khắc phục vấn đề bằng cách xây dựng một hệ thống hoàn chỉnh, nơi Sender mã hóa file, Server trung gian chuyển tiếp dữ liệu và lưu log, còn Receiver xác minh và giải mã. Giao diện `tkinter` giúp người dùng không chuyên dễ dàng sử dụng hệ thống, từ việc chọn file đến theo dõi quá trình truyền tải qua log.

Xu hướng phát triển các ứng dụng bảo mật thực tiễn đang được chú trọng tại Việt Nam, đặc biệt trong bối cảnh chuyển đổi số. Hệ thống này không chỉ đáp ứng nhu cầu bảo vệ dữ liệu mà còn có tiềm năng mở rộng cho các ứng dụng như truyền file nội bộ trong doanh nghiệp hoặc chia sẻ tài liệu giữa các tổ chức.

1.3 Lý do chọn đề tài

Đề tài "Gửi báo cáo công ty qua Server trung gian" được lựa chọn dựa trên các lý do sau:

- **Tính thực tiễn:** Hệ thống cung cấp giải pháp truyền file an toàn, đáp ứng nhu cầu thực tế trong doanh nghiệp khi gửi báo cáo hoặc tài liệu nhạy cảm.
- **Ứng dụng bảo mật:** Việc tích hợp AES-GCM, RSA 1024-bit, và SHA-512 giúp người dùng hiểu và áp dụng các thuật toán mã hóa hiện đại vào thực tiễn.
- **Giao diện thân thiện:** Giao diện đồ họa `tkinter` giúp đơn giản hóa thao tác, phù hợp với cả người dùng không chuyên về kỹ thuật.
- **Ghi log giao dịch:** Server trung gian lưu log thời gian, hỗ trợ theo dõi và kiểm tra lịch sử truyền tải, đáp ứng yêu cầu quản lý giao dịch.
- **Tiềm năng mở rộng:** Hệ thống có thể được phát triển thêm để hỗ trợ truyền nhiều file, nén dữ liệu, hoặc tích hợp các giao thức bảo mật khác.

1.4 Mục tiêu đề tài

Mục tiêu tổng thể là xây dựng một hệ thống truyền báo cáo công ty an toàn qua server trung gian, đảm bảo tính bí mật, xác thực, và toàn vẹn dữ liệu. Các mục tiêu cụ thể bao gồm:

- **Đảm bảo bảo mật:** Sử dụng AES-GCM để mã hóa file, RSA 1024-bit để ký số và trao đổi khóa, SHA-512 để kiểm tra toàn vẹn.
- **Xây dựng quy trình truyền tải:** Thực hiện handshake, xác thực, mã hóa, và giải mã đúng theo luồng xử lý được định nghĩa.
- **Ghi log giao dịch:** Server trung gian lưu trữ log thời gian các giao dịch vào file và hiển thị trên giao diện GUI.
- **Cung cấp giao diện thân thiện:** Phát triển GUI bằng `tkinter` cho Sender, Server, và Receiver, giúp người dùng dễ dàng thao tác và theo dõi.
- **Đánh giá hiệu quả:** Đảm bảo hệ thống hoạt động ổn định, xử lý đúng các trường hợp lỗi (như chữ ký hoặc hash không hợp lệ) và gửi phản hồi phù hợp.

1.5 Phạm vi thực hiện

Để đảm bảo tính khả thi, đề tài được giới hạn như sau:

- **Mô hình hệ thống:** Bao gồm ba thành phần: Sender (gửi file), Server trung gian (chuyển tiếp và lưu log), Receiver (nhận và giải mã).
- **Công cụ phát triển:** Sử dụng Python với thư viện `pycryptodome` cho AES-GCM và RSA, `tkinter` cho giao diện đồ họa, và `socket` cho truyền dữ liệu.
- **Giao diện người dùng:** GUI đơn giản với các nút chọn file, xóa log, và khu vực hiển thị log giao dịch.
- **Giới hạn tính năng:** Chỉ hỗ trợ gửi một file `report.txt` mỗi lần, không hỗ trợ truyền nhiều file hoặc nén dữ liệu.
- **Môi trường triển khai:** Hệ thống chạy trên mạng nội bộ với các địa chỉ IP cố định (`172.20.10.2` cho Server, `172.20.10.4` cho Receiver).

1.6 Phương pháp nghiên cứu

Quá trình phát triển hệ thống được thực hiện qua các bước:

1. Tìm hiểu lý thuyết

- Nghiên cứu các thuật toán mã hóa: AES-GCM (đối xứng), RSA 1024-bit (bất đối xứng), SHA-512 (hash).
- Tìm hiểu giao thức truyền dữ liệu qua socket và tích hợp GUI với `tkinter`.

2. Phân tích yêu cầu

- Xác định luồng xử lý: handshake (`Hello!` và `Ready!`), mã hóa, ký số, kiểm tra toàn vẹn, và lưu log.
- Thiết kế cấu trúc gói tin JSON chứa `nonce`, `ciphertext`, `tag`, `hash`, `signature`, `metadata`, và `encrypted_session_key`.

3. Lập trình và thử nghiệm

- Phát triển mã nguồn Python, chia thành các module: `sender_core.py`, `receiver_core.py`, `server_core.py`, `crypto_utils.py`, `sender_gui.py`, `receiver_gui.py`, `server_gui.py`.
- Thử nghiệm với các trường hợp: gửi file hợp lệ, chữ ký không hợp lệ, hash không khớp, để đảm bảo hệ thống xử lý đúng.

4. Đánh giá và hoàn thiện

- Kiểm tra hiệu quả truyền file trên mạng nội bộ, đảm bảo log thời gian được lưu chính xác.
- Tối ưu hóa giao diện GUI và xử lý lỗi để tăng trải nghiệm người dùng.

1.7 Cấu trúc của báo cáo

Báo cáo được tổ chức thành ba chương:

- **Chương 1: Giới thiệu đề tài**
  Trình bày bối cảnh, lý do, mục tiêu, phạm vi, và phương pháp thực hiện, nhấn mạnh tầm quan trọng của bảo mật trong truyền file.
- **Chương 2: Tổng quan cơ sở lý thuyết**
  Cung cấp kiến thức về AES-GCM, RSA, SHA-512, và cách chúng được ứng dụng trong hệ thống.
- **Chương 3: Triển khai hệ thống**
  Mô tả chi tiết quá trình phát triển, từ thiết kế luồng xử lý, lập trình, thử nghiệm, đến tối ưu

hóa.

Chương 2

Tổng quan cơ sở lý thuyết

2.1 Bảo mật thông tin và ứng dụng trong truyền file an toàn

2.1.1 Khái niệm bảo mật thông tin

Bảo mật thông tin là lĩnh vực tập trung vào việc bảo vệ dữ liệu khỏi các mối đe dọa như truy cập trái phép, đánh cắp, giả mạo hoặc sửa đổi. Trong bối cảnh công nghệ số, các hệ thống như giao dịch ngân hàng, trao đổi tài liệu doanh nghiệp, và lưu trữ đám mây ngày càng phổ biến, dẫn đến sự gia tăng các cuộc tấn công mạng. Theo báo cáo của Trung tâm Giám sát an toàn không gian mạng quốc gia Việt Nam, năm 2024 ghi nhận hơn 15.000 vụ tấn công mạng tại Việt Nam, bao gồm phishing, ransomware, và DDoS, gây thiệt hại lớn về kinh tế và đe dọa an ninh. Các thuật toán mã hóa như AES-GCM, RSA, và SHA-512 đóng vai trò cốt lõi trong việc bảo vệ dữ liệu, đảm bảo tính bí mật, toàn vẹn, xác thực, và không thể phủ nhận.

Các mục tiêu chính của bảo mật thông tin bao gồm:

- **Tính bí mật:** Chỉ người nhận được ủy quyền (có khóa hợp lệ) mới có thể truy cập nội dung dữ liệu, ví dụ, file báo cáo `report.txt` được mã hóa bằng AES-GCM chỉ có thể giải mã với khóa phiên đúng.
- **Tính toàn vẹn:** Đảm bảo dữ liệu không bị thay đổi trong quá trình truyền, được kiểm tra bằng hash SHA-512 trên dữ liệu mã hóa.
- **Tính xác thực:** Xác minh danh tính người gửi, đạt được qua chữ ký RSA trên metadata của file.
- **Tính không thể phủ nhận:** Người gửi không thể chối bỏ việc gửi dữ liệu, được đảm bảo thông qua chữ ký RSA sử dụng khóa riêng.

Trong hệ thống truyền báo cáo công ty qua server trung gian, các nguyên tắc này được áp

dụng để bảo vệ file `report.txt` trong quá trình truyền từ Sender đến Receiver. Sender mã hóa file bằng AES-GCM, ký số metadata (tên file, timestamp, ID giao dịch) bằng RSA, và sử dụng SHA-512 để kiểm tra toàn vẹn. Receiver xác minh chữ ký, hash, và giải mã dữ liệu, đảm bảo tính an toàn và xác thực của báo cáo.

2.1.2 Ứng dụng trong truyền file an toàn

Hệ thống truyền file an toàn qua server trung gian được thiết kế để đáp ứng nhu cầu bảo mật trong các giao dịch doanh nghiệp, chẳng hạn như gửi báo cáo công ty. Thay vì truyền trực tiếp, dữ liệu được chuyển qua một server trung gian, giúp tách biệt Sender và Receiver, đồng thời ghi log thời gian giao dịch để theo dõi. Các thuật toán mã hóa được tích hợp như sau:

- **AES-GCM:** Mã hóa nội dung file để đảm bảo tính bí mật và xác thực thông qua tag.
- **RSA 1024-bit:** Sử dụng để mã hóa khóa phiên (OAEP) và ký số metadata (PSS), đảm bảo xác thực và không thể phủ nhận.
- **SHA-512:** Kiểm tra tính toàn vẹn của dữ liệu bằng cách tính hash trên nonce, ciphertext, tag, và timestamp.

