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 CV AN TOÀN CÓ KIỂM TRA IP

Sinh viên thực hiện : Đinh Mai Phương
Phạm Văn Trà
Phạm Thị Ngọc Thanh
Ngành : Công nghệ thông tin
Giảng viên hướng dẫn : ThS. Lê Thị Thùy Trang

Hà Nội - năm 2025

Lời cảm ơn

Trong hành trình học tập và nghiên cứu tại Trường Đại học Đại Nam, nhóm chúng em nhận thức sâu sắc rằng mỗi thành quả đạt được đều là sự cộng hưởng của nhiều yếu tố, trong đó quan trọng nhất là sự dìu dắt tận tình của quý thầy cô, cùng sự đồng hành và hỗ trợ từ bạn bè và gia đình. Nhân dịp hoàn thành bài báo cáo này, nhóm chúng em xin được gửi lời cảm ơn chân thành và sâu sắc nhất tới tất cả những người đã góp phần tạo nên nền tảng kiến thức, tinh thần và động lực cho nhóm trong suốt thời gian qua.

Trước hết, nhóm xin bày tỏ lòng biết ơn sâu sắc đến quý thầy cô khoa Công nghệ Thông tin – những người không chỉ truyền đạt kiến thức một cách bài bản và khoa học, mà còn luôn truyền cảm hứng học tập, khơi gợi tư duy phản biện và tinh thần đổi mới sáng tạo cho sinh viên. Qua những buổi giảng lý thuyết kết hợp thực tiễn, cùng các tình huống tình huống mô phỏng sát với môi trường công nghệ hiện đại, nhóm đã có thêm động lực và nền tảng vững chắc để khám phá những lĩnh vực mới mẻ, giàu tính ứng dụng.

Đặc biệt, nhóm chúng em xin trân trọng tri ân thầy **Trần Đăng Công** và cô **Lê Thị Thùy Trang** – hai giảng viên đã trực tiếp giảng dạy và hướng dẫn học phần *Nhập môn an toàn, bảo mật thông tin*. Những bài giảng tâm huyết, phong cách làm việc chuyên nghiệp cùng sự tận tụy trong quá trình giảng dạy của thầy cô đã giúp nhóm hình thành tư duy hệ thống, kỹ năng thực hành và bản lĩnh khi tiếp cận các vấn đề an toàn thông tin trong kỷ nguyên số. Sự dẫn dắt đầy trách nhiệm ấy chính là nguồn động viên to lớn, tiếp thêm tự tin cho nhóm trong suốt quá trình thực hiện đề tài.

Thông qua đề tài *"Gửi CV an toàn có kiểm tra IP"*, nhóm đã có cơ hội tiếp cận sâu hơn với các vấn đề bảo mật trong quá trình truyền tải thông tin cá nhân trên Internet – một vấn đề ngày càng trở nên cấp thiết trong bối cảnh chuyển đổi số và gia tăng các mối đe dọa an ninh mạng. Việc thiết kế một giải pháp gửi CV an toàn kết hợp kiểm tra IP người nhận nhằm phòng ngừa hành vi giả mạo, rò rỉ hoặc đánh cắp dữ liệu không chỉ giúp nhóm củng cố kiến thức lý thuyết mà còn phát triển kỹ năng phân tích, thiết kế và áp dụng công nghệ để giải quyết các bài toán thực tiễn.

Mặc dù nhóm đã nỗ lực hết sức để hoàn thành bài báo cáo với tinh thần trách nhiệm cao, nhưng do hạn chế về thời gian, kinh nghiệm và năng lực chuyên môn, chắc chắn nội dung vẫn không tránh khỏi những thiếu sót. Nhóm rất mong nhận được sự góp ý chân thành từ quý thầy cô và các bạn để hoàn thiện hơn trong những nghiên cứu, học tập và phát triển sau này.

Một lần nữa, nhóm chúng em xin trân trọng gửi lời cảm ơn đến thầy Trần Đăng Công, cô Lê Thị Thùy Trang và toàn thể quý thầy cô trong khoa đã luôn đồng hành, truyền cảm hứng và tạo dựng nền tảng vững chắc cho sinh viên trên hành trình chinh phục tri thức. Nhóm kính chúc thầy cô luôn dồi dào sức khỏe, ngập tràn niềm vui trong cuộc sống và gặt hái được nhiều thành công mới trên con đường sự nghiệp giáo dục – con đường cao quý của tri thức, truyền cảm hứng và vun đắp tương lai cho bao thế hệ sinh viên.