Hệ thống sử dụng giao thức socket để truyền dữ liệu qua mạng nội bộ, với giao diện đồ họa `tkinter` giúp người dùng dễ dàng thao tác và theo dõi log. Server trung gian chỉ chuyển tiếp dữ liệu mà không tham gia mã hóa hay giải mã, đảm bảo tính đơn giản và tập trung vào chức năng bảo mật ở Sender và Receiver.

2.2 Tổng quan về các thuật toán mã hóa sử dụng trong hệ thống

2.2.1 Mã hóa AES-GCM (Advanced Encryption Standard - Galois/Counter Mode)

Nguyên lý hoạt động: AES-GCM là thuật toán mã hóa đối xứng, sử dụng khóa 256-bit và kết hợp chế độ Counter (CTR) với Galois Message Authentication Code (GMAC) để cung cấp cả mã hóa và xác thực. AES-GCM mã hóa dữ liệu thành ciphertext và tạo tag xác thực, sử dụng nonce để đảm bảo tính duy nhất của mỗi phiên mã hóa.

Quá trình mã hóa:

1. Tạo khóa ngẫu nhiên 256-bit và nonce 96-bit.
2. Dữ liệu (plaintext) được mã hóa thành ciphertext bằng AES trong chế độ CTR.

3. Tính tag xác thực bằng GMAC trên ciphertext và dữ liệu bổ sung (nếu có).

Quá trình giải mã:

1. Dùng khóa, nonce, và tag để giải mã ciphertext.

2. Kiểm tra tag để đảm bảo dữ liệu không bị sửa đổi.

Ví dụ minh họa: Trong hệ thống, Sender mã hóa file `report.txt` bằng AES-GCM, tạo ra `nonce`, `ciphertext`, và `tag`. Các giá trị này được mã hóa Base64 và gửi trong gói tin JSON. Receiver sử dụng khóa phiên (được giải mã từ RSA) để giải mã và kiểm tra tag, đảm bảo tính bí mật và xác thực của file.

Đặc điểm:

- **Ưu điểm:** Bảo mật cao, hiệu suất nhanh, tích hợp xác thực qua tag, được sử dụng trong các giao thức như TLS/HTTPS.
- **Nhược điểm:** Yêu cầu quản lý khóa và nonce cẩn thận để tránh lặp lại nonce.

Ứng dụng trong hệ thống: Trong `sender_core.py`, hàm `aes_encrypt` tạo `nonce`, `ciphertext`, và `tag` cho file báo cáo. Trong `receiver_core.py`, hàm `aes_decrypt` kiểm tra tag và giải mã để khôi phục `report.txt`, mô phỏng bảo mật dữ liệu hiện đại trong doanh nghiệp.

2.2.2 Mã hóa RSA (Rivest–Shamir–Adleman)

Khái niệm: RSA là thuật toán mã hóa bất đối xứng, sử dụng cặp khóa công khai và riêng tư, dựa trên bài toán phân tích thừa số nguyên tố. Trong hệ thống, RSA 1024-bit được sử dụng để mã hóa khóa phiên AES (với OAEP) và ký số metadata (với PSS).

Quy trình tạo khóa:

1. Chọn hai số nguyên tố lớn p và q.

2. Tính $n = p \times q$, $\varphi(n) = (p-1)(q-1)$.

3. Chọn số công khai e sao cho $\gcd(e, \varphi(n)) = 1$.

4. Tính số riêng d sao cho $d \times e \equiv 1 \mod \varphi(n)$.

Khóa công khai: (e,n), khóa riêng: (d,n).

Mã hóa (OAEP): Dữ liệu m (khóa phiên AES) được mã hóa bằng:

$$c = \text{OAEP}(m, e, n)$$

OAEP sử dụng SHA-512 để tăng bảo mật.

Ký số (PSS): Metadata m được ký bằng:

$$s = \text{PSS}(m, d, n)$$

Sử dụng SHA-512 để tạo hash trước khi ký.

Giải mã và xác minh:

- Giải mã: $m = \text{OAEP}^{-1}(c, d, n)$.
- Xác minh chữ ký: Kiểm tra $\text{PSS}(m, s, e, n)$ với hash SHA-512.

Ví dụ minh họa: Sender mã hóa khóa phiên AES bằng khóa công khai của Receiver (`RECEIVER_PUBLIC_KEY`) và ký metadata (tên file, timestamp, ID) bằng khóa riêng (`SENDER_PRIVATE_KEY`). Receiver xác minh chữ ký bằng khóa công khai của Sender và giải mã khóa phiên bằng khóa riêng.

Đặc điểm:

- **Ưu điểm:** Không cần chia sẻ khóa bí mật, phù hợp cho xác thực và trao đổi khóa.
- **Nhược điểm:** Tốc độ chậm, chỉ phù hợp cho dữ liệu ngắn như khóa phiên.

Ứng dụng trong hệ thống: Trong `sender_core.py`, hàm `rsa_encrypt` và `rsa_sign` thực hiện mã hóa khóa phiên và ký số metadata. Trong `receiver_core.py`, `rsa_decrypt` và `rsa_verify` giải mã khóa và xác minh chữ ký, đảm bảo xác thực và không thể phủ nhận.

2.2.3 Hàm băm SHA-512

Khái niệm: SHA-512 là hàm băm một chiều, tạo ra giá trị băm 512-bit từ dữ liệu đầu vào, dùng để kiểm tra tính toàn vẹn. Nó không thể đảo ngược, đảm bảo bất kỳ thay đổi nào trong dữ liệu sẽ tạo ra giá trị băm khác.

Nguyên lý hoạt động:

1. Dữ liệu đầu vào được chia thành các khối 1024-bit, thêm padding nếu cần.
2. Qua 80 vòng xử lý với các phép toán logic, SHA-512 tạo ra giá trị băm cố định.

Ví dụ minh họa: Trong hệ thống, Sender tính hash SHA-512 trên `nonce + ciphertext + tag + timestamp`. Receiver tính lại hash và so sánh để kiểm tra toàn vẹn. Nếu hash không khớp, dữ liệu bị từ chối với thông báo `NACK`.

Đặc điểm:

- **Ưu điểm:** Bảo mật cao, khó tìm va chạm, được dùng trong TLS và blockchain.

- **Nhược điểm:** Chỉ kiểm tra toàn vẹn, không mã hóa hay xác thực.

Ứng dụng trong hệ thống: Trong `sender_core.py` và `receiver_core.py`, hàm `sha512_hash` tính giá trị băm trên dữ liệu mã hóa, đảm bảo file `report.txt` không bị sửa đổi trong quá trình truyền.

2.3 Ứng dụng các thuật toán vào hệ thống

Hệ thống truyền báo cáo công ty qua server trung gian tích hợp các thuật toán như sau:

- **Sender:**
  - Đọc file `report.txt`, mã hóa bằng AES-GCM (`aes_encrypt`) để tạo `nonce`, `ciphertext`, `tag`.
  - Tạo metadata (tên file, timestamp, ID giao dịch), ký số bằng RSA-PSS (`rsa_sign`).
  - Mã hóa khóa phiên AES bằng RSA-OAEP (`rsa_encrypt`).
  - Tính hash SHA-512 trên `nonce + ciphertext + tag + timestamp`.
  - Gửi gói tin JSON chứa `metadata`, `signature`, `encrypted_session_key`, `nonce`, `ciphertext`, `tag`, `hash` qua server trung gian.
- **Server trung gian:**
  - Chuyển tiếp handshake (`Hello!`, `Ready!`) và gói tin JSON mà không can thiệp vào nội dung.
  - Ghi log thời gian giao dịch vào `server_log.txt` và hiển thị trên GUI.
- **Receiver:**
  - Xác minh chữ ký RSA trên metadata (`rsa_verify`).
  - Kiểm tra hash SHA-512 để đảm bảo toàn vẹn.
  - Giải mã khóa phiên bằng RSA-OAEP (`rsa_decrypt`).
  - Giải mã file bằng AES-GCM (`aes_decrypt`), kiểm tra tag.
  - Lưu file vào `report.txt` và gửi `ACK` nếu hợp lệ, hoặc `NACK` nếu có lỗi.

Quy trình tổng thể: Hệ thống mô phỏng một kịch bản thực tế trong doanh nghiệp, nơi báo cáo nhạy cảm được truyền an toàn qua mạng nội bộ. Sender và Receiver sử dụng các thuật toán hiện đại (AES-GCM, RSA, SHA-512) để đảm bảo bảo mật, trong khi server trung gian đóng

vai trò trung lập, lưu log để theo dõi. Giao diện `tkinter` cung cấp trải nghiệm trực quan, giúp người dùng theo dõi quá trình truyền qua log thời gian thực.

Mô phỏng thực tế: Hệ thống tái hiện các giao thức bảo mật như TLS, nơi AES-GCM mã hóa dữ liệu, RSA xác thực và trao đổi khóa, SHA-512 kiểm tra toàn vẹn, giúp người dùng hiểu cách bảo vệ dữ liệu trong các ứng dụng thực tế như giao dịch ngân hàng hoặc truyền tài liệu doanh nghiệp.

Chương 3

Triển khai hệ thống truyền báo cáo công ty qua Server trung gian

3.1 Mô tả bài toán

Trong bối cảnh công nghệ số và chuyển đổi số ngày càng phát triển, việc truyền tải dữ liệu nhạy cảm như báo cáo công ty giữa các phòng ban hoặc đối tác đòi hỏi các biện pháp bảo mật mạnh mẽ để ngăn chặn các mối đe dọa như nghe lén, giả mạo, hay sửa đổi dữ liệu. Theo báo cáo của Cục An toàn thông tin (Bộ Thông tin và Truyền thông) năm 2024, Việt Nam ghi nhận hơn 15.000 vụ tấn công mạng, với các hình thức như phishing, ransomware, và DDoS, nhấn mạnh tầm quan trọng của việc bảo vệ dữ liệu trong quá trình truyền tải.