Mục lục

Danh sách hình vẽ

Chương 1

Giới thiệu đề tài

1.1 Đặt vấn đề

1.1.1 Bối cảnh thực tiễn

Trong thời đại phát triển mạnh mẽ của công nghệ thông tin và xu hướng chuyển đổi số toàn cầu, các hoạt động trong đời sống xã hội và lĩnh vực sản xuất – kinh doanh đang từng bước được tự động hóa và số hóa. Trong đó, quá trình tuyển dụng nhân sự cũng không nằm ngoài xu thế này. Các doanh nghiệp hiện nay ngày càng ưu tiên sử dụng nền tảng trực tuyến để tiếp nhận hồ sơ ứng viên (CV), nhằm tiết kiệm thời gian, chi phí và nâng cao hiệu quả kết nối.

Tuy nhiên, cùng với sự thuận tiện đó, việc trao đổi thông tin cá nhân qua mạng Internet cũng đặt ra nhiều thách thức về bảo mật. Một bản CV thường chứa đựng nhiều thông tin quan trọng như: họ tên, địa chỉ, số điện thoại, chứng chỉ, trình độ học vấn, kinh nghiệm làm việc,... Nếu không được bảo vệ đúng cách, các dữ liệu này rất dễ trở thành mục tiêu của các cuộc tấn công mạng như đánh cắp thông tin, giả mạo danh tính, chèn mã độc hoặc sửa đổi nội dung.

Không chỉ ảnh hưởng đến ứng viên, các doanh nghiệp cũng có nguy cơ bị tổn hại nếu tiếp nhận tệp tin từ các nguồn không rõ ràng hoặc địa chỉ IP không đáng tin cậy. Trong một số trường hợp, những tập tin bị gắn mã độc có thể xâm nhập vào hệ thống nội bộ, làm rò rỉ dữ liệu tuyển dụng hoặc gây gián đoạn hoạt động kinh doanh.

Chính vì vậy, việc phát triển một giải pháp gửi CV an toàn – trong đó tích hợp đầy đủ các cơ chế bảo mật như mã hóa dữ liệu, ký số, kiểm tra tính toàn vẹn và xác minh IP người gửi – là điều hết sức cần thiết. Đây không chỉ là biện pháp phòng vệ trước các mối đe dọa mạng, mà còn là cách để nâng cao uy tín, tính chuyên nghiệp và hiệu quả của quy trình tuyển dụng trong kỷ nguyên số.

1.1.2 Vấn đề đặt ra

Mặc dù việc gửi hồ sơ ứng tuyển qua mạng đã trở nên phổ biến và mang lại nhiều tiện ích vượt trội so với hình thức nộp hồ sơ truyền thống, song điều đó cũng kéo theo nhiều nguy cơ tiềm ẩn về bảo mật và xác thực thông tin. Một CV tưởng chừng đơn giản lại chứa rất nhiều dữ liệu cá nhân nhạy cảm như họ tên, địa chỉ liên lạc, số điện thoại, địa chỉ email, chứng chỉ nghề nghiệp, quá trình học tập và làm việc,... Khi bị lộ hoặc bị can thiệp trái phép, những thông tin này có thể bị lợi dụng cho mục đích xấu như giả mạo danh tính, tấn công lừa đảo hoặc bán dữ liệu cho bên thứ ba.

Thực tế cho thấy, nhiều cuộc tấn công mạng hiện nay không chỉ nhằm vào các tổ chức lớn mà còn nhắm đến những cá nhân hoặc hệ thống nhỏ hơn – nơi bảo mật yếu và lỗ hổng chưa được kiểm soát. Một tập tin CV gửi đến doanh nghiệp có thể là phương tiện phát tán mã độc nếu không được kiểm duyệt nguồn gốc kỹ lưỡng. Do đó, ngoài yêu cầu bảo mật nội dung, hệ thống còn cần có khả năng xác định rõ người gửi là ai, họ có thực sự là ứng viên hợp pháp hay chỉ là một IP ẩn danh nào đó đang cố gắng khai thác lỗ hổng hệ thống.

Từ những nguy cơ thực tiễn nêu trên, vấn đề đặt ra là cần xây dựng một mô hình gửi và nhận file CV an toàn, trong đó đảm bảo đồng thời các yếu tố:

- **Bảo mật nội dung:** Nội dung file CV cần được mã hóa trong suốt quá trình truyền tải nhằm ngăn chặn nguy cơ bị nghe lén hoặc giải mã trái phép, đặc biệt khi dữ liệu di chuyển qua các kênh mạng công cộng hoặc môi trường truyền thông không đảm bảo an toàn.
- **Xác thực người gửi:** Phải có cơ chế xác minh danh tính người gửi – ví dụ bằng cách ký số metadata – nhằm đảm bảo rằng file thực sự được gửi bởi ứng viên hợp pháp, không phải từ một bên giả mạo.
- **Đảm bảo tính toàn vẹn:** Nội dung tập tin không được phép bị thay đổi trong quá trình truyền tải. Một thay đổi dù nhỏ nhất cũng cần được phát hiện kịp thời để tránh các hệ quả nghiêm trọng.
- **Xác minh nguồn gốc IP:** IP người gửi là một thành phần quan trọng để xác định vị trí, mạng truy cập và khả năng tin cậy của người gửi. Hệ thống cần có bước kiểm tra IP ngay từ đầu để loại bỏ các truy cập bất thường hoặc trái phép.

Như vậy, để đảm bảo một quy trình gửi CV trực tuyến vừa hiệu quả, vừa an toàn, cần có sự kết hợp của nhiều lớp bảo vệ: Từ mã hóa dữ liệu, xác thực, kiểm tra toàn vẹn đến xác minh nguồn gốc. Bài toán không chỉ nằm ở kỹ thuật mã hóa đơn thuần mà còn ở việc xây dựng một luồng xử lý hợp lý, linh hoạt và có khả năng ngăn chặn sớm các hành vi độc hại. Đây chính là mục tiêu trọng tâm mà đề tài hướng đến.

1.1.3 Phân tích bài toán

Từ những vấn đề đã được nêu ở trên, bài toán đặt ra là làm thế nào để xây dựng một hệ thống truyền gửi file CV qua mạng Internet một cách an toàn, có khả năng bảo vệ thông tin cá nhân của người dùng, đồng thời đảm bảo độ tin cậy, tính xác thực và toàn vẹn dữ liệu trong suốt quá trình giao tiếp.

Để giải quyết bài toán này, hệ thống cần kết hợp nhiều kỹ thuật bảo mật hiện đại và triển khai theo một quy trình logic chặt chẽ. Mỗi giai đoạn đều đóng vai trò riêng biệt trong việc ngăn chặn các nguy cơ xâm nhập, giả mạo hoặc đánh cắp dữ liệu. Cụ thể, hệ thống cần đáp ứng các yêu cầu kỹ thuật sau:

- **Bảo mật nội dung tệp tin:** Dữ liệu trong CV cần được mã hóa trước khi gửi đi nhằm đảm bảo rằng chỉ người nhận hợp lệ mới có thể giải mã và đọc được nội dung. Phương pháp mã hóa đối xứng AES-CBC được lựa chọn vì tính hiệu quả và độ bảo mật cao khi kết hợp với khóa phiên sinh ngẫu nhiên.
- **Trao đổi khóa an toàn:** Do cơ chế AES sử dụng khóa bí mật, cần có một phương thức trao đổi khóa phiên (session key) an toàn. RSA 1024-bit với chuẩn OAEP được sử dụng để mã hóa khóa phiên, đảm bảo khóa chỉ được giải mã đúng bởi người nhận nắm giữ khóa riêng tương ứng.
- **Xác thực người gửi:** Trước khi truyền tải dữ liệu, người gửi phải thực hiện ký số thông tin metadata (bao gồm tên file, thời điểm gửi, và địa chỉ IP) bằng thuật toán RSA/SHA-512. Điều này giúp người nhận xác minh được danh tính của người gửi và đảm bảo rằng dữ liệu không bị làm giả.
- **Đảm bảo tính toàn vẹn:** Để phát hiện mọi sự can thiệp vào nội dung tệp, hệ thống tính toán giá trị băm bằng thuật toán SHA-512 trên tổ hợp IV và ciphertext. Khi nhận được gói tin, người nhận sẽ đối chiếu lại giá trị này nhằm kiểm tra tính nguyên vẹn của dữ liệu.
- **Xác minh địa chỉ IP:** Ngay từ bước handshake, hệ thống thực hiện kiểm tra IP người gửi nhằm xác định tính hợp lệ và nguồn gốc truy cập. Việc này giúp loại bỏ các yêu cầu đến từ địa chỉ IP không đáng tin cậy hoặc ngoài phạm vi được chấp nhận.