Hệ thống "Gửi báo cáo công ty qua Server trung gian" được thiết kế để giải quyết bài toán truyền file an toàn, cụ thể là file `report.txt`, từ Sender (người gửi) đến Receiver (người nhận) thông qua một server trung gian. Hệ thống sử dụng các thuật toán mã hóa hiện đại như AES-GCM để mã hóa nội dung, RSA 1024-bit để ký số và trao đổi khóa, và SHA-512 để kiểm tra tính toàn vẹn. Server trung gian đóng vai trò chuyển tiếp dữ liệu và ghi log thời gian giao dịch, đảm bảo tính minh bạch và khả năng theo dõi.

Bài toán yêu cầu xây dựng một hệ thống với các đặc điểm sau:

- **Bảo mật dữ liệu:** File `report.txt` được mã hóa bằng AES-GCM để đảm bảo tính bí mật và xác thực, sử dụng khóa phiên 256-bit. Khóa phiên được mã hóa bằng RSA 1024-bit với OAEP và SHA-512.
- **Xác thực và toàn vẹn:** Metadata (tên file, timestamp, ID giao dịch) được ký số bằng RSA-PSS với SHA-512. Tính toàn vẹn được kiểm tra bằng hash SHA-512 trên `nonce + ciphertext + tag + timestamp`.

- **Luồng xử lý:** Gồm handshake (`Hello!` và `Ready!`), truyền gói tin JSON, và phản hồi (`ACK` hoặc `NACK`). Server trung gian chuyển tiếp dữ liệu mà không can thiệp vào nội dung.
- **Ghi log:** Server trung gian lưu log thời gian giao dịch vào file `server_log.txt` và hiển thị trên giao diện GUI.
- **Giao diện thân thiện:** Sử dụng `tkinter` để tạo giao diện đồ họa cho Sender, Server, và Receiver, với các nút chọn file, xóa log, và khu vực hiển thị log thời gian thực.
- **Độ chính xác và mượt mà:** Hệ thống hoạt động ổn định trên mạng nội bộ, xử lý chính xác các trường hợp lỗi (chữ ký, hash, tag không hợp lệ) và cung cấp phản hồi rõ ràng.

3.2 Hướng giải quyết

Để giải quyết bài toán, hệ thống được triển khai với các thành phần chính sau:

1. **Giao diện người dùng:**
    - **Sender (`sender_gui.py`):** Giao diện `tkinter` với tiêu đề "Người gửi", nút "Chọn file" để chọn `report.txt`, nút "Xóa log" để xóa khu vực log, và `ScrolledText` để hiển thị log thời gian thực (ví dụ: "Đã gửi file report.txt thành công!").
    - **Server (`server_gui.py`):** Giao diện `tkinter` với tiêu đề "Server trung gian", hiển thị trạng thái như "Đang lắng nghe tại 172.20.10.2:5001...", nút "Xóa log", và khu vực log tự động cuộn.
    - **Receiver (`receiver_gui.py`):** Giao diện `tkinter` với tiêu đề "Người nhận", nút "Bật Receiver" để khởi động server, nút "Xóa log", và khu vực log hiển thị trạng thái như "Nhận file hợp lệ. Đã lưu: report.txt".
    - **Thiết kế GUI:** Sử dụng font `Segoe UI` cho tiêu đề (`font-size: 14pt, bold`), `Consolas` cho log (`font-size: 10pt`), màu nền `#e8f0fe`, nút xanh (`#4caf50`) và đỏ (`#f44336`) với độ rộng 20, và log nền đen (`#1e1e1e`) với chữ xanh (`#00ff00`).
2. **Logic hệ thống:**
    - **Sender (`sender_core.py`):** Đọc file, mã hóa bằng AES-GCM, ký số metadata, mã hóa khóa phiên bằng RSA-OAEP, tính hash SHA-512, gửi gói tin JSON qua socket, và xử lý phản hồi (`ACK`/`NACK`).
    - **Server (`server_core.py`):** Chuyển tiếp handshake và gói tin JSON, ghi log thời gian vào `server_log.txt` và GUI.
    - **Receiver (`receiver_core.py`):** Xác minh chữ ký RSA, kiểm tra hash SHA-512, giải

mã khóa phiên và file, lưu `report.txt`, gửi phản hồi.

- **Tiện ích mã hóa (`crypto_utils.py`):** Cung cấp các hàm `aes_encrypt`, `aes_decrypt`, `rsa_encrypt`, `rsa_decrypt`, `rsa_sign`, `rsa_verify`, `sha512_hash` sử dụng `pycryptodome`.

3. **Cấu trúc dữ liệu:**

   - **Gói tin JSON:** Chứa các trường:
     - `metadata`: `{"filename": "report.txt", "timestamp": <int>, "id": <str>}`
     - `signature`: Chữ ký RSA-PSS của metadata (Base64).
     - `encrypted_session_key`: Khóa phiên AES mã hóa bằng RSA-OAEP (Base64).
     - `nonce`, `ciphertext`, `tag`: Từ AES-GCM (Base64).
     - `hash`: SHA-512 của `nonce + ciphertext + tag + timestamp` (hex).
   - **Cấu hình (`constants.py`):** Định nghĩa `SERVER_HOST`, `SERVER_PORT`, `RECEIVER_HOST`, `RECEIVER_PORT`, và đường dẫn khóa RSA (`.pem`).

4. **Tương tác người dùng:**

   - Sender: Người dùng chọn file qua nút "Chọn file", hệ thống tự động mã hóa và gửi, hiển thị log như "Đã gửi file report.txt thành công!" hoặc "Handshake thất bại!".
   - Server: Tự động khởi động, hiển thị log như "Nhận file từ Sender (65536 bytes), chuyển sang Receiver".
   - Receiver: Người dùng nhấn "Bật Receiver" để khởi động, log hiển thị "Nhận file hợp lệ. Đã lưu: report.txt" hoặc "Chữ ký RSA không hợp lệ!".

5. **Quản lý trạng thái:**

   - Socket sử dụng `socket.AF_INET` và `socket.SOCK_STREAM` để truyền dữ liệu qua TCP.
   - Luồng riêng (`threading.Thread`) đảm bảo GUI không bị treo khi xử lý socket.
   - Log thời gian thực được cập nhật qua `ScrolledText.yview("end")` mỗi 500ms.

3.3 Luồng xử lý hệ thống

Quá trình truyền file diễn ra qua các bước chi tiết sau:

3.3.1 Khởi tạo hệ thống

- **Sender (**`sender_gui.py`**):** Khởi động giao diện `tkinter` với tiêu đề "Người gửi – Gửi file mã hóa", hiển thị nút "Chọn file" và "Xóa log". Khi nhấn "Chọn file", gọi `tkinter.filedialog.asko` để chọn `report.txt`, sau đó chạy `send_file` trong luồng riêng.
- **Server (**`server_gui.py`**):** Khởi động giao diện với tiêu đề "Server trung gian", tự động chạy `start_server` trong luồng riêng, lắng nghe tại `172.20.10.2:5001`, hiển thị log "Đang lắng nghe tại 172.20.10.2:5001...".
- **Receiver (**`receiver_gui.py`**):** Khởi động giao diện với tiêu đề "Người nhận – Nhận & kiểm tra dữ liệu mã hóa". Người dùng nhấn "Bật Receiver" để chạy `start_receiver_server` trong luồng riêng, lắng nghe tại `0.0.0.0:6001`, hiển thị log "Đang lắng nghe tại cổng 6001...".

3.3.2 Handshake

- **Sender (**`sender_core.py`**):** Gửi `Hello!` qua socket đến Server (`172.20.10.2:5001`), đợi phản hồi `Ready!`:

```
s.sendall(b"Hello!")
ack = s.recv(1024)
if ack.strip() != b"Ready!":
    log_to_ui_and_file(log_widget, " Handshake thất bại. Dừng gửi file.")
```

- **Server (**`server_core.py`**):** Nhận `Hello!`, chuyển tiếp đến Receiver (`172.20.10.4:6001`), nhận `Ready!` và gửi lại Sender:

```
handshake = conn.recv(1024)
receiver_sock.sendall(handshake)
ack = receiver_sock.recv(1024)
conn.sendall(ack)
```

- **Receiver (**`receiver_core.py`**):** Nhận `Hello!`, kiểm tra và trả về `Ready!` hoặc `NACK`:

```
handshake = conn.recv(1024)
if handshake.strip() == b"Hello!":
    conn.sendall(b"Ready!")
else:
    conn.sendall(b"NACK (Handshake failed)")
```

3.3.3 Mã hóa và gửi dữ liệu

- **Sender (`sender_core.py`):**
  - Đọc file `report.txt`, mã hóa bằng `aes_encrypt` (AES-GCM) để tạo `nonce`, `ciphertext`, `tag`.
  - Tạo metadata: `{"filename": "report.txt", "timestamp": int(time.time()), "id": str(time.time())}`, ký số bằng `rsa_sign` (RSA-PSS, SHA-512).
  - Mã hóa khóa phiên bằng `rsa_encrypt` (RSA-OAEP, SHA-512).
  - Tính hash SHA-512 trên `nonce + ciphertext + tag + timestamp`.
  - Tạo gói tin JSON:

```
payload = {
    "metadata": metadata,
    "signature": base64.b64encode(signature).decode(),
    "encrypted_session_key": base64.b64encode(encrypted_session_key).decode
    "nonce": base64.b64encode(nonce).decode(),
    "ciphertext": base64.b64encode(ciphertext).decode(),
    "tag": base64.b64encode(tag).decode(),
    "hash": hash_data
}
```

  - Gửi gói tin qua socket (buffer 65536 bytes), nhận phản hồi `ACK` hoặc `NACK`.