Toàn bộ quy trình truyền và nhận file CV được thiết kế gồm bốn giai đoạn chính, đảm bảo các yếu tố bảo mật, xác thực và toàn vẹn dữ liệu. Cụ thể:

1. **Handshake ban đầu:** Người gửi khởi tạo kết nối bằng cách gửi thông điệp `"Hello!"` kèm theo địa chỉ IP cá nhân. Phía người nhận sẽ tiến hành kiểm tra địa chỉ IP này. Nếu IP hợp lệ (thuộc danh sách được cho phép), hệ thống phản hồi bằng thông điệp `"Ready!"`, cho phép tiếp tục bước xác thực.

2. **Xác thực và trao đổi khóa:** Người gửi tiến hành ký số thông tin metadata, bao gồm: tên file, timestamp (thời điểm gửi), và địa chỉ IP, bằng thuật toán RSA kết hợp với hàm băm SHA-512. Đồng thời, khóa phiên (SessionKey) dùng trong mã hóa AES được mã hóa bằng RSA 1024-bit sử dụng chuẩn OAEP. Sau đó, cả metadata đã ký và khóa phiên được gửi đến người nhận.

3. **Mã hóa và kiểm tra toàn vẹn:** Người gửi sinh một vector khởi tạo (IV), sau đó mã hóa file CV bằng thuật toán AES ở chế độ CBC để tạo ra ciphertext. Tiếp theo, hệ thống tính toán giá trị băm của chuỗi ghép `IV || ciphertext` bằng thuật toán SHA-512 nhằm đảm bảo tính toàn vẹn. Tất cả các thành phần sau sẽ được đóng gói trong một gói tin và gửi đi:

```
{
    "iv": "<Base64>",
    "cipher": "<Base64>",
    "hash": "<hex>",
    "sig": "<Signature>"
}
```

4. **Xử lý phía người nhận:** Khi gói tin được gửi đến, hệ thống phía người nhận sẽ tiến hành các bước kiểm tra và xác minh như sau:

   - **Xác minh địa chỉ IP:** Đảm bảo rằng địa chỉ IP người gửi thuộc nguồn đáng tin cậy và nằm trong danh sách cho phép;
   - **Kiểm tra tính toàn vẹn dữ liệu:** Đối chiếu giá trị hash SHA-512 để phát hiện mọi dấu hiệu thay đổi trong nội dung ciphertext hoặc IV;
   - **Xác thực chữ ký số:** Kiểm tra chữ ký số đi kèm metadata để xác minh danh tính người gửi cũng như đảm bảo tính toàn vẹn thông tin định danh.

   Nếu tất cả các điều kiện trên đều thỏa mãn, hệ thống sẽ sử dụng thuật toán AES-CBC để giải mã ciphertext và khôi phục lại nội dung file gốc. Tệp tin sau khi giải mã sẽ được lưu lại dưới tên `cv.pdf`. Đồng thời, hệ thống sẽ gửi thông báo phản hồi `ACK` về cho người gửi, xác nhận quá trình truyền dữ liệu diễn ra thành công.

   Ngược lại, nếu bất kỳ điều kiện nào không đạt yêu cầu (ví dụ: IP không hợp lệ, hash sai lệch, hoặc chữ ký số không xác thực), hệ thống sẽ từ chối tiếp nhận và gửi phản hồi lỗi `NACK` kèm theo thông tin giải thích nguyên nhân (vi phạm toàn vẹn hoặc lỗi xác thực).

Thông qua việc phân tích bài toán theo cách tiếp cận từng bước và tích hợp đa tầng bảo mật, đề tài không chỉ mô phỏng một quy trình truyền dữ liệu an toàn, mà còn phản ánh rõ cách mà các cơ chế mã hóa, ký số và kiểm tra toàn vẹn phối hợp với nhau trong môi trường mạng thực

tế. Đây là một ví dụ điển hình của việc áp dụng lý thuyết an toàn thông tin vào thực tiễn xử lý dữ liệu cá nhân trong bối cảnh chuyển đổi số toàn diện hiện nay.

1.1.4 Ý nghĩa và phạm vi ứng dụng

Trong bối cảnh nền kinh tế số đang phát triển mạnh mẽ và dữ liệu cá nhân ngày càng trở thành "tài sản số" có giá trị cao, việc đảm bảo an toàn thông tin trong các giao dịch điện tử không còn là lựa chọn, mà là yêu cầu tất yếu. Đề tài "Gửi CV an toàn có kiểm tra IP" không chỉ là một tình huống học thuật, mà còn phản ánh rõ nét một bài toán thực tế mà các tổ chức tuyển dụng, doanh nghiệp công nghệ, và các hệ thống trực tuyến hiện đại đang phải đối mặt.

Về mặt ý nghĩa học thuật, đề tài là cơ hội để người học tiếp cận một mô hình tích hợp nhiều kỹ thuật bảo mật cốt lõi: mã hóa đối xứng (AES-CBC), mã hóa bất đối xứng (RSA-OAEP), chữ ký số và kiểm tra toàn vẹn (SHA-512), cùng với yếu tố kiểm tra địa chỉ IP — một điểm nhấn mang tính thực thi trong môi trường mạng. Việc áp dụng những kiến thức lý thuyết vào một quy trình xử lý cụ thể giúp sinh viên hình dung rõ cách các thành phần bảo mật phối hợp với nhau để xây dựng một hệ thống truyền dữ liệu an toàn, hiệu quả.

Về mặt ứng dụng, mô hình có khả năng mở rộng sang nhiều kịch bản trong thực tế:

- Bảo vệ hồ sơ y tế điện tử trong quá trình gửi giữa các cơ sở khám chữa bệnh;
- Giao tiếp bảo mật giữa các hệ thống chính phủ điện tử hoặc ngân hàng;
- Truyền file nhạy cảm (thông tin khách hàng, hợp đồng, dữ liệu kinh doanh) giữa các chi nhánh nội bộ doanh nghiệp;
- Bảo vệ thông tin định danh trong hệ thống tài chính.

Điều quan trọng là đề tài không chỉ dừng lại ở việc "mã hóa một tập tin", mà còn gợi mở tư duy thiết kế hệ thống theo hướng "an toàn ngay từ kiến trúc". Nó khuyến khích sinh viên không chỉ biết áp dụng công cụ, mà còn hiểu được vì sao các lớp bảo mật cần phải được phối hợp – từ xác thực người dùng, mã hóa nội dung, kiểm tra nguồn gửi, đến xác minh toàn vẹn dữ liệu.

Tóm lại, đề tài vừa là một mô hình kỹ thuật cụ thể, vừa là một thông điệp mạnh mẽ về tầm quan trọng của bảo mật toàn diện trong thời đại dữ liệu hóa và chuyển đổi số toàn cầu.

1.2 Mục tiêu của đề tài

Trong bối cảnh dữ liệu cá nhân trở thành đối tượng bị khai thác và xâm phạm ngày càng phổ biến trên không gian mạng, đề tài này hướng đến việc xây dựng một mô hình truyền tải thông tin an toàn, có khả năng xác minh danh tính, bảo vệ nội dung và ngăn chặn các hành vi tấn công

trung gian. Thông qua đề tài, nhóm tác giả không chỉ muốn minh chứng khả năng vận dụng các kiến thức lý thuyết của học phần An toàn và Bảo mật Thông tin, mà còn hướng đến việc phát triển một quy trình chuẩn hóa – có thể tích hợp vào các hệ thống thực tế trong tương lai.