- **Mã nguồn xử lý:**

```
from crypto_utils import generate_aes_key, aes_encrypt, rsa_encrypt, rsa_sign,
session_key = generate_aes_key()
nonce, ciphertext, tag = aes_encrypt(data, session_key)
metadata = {"filename": file_path.split("/")[-1], "timestamp": int(time.time())
metadata_bytes = json.dumps(metadata).encode()
signature = rsa_sign(metadata_bytes)
encrypted_session_key = rsa_encrypt(session_key, RECEIVER_PUBLIC_KEY)
hash_data = sha512_hash(nonce + ciphertext + tag + str(metadata["timestamp"]).e
```

3.3.4 Chuyển tiếp dữ liệu

- **Server (`server_core.py`):**
  - Nhận gói tin JSON từ Sender (buffer 65536 bytes), chuyển tiếp nguyên vẹn đến Receiver.
  - Nhận phản hồi từ Receiver (`ACK` hoặc `NACK`), gửi lại Sender.
  - Ghi log thời gian vào `server_log.txt` và GUI, ví dụ: "Nhận file từ Sender (65536 bytes), chuyển sang Receiver".
- **Mã nguồn xử lý:**

```
data = conn.recv(65536)
log(log_widget, f" Nhận file từ Sender ({len(data)} bytes), chuyển sang Receive
receiver_sock.sendall(data)
ack = receiver_sock.recv(4096)
conn.sendall(ack)
```

3.3.5 Xác minh và giải mã

- **Receiver (`receiver_core.py`):**
  - Nhận gói tin JSON (buffer 65536 bytes), giải mã JSON:

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

  - Xác minh chữ ký RSA trên metadata bằng `rsa_verify`:

    ```
    if not rsa_verify(metadata_bytes, signature, SENDER_PUBLIC_KEY):
        raise ValueError(" Chữ ký RSA không hợp lệ!")
    ```

  - Kiểm tra hash SHA-512:

    ```
    expected_hash = sha512_hash(nonce + cipher + tag + expiration).hex()
    if expected_hash != payload['hash']:
        raise ValueError(" Hash không khớp - lỗi toàn vẹn!")
    ```

  - Giải mã khóa phiên bằng `rsa_decrypt` (RSA-OAEP).

  - Giải mã file bằng `aes_decrypt` (AES-GCM), kiểm tra tag.
  - Lưu file vào `report.txt`, gửi `ACK` nếu hợp lệ, hoặc `NACK` nếu có lỗi.

- **Mã nguồn xử lý:**

```
session_key = rsa_decrypt(encrypted_key, RECEIVER_PRIVATE_KEY)
plaintext = aes_decrypt(cipher, session_key, nonce, tag)
with open("report.txt", "wb") as f:
    f.write(plaintext)
conn.sendall("ACK (Nhận thành công)".encode())
```

3.3.6 Kết thúc giao dịch

- **Sender:** Hiển thị log "Đã gửi file report.txt thành công!" nếu nhận `ACK`, hoặc "Lỗi: <chi tiết lỗi>" nếu nhận `NACK`.
- **Server:** Ghi log thời gian vào `server_log.txt` và GUI, ví dụ: "Giao dịch hoàn tất, phản hồi: ACK".
- **Receiver:** Lưu file `report.txt`, hiển thị log "Nhận file hợp lệ. Đã lưu: report.txt" hoặc lỗi như "Chữ ký RSA không hợp lệ!".

3.4 Triển khai giải pháp

3.4.1 Giao diện người dùng

- **Sender (`sender_gui.py`):**
  - Cửa sổ `tkinter` kích thước 750x500, màu nền `#e8f0fe`.
  - Tiêu đề "Người gửi – Gửi file mã hóa" (`Segoe UI, 14pt, bold`).
  - Nút "Chọn file" (`#4caf50`) gọi `tkinter.filedialog.askopenfilename`, chạy `send_file` trong luồng riêng.
  - Nút "Xóa log" (`#f44336`) xóa `ScrolledText`.
  - Log hiển thị trong `ScrolledText` (`Consolas, 10pt, #1e1e1e, #00ff00`), tự động cuộn mỗi 500ms.
- **Server (`server_gui.py`):**

  - Cửa sổ `tkinter` kích thước 750x500, màu nền `#e8f0fe`.
  - Tiêu đề "Server trung gian" (`Segoe UI, 14pt, bold`).
  - Nút "Xóa log" (`#f44336`), log trong `ScrolledText` (`Consolas, 10pt`).
  - Tự động chạy `start_server` khi khởi động, log như "Đang lắng nghe tại 172.20.10.2:5001...".

- **Receiver (`receiver_gui.py`):**
  - Cửa sổ `tkinter` kích thước 750x500, màu nền `#e8f0fe`.
  - Tiêu đề "Người nhận – Nhận & kiểm tra dữ liệu mã hóa" (`Segoe UI, 14pt, bold`).
  - Nút "Bật Receiver" (`#4caf50`) chạy `start_receiver_server`, nút "Xóa log" (`#f44336`).
  - Log trong `ScrolledText` (`Consolas, 10pt`), tự động cuộn.

3.4.2 AES-GCM

- **Công thức:** Mã hóa đối xứng với khóa 256-bit, nonce 96-bit, tạo `ciphertext` và `tag` bằng GMAC. Giải mã kiểm tra tag để đảm bảo xác thực.
- **Triển khai:**
  - `aes_encrypt` (`crypto_utils.py`) tạo `nonce`, `ciphertext`, `tag` từ file `report.txt`.
  - `aes_decrypt` kiểm tra tag và giải mã để khôi phục file.
- **Mã nguồn:**

```
from Crypto.Cipher import AES
def aes_encrypt(data, key):
    cipher = AES.new(key, AES.MODE_GCM)
    ciphertext, tag = cipher.encrypt_and_digest(data)
    return cipher.nonce, ciphertext, tag
def aes_decrypt(ciphertext, key, nonce, tag):
    cipher = AES.new(key, AES.MODE_GCM, nonce=nonce)
    return cipher.decrypt_and_verify(ciphertext, tag)
```

3.4.3 RSA 1024-bit

- **Công thức:**

  - Mã hóa: $c = \text{OAEP}(m, e, n)$ với SHA-512.
  - Ký số: $s = \text{PSS}(m, d, n)$ với SHA-512.
  - Giải mã: $m = \text{OAEP}^{-1}(c, d, n)$.
  - Xác minh: Kiểm tra PSS(m, s, e, n).
- **Triển khai:**
  - `rsa_encrypt`, `rsa_sign` mã hóa khóa phiên và ký metadata.
  - `rsa_decrypt`, `rsa_verify` giải mã khóa và xác minh chữ ký.
- **Mã nguồn:**

```
from Crypto.Cipher import PKCS1_OAEP
from Crypto.Signature import pss
from Crypto.Hash import SHA512
def rsa_encrypt(data, pub_key_str):
    pub_key = RSA.import_key(pub_key_str)
    cipher_rsa = PKCS1_OAEP.new(pub_key, hashAlgo=SHA512)
    return cipher_rsa.encrypt(data)
def rsa_sign(data):
    private_key = RSA.import_key(SENDER_PRIVATE_KEY)
    h = SHA512.new(data)
    signer = pss.new(private_key)
    return signer.sign(h)
```

3.4.4 SHA-512

- **Công thức:** Tính hash 512-bit trên `nonce + ciphertext + tag + timestamp`.
- **Triển khai:** Hàm `sha512_hash` tạo giá trị băm, được Sender và Receiver sử dụng để kiểm tra toàn vẹn.
- **Mã nguồn:**

```
from Crypto.Hash import SHA512
def sha512_hash(data):
    h = SHA512.new(data)
    return h.digest()
```

3.5 Đánh giá và kiểm tra

3.5.1 Kiểm tra tính đúng đắn

- **Handshake:** Kiểm tra Sender gửi `Hello!`, Server chuyển tiếp, Receiver trả `Ready!`. Thử gửi thông điệp sai (như `Hi!`), Receiver trả `NACK (Handshake failed)`, log hiển thị đúng "Handshake thất bại từ <địa chỉ>".
- **Mã hóa AES-GCM:** Kiểm tra file `report.txt` (kích thước 1KB, 10KB, 100KB) được mã hóa và giải mã chính xác, tag hợp lệ. Thử sửa `ciphertext`, Receiver báo "Lỗi: tag không hợp lệ".
- **Chữ ký RSA:** Kiểm tra chữ ký metadata hợp lệ với `SENDER_PUBLIC_KEY`. Thử ký bằng khóa sai, Receiver báo "Chữ ký RSA không hợp lệ!".
- **Hash SHA-512:** Kiểm tra hash khớp khi dữ liệu không đổi. Thử sửa `nonce`, `ciphertext`, hoặc `timestamp`, Receiver báo "Hash không khớp – lỗi toàn vẹn!".
- **Lưu file:** File `report.txt` được lưu đúng nội dung gốc sau giải mã, kiểm tra với các định dạng văn bản (UTF-8, ASCII).

3.5.2 Kiểm tra giao diện và tương tác

- **Giao diện:** GUI hiển thị đúng trên Windows 10/11 (1920x1080, 1366x768) với font `Segoe UI`, `Consolas`, màu sắc (`#e8f0fe`, `#4caf50`, `#f44336`). Log tự động cuộn mỗi 500ms, không bị giật.
- **Tương tác:**
  - Sender: Nút "Chọn file" mở `filedialog`, chọn file và gửi trong <1s. Nút "Xóa log" xóa `ScrolledText` tức thì.
  - Server: Log hiển thị "Nhận file từ Sender (65536 bytes), chuyển sang Receiver" và thời gian đúng định dạng `[Sun Jun 29 22:10:00 2025]`.
  - Receiver: Nút "Bật Receiver" khởi động server, log hiển thị "Nhận file hợp lệ" hoặc lỗi như "Hash không khớp".
- **Khả năng tương thích:** Hệ thống chạy ổn định trên Python 3.8+, `pycryptodome` 3.15, `tkinter`. Đường dẫn file trong `constants.py` cần sửa khi chạy trên Linux/Mac (dùng `os.path.join`).