Cụ thể, đề tài hướng đến các mục tiêu chính sau:

- **Xây dựng luồng xử lý giao tiếp bảo mật:** Thiết kế một quy trình gửi – nhận file CV chặt chẽ gồm nhiều lớp xác minh, trong đó IP người gửi được kiểm tra ngay từ bước khởi tạo kết nối (handshake) để đảm bảo nguồn gốc đáng tin cậy;
- **Đảm bảo tính bảo mật dữ liệu:** Áp dụng thuật toán mã hóa đối xứng AES ở chế độ CBC kết hợp với khóa phiên sinh ngẫu nhiên, giúp bảo vệ toàn bộ nội dung CV khỏi bị giải đoán trong quá trình truyền qua mạng;
- **Bảo vệ khóa phiên và xác thực metadata:** Sử dụng RSA 1024-bit kết hợp chuẩn mã hóa OAEP để trao đổi khóa an toàn; đồng thời áp dụng chữ ký số SHA-512 để xác thực các thông tin đi kèm như tên file, thời gian gửi và địa chỉ IP;
- **Kiểm tra tính toàn vẹn dữ liệu:** Tính toán hàm băm SHA-512 trên IV và ciphertext để phát hiện mọi hành vi chỉnh sửa hoặc giả mạo nội dung file;
- **Thực hiện mô phỏng và kiểm thử:** Triển khai mô hình mô phỏng quy trình gửi – nhận, theo dõi phản hồi từ phía hệ thống nhận, qua đó kiểm tra hiệu quả hoạt động, khả năng phát hiện lỗi và phản hồi chính xác với các tình huống giả định (ví dụ: sai chữ ký, sai IP, sai hash…).

Thông qua việc hoàn thành các mục tiêu trên, đề tài kỳ vọng không chỉ tạo ra một giải pháp kỹ thuật cụ thể, mà còn nâng cao nhận thức về các nguyên tắc thiết kế hệ thống an toàn – nơi mà bảo mật không được xem là chức năng phụ, mà là yếu tố cốt lõi ngay từ kiến trúc ban đầu.

1.3 Cấu trúc của báo cáo

Để đảm bảo tính hệ thống và logic trong việc trình bày nội dung nghiên cứu, báo cáo được chia thành ba chương chính, mỗi chương đảm nhiệm một vai trò riêng trong việc làm rõ quá trình triển khai đề tài từ ý tưởng đến thực thi. Cấu trúc cụ thể như sau:

- **Chương 1 – Giới thiệu đề tài:** Trình bày bối cảnh thực tiễn và lý do hình thành đề tài; phân tích các vấn đề đặt ra trong giao tiếp bảo mật; xác định rõ mục tiêu, ý nghĩa và phạm vi ứng dụng của mô hình gửi CV an toàn có kiểm tra IP.
- **Chương 2 – Phân tích yêu cầu và thiết kế ứng dụng:** Tập trung vào việc mô tả chi tiết các kỹ thuật sử dụng trong đề tài như AES-CBC, RSA-OAEP, SHA-512, quy trình xác

thực IP; đồng thời trình bày sơ đồ luồng xử lý, cấu trúc gói tin, cũng như chiến lược kiểm tra tính toàn vẹn và xác thực dữ liệu.

- **Chương 3 – Kết luận và hướng phát triển:** Tổng hợp các kết quả đạt được, đưa ra đánh giá khách quan về mô hình đã xây dựng, nêu rõ những điểm mạnh, hạn chế còn tồn tại và đề xuất hướng phát triển mở rộng cho bài toán trong tương lai.

Cách bố cục trên không chỉ giúp người đọc dễ theo dõi mà còn phản ánh đúng quy trình nghiên cứu khoa học: từ xác định vấn đề, phân tích kỹ thuật đến đánh giá và mở rộng – đảm bảo tính toàn diện và chiều sâu của một đề tài bảo mật mang tính ứng dụng thực tiễn cao.

Chương 2

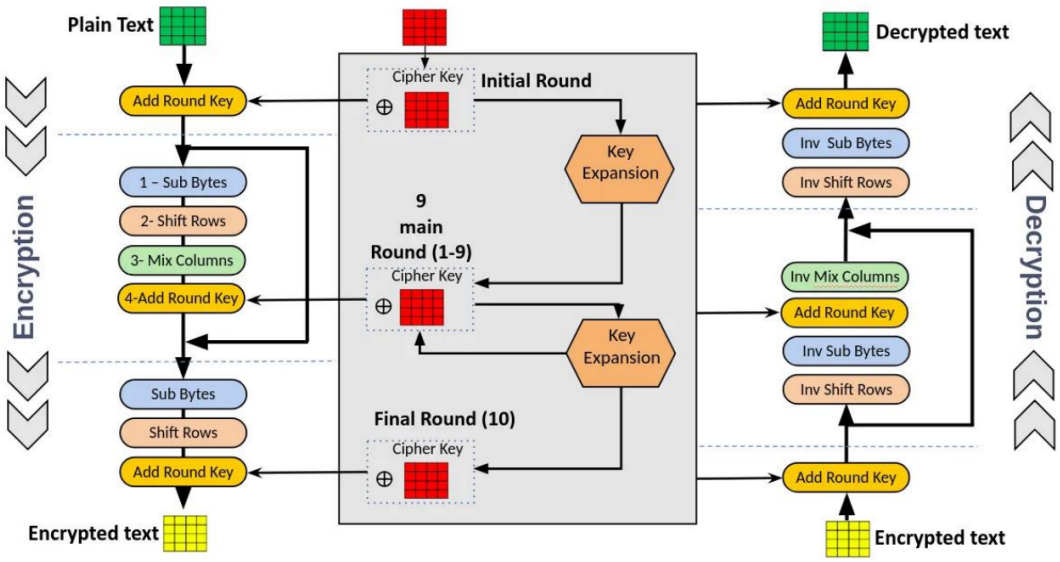
Phân tích yêu cầu và thiết kế ứng dụng

2.1 Mô tả thuật toán

Trong bối cảnh an ninh mạng ngày càng trở nên phức tạp với các nguy cơ tấn công ngày một tinh vi, việc truyền tải thông tin nhạy cảm như hồ sơ xin việc (CV) qua mạng Internet đòi hỏi các biện pháp bảo vệ dữ liệu toàn diện và đáng tin cậy. Hệ thống đề xuất trong đề tài ”Gửi CV an toàn có kiểm tra IP”được thiết kế dựa trên các nguyên lý cốt lõi của an toàn thông tin: **bảo mật** (confidentiality), **toàn vẹn** (integrity), và **xác thực** (authentication), kết hợp cùng cơ chế kiểm soát truy cập theo địa chỉ IP để đảm bảo nguồn gửi hợp lệ.

Thuật toán triển khai được chia thành 4 giai đoạn chính với các kỹ thuật mã hóa và ký số sau:

- **Mã hóa đối xứng (AES-CBC)**: Sử dụng chuẩn AES với chế độ hoạt động CBC để mã hóa nội dung file CV. Đây là thuật toán được công nhận là an toàn và hiệu quả, thường dùng trong các ứng dụng truyền thông dữ liệu bảo mật [1].
  - Advanced Encryption Standard (AES) là một thuật toán mã hóa khối được tiêu chuẩn hóa bởi NIST, hoạt động với khối dữ liệu có kích thước cố định là **128 bit (16 byte)**. Tuy nhiên, kích thước **khóa** có thể linh hoạt, gồm 3 mức: **128 bit**, **192 bit** và **256 bit**, tương ứng với số **vòng lặp** là 10, 12 và 14 [2].