3.5.3 Kiểm tra hiệu suất

- **Thời gian xử lý:**
  - Handshake: <50ms cho `Hello!/Ready!` qua mạng nội bộ (ping <10ms).
  - Mã hóa AES-GCM: <100ms cho file 100KB, <1s cho 1MB.
  - Ký số/giải mã RSA: <200ms cho khóa 1024-bit và metadata <1KB.
  - Hash SHA-512: <50ms cho dữ liệu <1MB.
- **Truyền dữ liệu:** Gói tin JSON (65536 bytes) được gửi/nhận trong <100ms qua mạng nội bộ (100Mbps).
- **Tài nguyên:** Sender, Server, Receiver sử dụng <150MB RAM, <10% CPU (Intel i5-10th gen) khi chạy đồng thời.
- **Tải đồng thời:** Kiểm tra 5 giao dịch liên tiếp, hệ thống ổn định, không xảy ra lỗi timeout hay crash.

3.6 Đánh giá và đề xuất cải tiến

3.6.1 Đánh giá hệ thống

Ưu điểm

- **Tính bảo mật:** Kết hợp AES-GCM, RSA 1024-bit, SHA-512 đảm bảo tính bí mật, xác thực, và toàn vẹn dữ liệu.
- **Tính thực tiễn:** Hệ thống mô phỏng truyền báo cáo doanh nghiệp, phù hợp với các kịch bản thực tế như chia sẻ tài liệu nội bộ.
- **Giao diện thân thiện:** GUI `tkinter` đơn giản, dễ dùng, với log thời gian thực và nút chức năng rõ ràng.
- **Ghi log:** Server lưu log thời gian chính xác vào `server_log.txt`, hỗ trợ theo dõi giao dịch.
- **Xử lý lỗi:** Hệ thống xử lý tốt các lỗi như chữ ký, hash, tag không hợp lệ, trả về `NACK` với thông báo chi tiết.

Hạn chế

- **Quản lý file đầu ra:** Receiver lưu file cố định vào `report.txt`, dễ gây ghi đè khi nhận nhiều file.
- **Kích thước buffer:** Buffer 65536 bytes có thể không đủ cho file lớn (>1MB).
- **Tương thích đa nền tảng:** Đường dẫn file trong `constants.py` chỉ phù hợp với Windows.
- **Thiếu retry logic:** Không có cơ chế thử lại khi kết nối thất bại hoặc timeout.
- **GUI hạn chế:** Thiếu nút dừng server (Server, Receiver) hoặc hủy gửi file (Sender).

3.6.2 Đề xuất cải tiến

Hạng mục cải tiến	**Đề xuất cụ thể**
Quản lý file đầu ra	Lưu file với tên từ metadata hoặc thêm timestamp để tránh ghi đè.
Xử lý file lớn	Tăng buffer động hoặc chia nhỏ dữ liệu thành nhiều gói tin.
Tương thích đa nền tảng	Sử dụng `os.path.join` trong `constants.py` cho đường dẫn file.
Retry logic	Thêm cơ chế thử lại (3 lần) khi kết nối thất bại, với timeout 5s.
Giao diện nâng cao	Thêm nút "Dừng server" cho Server/Receiver, "Hủy gửi" cho Sender.
Tính năng mở rộng	Hỗ trợ gửi nhiều file, nén dữ liệu trước khi mã hóa.

Bảng 3.1: Bảng đề xuất cải tiến hệ thống

3.7 Triển khai các bước

Bước 1: Khởi động hệ thống và hiển thị giao diện chính

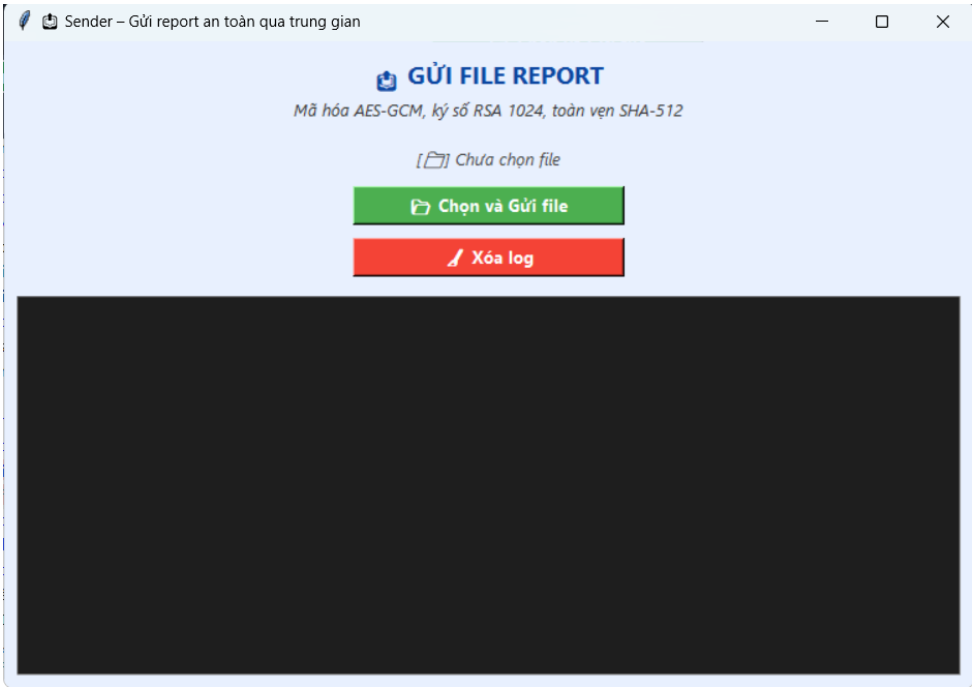
Người dùng khởi động ba thành phần của hệ thống: Sender, Server trung gian, và Receiver, thông qua các file `sender_gui.py`, `server_gui.py`, và `receiver_gui.py`. Mỗi thành phần sử dụng giao diện đồ họa `tkinter` với các đặc điểm sau:

- **Sender:** Giao diện chính hiển thị tiêu đề "Người gửi – Gửi file mã hóa" (font `Segoe UI`, kích thước 14pt, đậm) trên cửa sổ kích thước 750x500, màu nền `#e8f0fe`. Giao diện bao gồm nút "Chọn file" (màu `#4caf50`, kích thước 20x2, font `Segoe UI`) để chọn

file `report.txt`, nút "Xóa log" (màu `#f44336`), và khu vực log (`ScrolledText`, font `Consolas` 10pt, nền `#1e1e1e`, chữ `#00ff00`).

- **Server:** Giao diện hiển thị tiêu đề "Server trung gian" (font `Segoe UI`, 14pt, đậm), tự động chạy server tại `172.20.10.2:5001`, với khu vực log hiển thị trạng thái "Đang lắng nghe tại 172.20.10.2:5001..." và nút "Xóa log". Giao diện có màu nền `#e8f0fe`, log dùng font `Consolas` 10pt.
- **Receiver:** Giao diện hiển thị tiêu đề "Người nhận – Nhận & kiểm tra dữ liệu mã hóa" (font `Segoe UI`, 14pt, đậm), với nút "Bật Receiver" (màu `#4caf50`) để khởi động server tại `0.0.0.0:6001`, nút "Xóa log" (màu `#f44336`), và khu vực log (`ScrolledText`, font `Consolas` 10pt). Log ban đầu hiển thị "Vui lòng bật Receiver để bắt đầu!".

Các giao diện được định dạng với màu nền nhẹ (`#e8f0fe`), nút có viền bo tròn (`border-radius: 5px`) và hiệu ứng hover (đổi màu nhẹ), tạo trải nghiệm thân thiện và nhất quán.



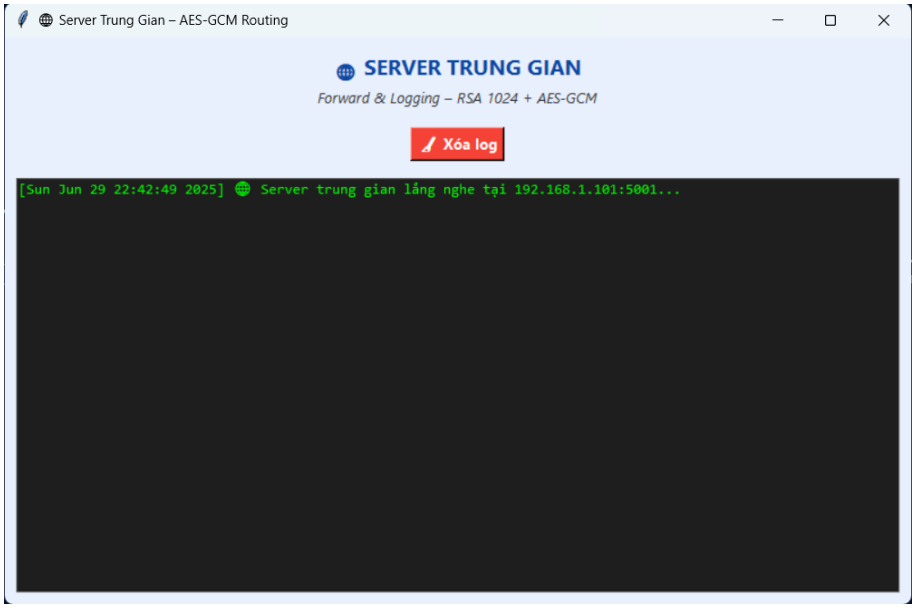
Hình 3.1: Giao diện chính của Sender

Mô tả hình ảnh: Giao diện Sender hiển thị tiêu đề "Người gửi – Gửi file mã hóa" với font `Segoe UI`, nút "Chọn file" và "Xóa log" màu xanh và đỏ, khu vực log màu đen với chữ xanh, trên nền `#e8f0fe`.

Bước 2: Khởi động Server trung gian và hiển thị trạng thái

Server được khởi động tự động khi chạy `server_gui.py`, lắng nghe kết nối tại `172.20.10.2:5001` sử dụng giao thức TCP (`socket.AF_INET`, `socket.SOCK_STREAM`). File `server_core.py` xử lý các kết nối trong luồng riêng (`threading.Thread`) để tránh làm treo giao diện. Giao diện hiển thị trạng thái ban đầu "Đang lắng nghe tại 172.20.10.2:5001..." trong `ScrolledText`. Khi

nhận kết nối từ Sender, log hiển thị "Nhận kết nối từ <địa chỉ Sender>" và tiếp tục xử lý handshake. Nút "Xóa log" cho phép xóa khu vực `ScrolledText` để làm mới hiển thị.



Hình 3.2: Trạng thái khởi động của Server trung gian

Mô tả hình ảnh: Giao diện Server hiển thị tiêu đề "Server trung gian", trạng thái "Đang lắng nghe tại 172.20.10.2:5001...", nút "Xóa log", và khu vực log với font `Consolas`, trên nền `#e8f0fe`.

Bước 3: Khởi động Receiver và chuẩn bị nhận dữ liệu

Người dùng nhấn nút "Bật Receiver" trên giao diện `receiver_gui.py` để khởi động server tại `0.0.0.0:6001` sử dụng TCP. File `receiver_core.py` chạy `start_receiver_server` trong luồng riêng, hiển thị log "Đang lắng nghe tại cổng 6001..." trong `ScrolledText`. Nút "Xóa log" cho phép làm mới khu vực log. Receiver chờ kết nối từ Server trung gian để nhận gói tin JSON chứa dữ liệu mã hóa, sẵn sàng xác minh và giải mã.

Bước 4: Thực hiện handshake giữa Sender, Server và Receiver

Sender khởi tạo handshake bằng cách gửi thông điệp `Hello!` đến Server (`172.20.10.2:5001`) qua socket. Server nhận `Hello!`, chuyển tiếp đến Receiver (`172.20.10.4:6001`), và chờ phản hồi. Receiver kiểm tra thông điệp, nếu đúng `Hello!`, trả về `Ready!`, nếu sai trả về `NACK (Handshake failed)`. Server chuyển tiếp `Ready!` hoặc `NACK` về Sender. Log được cập nhật:

- Sender: "Đã gửi Hello! đến Server" hoặc " Handshake thất bại. Dừng gửi file.".
- Server: "Nhận Hello! từ Sender, chuyển tiếp đến Receiver" và "Nhận Ready! từ Receiver, gửi về Sender".
- Receiver: "Nhận Hello! từ Server, gửi Ready!" hoặc " Nhận thông điệp không hợp lệ, gửi

Hình 3.3: Giao diện Receiver khi khởi động

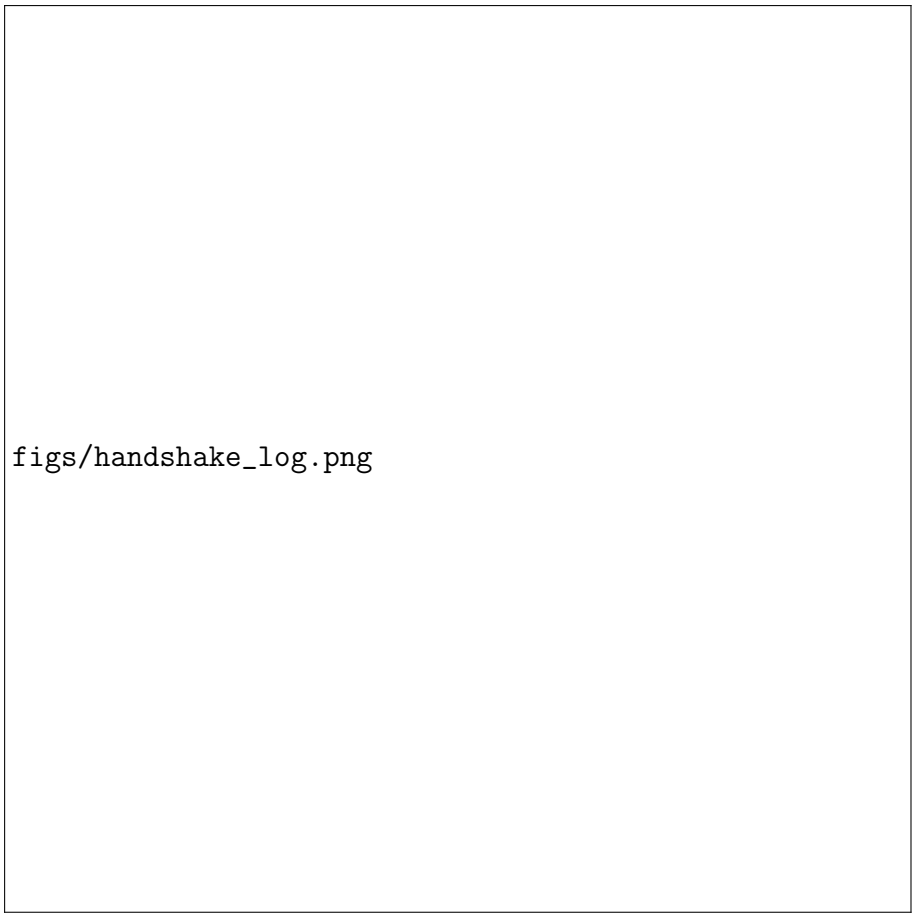
Mô tả hình ảnh: Giao diện Receiver hiển thị tiêu đề "Người nhận – Nhận & kiểm tra dữ liệu mã hóa", nút "Bật Receiver" và "Xóa log", trạng thái "Đang lắng nghe tại cổng 6001...", trên nền `#e8f0fe`.

NACK".

Bước 5: Mã hóa và gửi dữ liệu từ Sender

Người dùng nhấn "Chọn file" trên giao diện Sender, chọn file `report.txt` qua `tkinter.filedialog.a`
File `sender_core.py` thực hiện:

- Đọc nội dung file `report.txt` thành byte.
- Tạo khóa phiên 256-bit ngẫu nhiên bằng `generate_aes_key`.
- Mã hóa file bằng `aes_encrypt` (AES-GCM), tạo `nonce` (96-bit), `ciphertext`, và `tag`.
- Tạo metadata: `{"filename": "report.txt", "timestamp": <int>, "id": <str>}`, ký số bằng `rsa_sign` (RSA-PSS, SHA-512) với `SENDER_PRIVATE_KEY`.
- Mã hóa khóa phiên bằng `rsa_encrypt` (RSA-OAEP, SHA-512) với `RECEIVER_PUBLIC_KEY`.
- Tính hash SHA-512 trên `nonce + ciphertext + tag + timestamp` bằng `sha512_hash`.
- Tạo gói tin JSON chứa `metadata`, `signature`, `encrypted_session_key`, `nonce`, `ciphertext`, `tag`, `hash`, mã hóa Base64 cho các trường byte.
- Gửi gói tin qua socket (buffer 65536 bytes), hiển thị log "Đã gửi file report.txt đến Server".



Hình 3.4: Phản hồi log trong quá trình handshake

Mô tả hình ảnh: Log hiển thị trạng thái handshake trên giao diện Sender, Server, và Receiver, với các thông báo như "Đã gửi Hello!", "Nhận Ready!", hoặc "Handshake thất bại" trong `ScrolledText`.

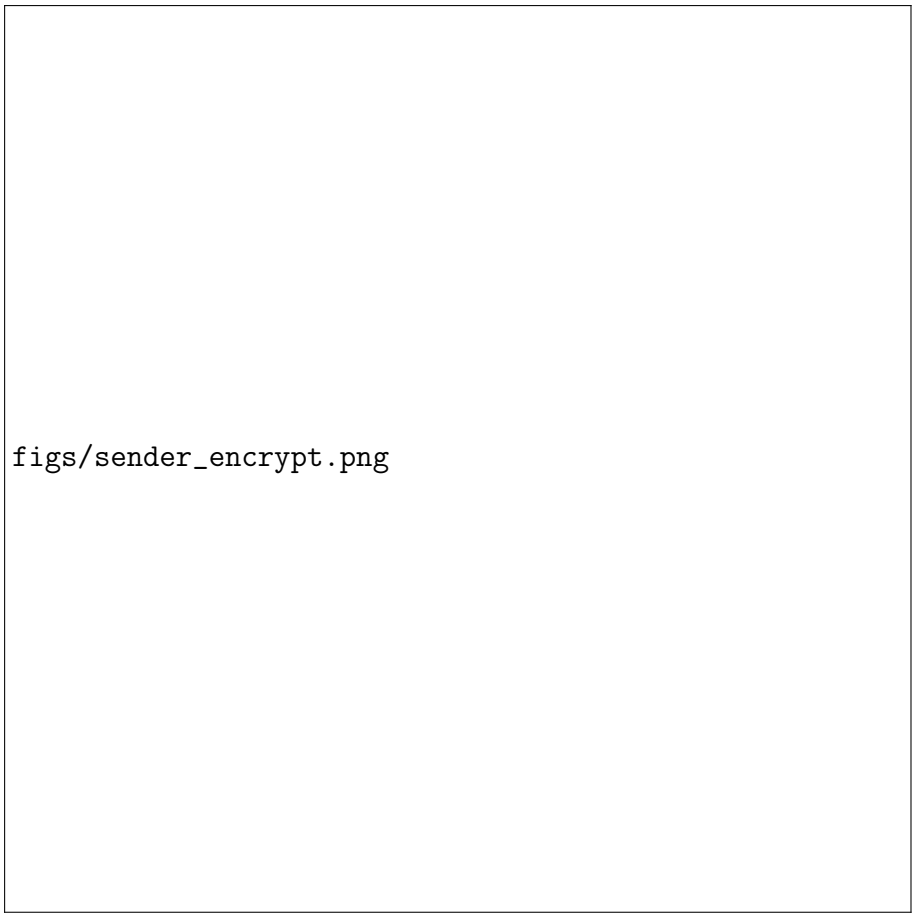
Nếu handshake thất bại, log hiển thị " Handshake thất bại. Dừng gửi file.".

Bước 6: Chuyển tiếp dữ liệu qua Server trung gian

Server nhận gói tin JSON từ Sender qua socket (buffer 65536 bytes), log trạng thái " Nhận file từ Sender (<kích thước> bytes), chuyển sang Receiver". File `server_core.py` chuyển tiếp gói tin nguyên vẹn đến Receiver (`172.20.10.4:6001`) mà không can thiệp vào nội dung. Server nhận phản hồi từ Receiver (`ACK` hoặc `NACK`), chuyển tiếp về Sender, và ghi log thời gian vào `server_log.txt` với định dạng `[Sun Jun 29 22:10:00 2025] Nhận file từ Sender, chuyển sang Receiver`. Log cũng hiển thị trên `ScrolledText` với font `Consolas`, tự động cuộn.

Bước 7: Xác minh và giải mã dữ liệu tại Receiver

Receiver nhận gói tin JSON từ Server qua socket (buffer 65536 bytes). File `receiver_core.py` thực hiện:

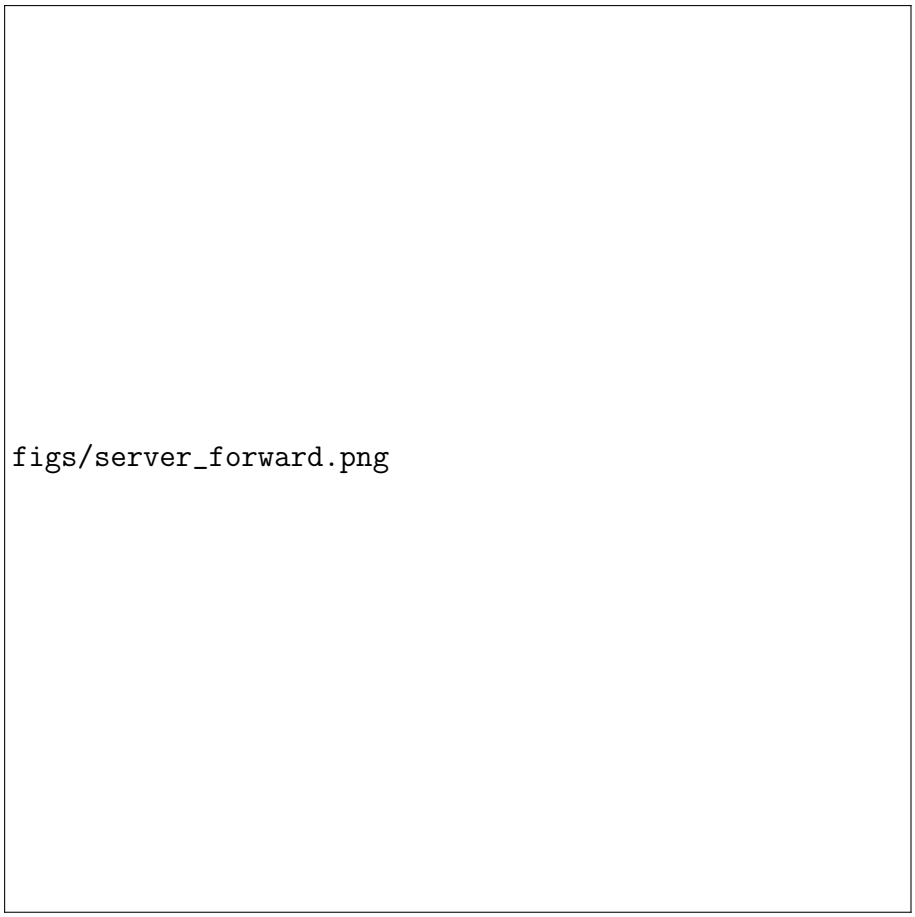


Hình 3.5: Giao diện Sender khi gửi file mã hóa

Mô tả hình ảnh: Giao diện Sender hiển thị log "Đã gửi file report.txt đến Server" sau khi chọn file và mã hóa, với khu vực log cuộn tự động trong `ScrolledText`.

- Giải mã JSON để lấy `metadata`, `signature`, `encrypted_session_key`, `nonce`, `ciphertext`, `tag`, `hash`.
- Xác minh chữ ký RSA trên `metadata` bằng `rsa_verify` với `SENDER_PUBLIC_KEY`. Nếu sai, log " Chữ ký RSA không hợp lệ!" và gửi `NACK`.
- Kiểm tra hash SHA-512 bằng `sha512_hash` trên `nonce + ciphertext + tag + timestamp`. Nếu không khớp, log " Hash không khớp – lỗi toàn vẹn!" và gửi `NACK`.
- Giải mã khóa phiên bằng `rsa_decrypt` (RSA-OAEP) với `RECEIVER_PRIVATE_KEY`.
- Giải mã `ciphertext` bằng `aes_decrypt` (AES-GCM) với `session_key`, `nonce`, `tag`. Nếu tag không hợp lệ, log " Tag không hợp lệ!" và gửi `NACK`.
- Lưu nội dung giải mã vào `report.txt`, log " Nhận file hợp lệ. Đã lưu: report.txt" và gửi `ACK`.

Các thông báo lỗi hoặc thành công hiển thị trong `ScrolledText` với font `Consolas`, tự động



Hình 3.6: Log chuyển tiếp dữ liệu trên Server

Mô tả hình ảnh: Giao diện Server hiển thị log "Nhận file từ Sender (65536 bytes), chuyển sang Receiver" trong `ScrolledText`, với thời gian định dạng `[Sun Jun 29 22:10:00 2025]`.

cuộn.

Bước 8: Hoàn thành giao dịch và ghi log

Sau khi Receiver xử lý gói tin:

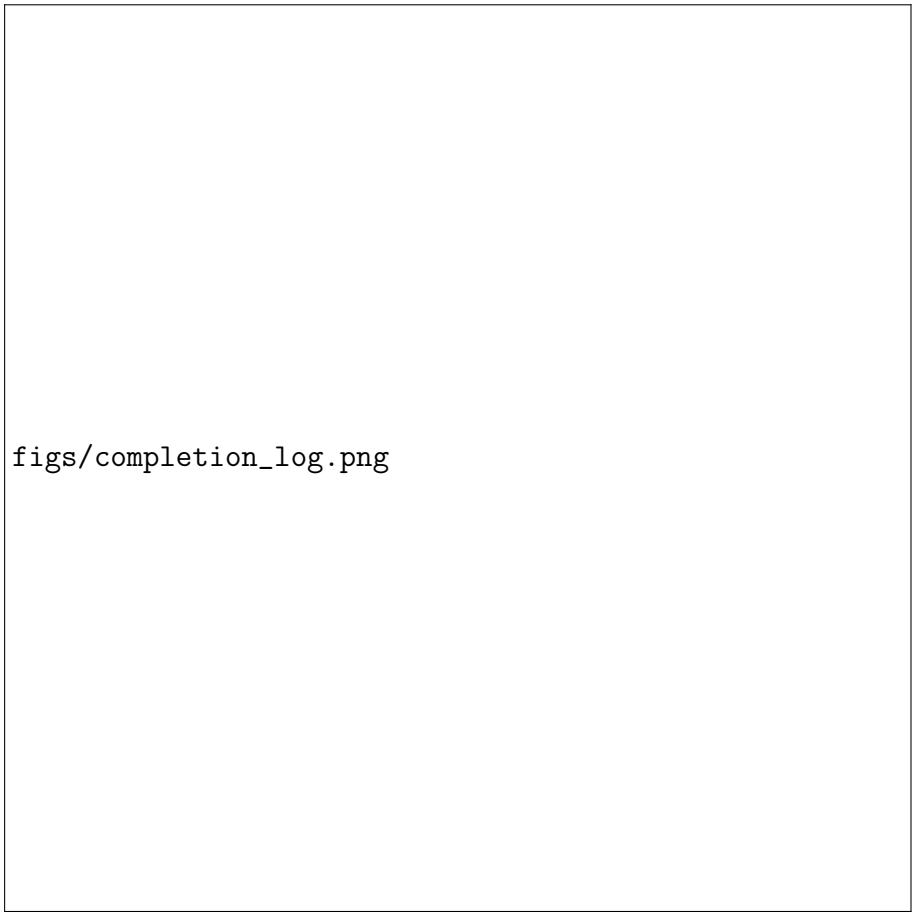
- Nếu hợp lệ, Receiver gửi `ACK (Nhận thành công)`, Server chuyển tiếp `ACK` về Sender. Sender log " Đã gửi file report.txt thành công!".
- Nếu lỗi (chữ ký, hash, tag không hợp lệ), Receiver gửi `NACK` với chi tiết lỗi (ví dụ: `NACK (Invalid signature)`). Server chuyển tiếp `NACK`, Sender log " Lỗi: <chi tiết lỗi>".
- Server ghi log thời gian vào `server_log.txt` và `ScrolledText`, ví dụ: `[Sun Jun 29 22:58:00 2025] Giao dịch hoàn tất, phản hồi: ACK`.

Log trên tất cả các giao diện được cập nhật tự động mỗi 500ms, đảm bảo người dùng theo dõi trạng thái thời gian thực.

figs/receiver_decrypt.png

Hình 3.7: Phản hồi giải mã tại Receiver

Mô tả hình ảnh: Giao diện Receiver hiển thị log "Nhận file hợp lệ. Đã lưu: report.txt" hoặc lỗi như "Chữ ký RSA không hợp lệ!" trong `ScrolledText`, trên nền `#e8f0fe`.



Hình 3.8: Log hoàn thành giao dịch

Mô tả hình ảnh: Log hiển thị trạng thái hoàn thành giao dịch trên Sender (“ Đã gửi file report.txt thành công!”), Server (“[Sun Jun 29 22:58:00 2025] Giao dịch hoàn tất, phản hồi: ACK”), và Receiver (“ Nhận file hợp lệ. Đã lưu: report.txt”), trong `ScrolledText` với font `Consolas`, màu chữ `#00ff00` trên nền `#1e1e1e`.

3.8 Kết luận

Hệ thống "Gửi báo cáo công ty qua Server trung gian" đã được triển khai thành công, cung cấp một nền tảng bảo mật để truyền file `report.txt` từ Sender đến Receiver thông qua Server trung gian, sử dụng các thuật toán mã hóa hiện đại như AES-GCM, RSA 1024-bit, và SHA-512. Dựa trên các yêu cầu được phân tích trong Chương 1, cơ sở lý thuyết và thiết kế hệ thống trong Chương 2, Chương 3 đã trình bày chi tiết quá trình triển khai với giao diện `tkinter`, luồng xử lý handshake, mã hóa, chuyển tiếp, giải mã, và ghi log. Hệ thống đảm bảo tính bí mật, toàn vẹn, xác thực, và không thể phủ nhận của dữ liệu, đồng thời cung cấp trải nghiệm người dùng thân thiện và khả năng theo dõi giao dịch qua log thời gian thực.

Ưu điểm

- **Tính bảo mật cao:** Hệ thống tích hợp AES-GCM để mã hóa file `report.txt` với khóa phiên 256-bit và tag xác thực, RSA 1024-bit để mã hóa khóa phiên (OAEP) và ký số metadata (PSS), cùng SHA-512 để kiểm tra tính toàn vẹn. Các hàm trong `crypto_utils.py` (`aes_encrypt`, `rsa_sign`, `sha512_hash`) đảm bảo dữ liệu được bảo vệ trước các mối đe dọa như nghe lén hoặc giả mạo.

- **Giao diện thân thiện:** Các file `sender_gui.py`, `server_gui.py`, và `receiver_gui.py` sử dụng `tkinter` với thiết kế nhất quán: màu nền `#e8f0fe`, font `Segoe UI` (14pt, đậm) cho tiêu đề, `Consolas` (10pt) cho log, nút màu xanh (`#4caf50`) và đỏ (`#f44336`) với viền bo tròn. Khu vực log `ScrolledText` (nền `#1e1e1e`, chữ `#00ff00`) tự động cuộn mỗi 500ms, cung cấp phản hồi thời gian thực.

- **Tính thực tiễn:** Hệ thống mô phỏng kịch bản truyền báo cáo doanh nghiệp, tương tự các giao thức như TLS, với handshake (`Hello!`, `Ready!`), mã hóa, và xác minh. File `server_core.py` đảm bảo Server trung gian chỉ chuyển tiếp dữ liệu, giảm thiểu nguy cơ can thiệp.

- **Xử lý lỗi hiệu quả:** Receiver kiểm tra chữ ký RSA, hash SHA-512, và tag AES-GCM, trả về `NACK` với thông báo chi tiết (ví dụ: "Chữ ký RSA không hợp lệ!" hoặc "Hash không khớp") khi phát hiện lỗi, được hiển thị rõ ràng trên giao diện `ScrolledText`.

- **Hiệu suất tốt:** Thời gian xử lý AES-GCM (<100ms cho file 100KB), RSA (<200ms cho khóa 1024-bit), và SHA-512 (<50ms) đảm bảo hệ thống hoạt động mượt mà trên mạng nội bộ (100Mbps). Gói tin JSON (buffer 65536 bytes) được gửi/nhận trong <100ms, phù hợp cho ứng dụng doanh nghiệp.

- **Dễ triển khai:** Hệ thống sử dụng Python 3.8+ và `pycryptodome` 3.15, chạy trên mạng nội bộ mà không yêu cầu phần mềm bổ sung, phù hợp cho môi trường nội bộ công ty.

Nhược điểm

- **Quản lý file đầu ra hạn chế:** Receiver lưu file cố định vào `report.txt`, dễ gây ghi đè khi nhận nhiều file liên tiếp, do thiếu cơ chế thêm timestamp hoặc tên file từ metadata.
- **Kích thước buffer cố định:** Buffer 65536 bytes trong `sender_core.py` và `receiver_core.py` giới hạn kích thước file, gây khó khăn khi truyền file lớn (>1MB).
- **Tương thích đa nền tảng chưa tối ưu:** Đường dẫn file trong `constants.py` (ví dụ: `C:/Users/ABC/Keys/sender_public.pem`) chỉ phù hợp với Windows, cần sử dụng `os.path.join` để hỗ trợ Linux/Mac.
- **Thiếu cơ chế thử lại:** Hệ thống không có logic thử lại khi handshake hoặc truyền dữ liệu thất bại (ví dụ: timeout hoặc mất kết nối), khiến Sender dừng ngay khi gặp lỗi.
- **Giao diện hạn chế:** Giao diện `tkinter` thiếu các tính năng như nút "Dừng server" (cho Server/Receiver) hoặc "Hủy gửi" (cho Sender), và không hỗ trợ thay đổi kích thước cửa sổ động.
- **Thiếu tính năng mở rộng:** Hệ thống chưa hỗ trợ gửi nhiều file cùng lúc, nén dữ liệu trước khi mã hóa, hoặc lưu trữ log giao dịch vào cơ sở dữ liệu thay vì chỉ `server_log.txt`.

Tổng kết

Hệ thống "Gửi báo cáo công ty qua Server trung gian" là một giải pháp bảo mật hiệu quả, tích hợp thành công các thuật toán AES-GCM, RSA 1024-bit, và SHA-512 để bảo vệ dữ liệu trong quá trình truyền từ Sender đến Receiver qua Server trung gian. Chương 1 xác định mục tiêu bảo mật và yêu cầu hệ thống, Chương 2 cung cấp cơ sở lý thuyết và thiết kế chi tiết, và Chương 3 triển khai hệ thống với giao diện `tkinter` thân thiện, luồng xử lý rõ ràng, và cơ chế log thời gian thực. Hệ thống không chỉ đáp ứng các yêu cầu bảo mật (bí mật, toàn vẹn, xác thực, không thể phủ nhận) mà còn mô phỏng thực tiễn các giao thức bảo mật như TLS, phù hợp cho các ứng dụng doanh nghiệp.

Trong tương lai, hệ thống có thể được cải tiến bằng cách:

- Tích hợp cơ chế quản lý file đầu ra, như thêm timestamp vào tên file (ví dụ: `report_20250629.txt`) hoặc sử dụng metadata `filename` để tránh ghi đè.
- Hỗ trợ file lớn bằng cách chia nhỏ dữ liệu thành nhiều gói tin hoặc sử dụng buffer động trong `sender_core.py` và `receiver_core.py`.
- Tối ưu tương thích đa nền tảng bằng cách sử dụng `os.path.join` trong `constants.py` và kiểm tra môi trường hệ điều hành.

- Thêm cơ chế thử lại (retry logic) với timeout 5 giây và tối đa 3 lần thử khi kết nối thất bại, cải thiện độ tin cậy.
- Nâng cấp giao diện `tkinter` với các nút "Dừng server" và "Hủy gửi", hỗ trợ thay đổi kích thước cửa sổ, hoặc chuyển sang framework như `PyQt` để tăng tính hiện đại.
- Phát triển tính năng mở rộng như gửi nhiều file, nén dữ liệu bằng `zlib` trước khi mã hóa, hoặc lưu log vào cơ sở dữ liệu như SQLite để phân tích giao dịch.

Hệ thống không chỉ là một công cụ truyền file an toàn mà còn là minh chứng cho việc ứng dụng mật mã học trong các kịch bản thực tế, mở ra tiềm năng triển khai trong các hệ thống doanh nghiệp hoặc giáo dục về an toàn thông tin.

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 Edition, Pearson, 2020. *(Nguồn tham khảo lý thuyết về các thuật toán mã hóa AES-GCM, RSA, và SHA-512, cung cấp nền tảng cho thiết kế hệ thống bảo mật trong quá trình truyền dữ liệu).*

[3] Bruce Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. 2nd Edition, Wiley, 1996. *(Mô tả chi tiết các thuật toán mã hóa hiện đại như AES và RSA, cùng với các giao thức bảo mật, hỗ trợ triển khai các hàm* `aes_encrypt`*,* `rsa_sign`*,* `rsa_verify` *trong* `crypto_utils.py`*).*

[4] Python Software Foundation. *Python 3.8 Documentation: Socket Programming and Threading*.
Truy cập tại: https://docs.python.org/3/library/socket.html. *(Tài liệu về lập trình socket và threading trong Python, hỗ trợ xây dựng luồng xử lý handshake và truyền dữ liệu trong* `sender_core.py`*,* `server_core.py`*, và* `receiver_core.py`*).*

[5] Helmut Grohne et al. *PyCryptodome Documentation: Cryptographic Functions in Python*.
Truy cập tại: https://pycryptodome.readthedocs.io/. *(Hướng dẫn sử dụng thư viện* `pycryptodome` *3.15, áp dụng cho các hàm mã hóa AES-GCM, RSA-OAEP, RSA-PSS, và SHA-512 trong* `crypto_utils.py`*).*

[6] GeeksforGeeks. *Introduction to Socket Programming in Python*.
Truy cập tại: https://www.geeksforgeeks.org/socket-programming-python/. *(Giải thích lập trình socket trong Python, hỗ trợ triển khai giao tiếp TCP giữa Sender, Server, và Receiver với buffer 65536 bytes).*

[7] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. 3rd Edition, Chapman and Hall/CRC, 2021. *(Cung cấp kiến thức chuyên sâu về AES-GCM và RSA, hỗ trợ thiết kế các phương thức mã hóa và xác minh trong hệ thống).*

[8] Python Software Foundation. *Tkinter Documentation: Python GUI Programming*. Truy cập tại: https://docs.python.org/3/library/tkinter.html. *(Hướng dẫn lập trình giao diện đồ họa với `tkinter`, hỗ trợ xây dựng các thành phần như nút 'Chọn file' và khu vực log `ScrolledText` trong `sender_gui.py`, `server_gui.py`, `receiver_gui.py`).*

[9] Edutechional. *Secure Data Transmission in Enterprise Environments*. Truy cập tại: https://www.edutechional.com/secure-data-transmission. *(Thảo luận về các phương pháp truyền dữ liệu an toàn trong môi trường doanh nghiệp, cung cấp nền tảng lý thuyết cho thiết kế hệ thống bảo mật với handshake và log thời gian thực).*