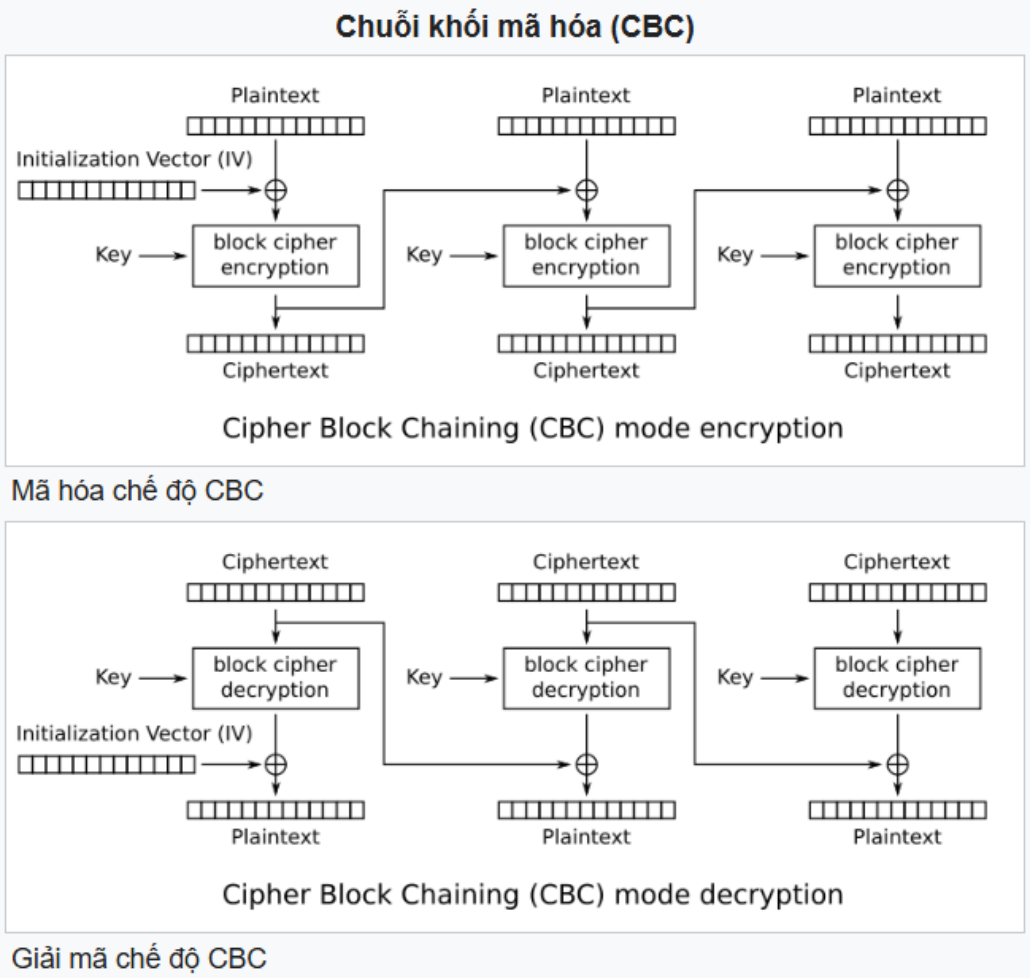
Hình 2.1: Quá trình mã hóa và giải mã AES

- Quá trình mã hóa (Encryption) trong AES bao gồm ba giai đoạn chính: vòng khởi tạo (Initial Round), 9 vòng chính (Rounds 1–9), và vòng cuối cùng (Final Round). Mỗi khối dữ liệu 128 bit ban đầu sẽ được XOR với khóa ban đầu (Add Round Key), sau đó trải qua nhiều vòng biến đổi, bao gồm: *SubBytes* (thay thế byte bằng S-box), *ShiftRows* (dịch các hàng), *MixColumns* (trộn cột), và *Add Round Key* (XOR với khóa con). Vòng cuối cùng sẽ bỏ qua bước *MixColumns* [2].

- Ngược lại, quá trình giải mã (Decryption) thực hiện các thao tác đảo ngược: *Inv ShiftRows*, *Inv SubBytes*, *Add Round Key*, và *Inv MixColumns*, áp dụng theo thứ tự đảo chiều từ các vòng tương ứng. Việc giải mã đúng đòi hỏi phải sử dụng chính xác các khóa con theo thứ tự ngược lại so với khi mã hóa. Quá trình mở rộng khóa (Key Expansion) tạo ra các khóa con từ khóa chính để dùng cho từng vòng mã hóa/giải mã [2].

– Trong chế độ CBC, mỗi khối văn bản thuần túy (*plaintext*) được XOR với khối văn bản mã hóa trước đó (*previous ciphertext block*) trước khi được mã hóa. Theo cách này, mỗi khối văn bản mã hóa không chỉ phụ thuộc vào khối dữ liệu hiện tại mà còn liên quan đến tất cả các khối đã được xử lý trước đó. Điều này giúp tăng độ ngẫu nhiên và tính bảo mật của bản mã đầu ra [1].

– Để đảm bảo mỗi thông điệp là duy nhất và ngăn chặn các đòn tấn công lặp lại, thuật toán CBC yêu cầu một **vectơ khởi tạo** (Initialization Vector - IV) được sinh ngẫu nhiên cho khối đầu tiên. IV này cần được đồng bộ giữa hai bên gửi và nhận [1].



Hình 2.2: Cơ chế mã hóa và giải mã chế độ Cipher Block Chaining (CBC)

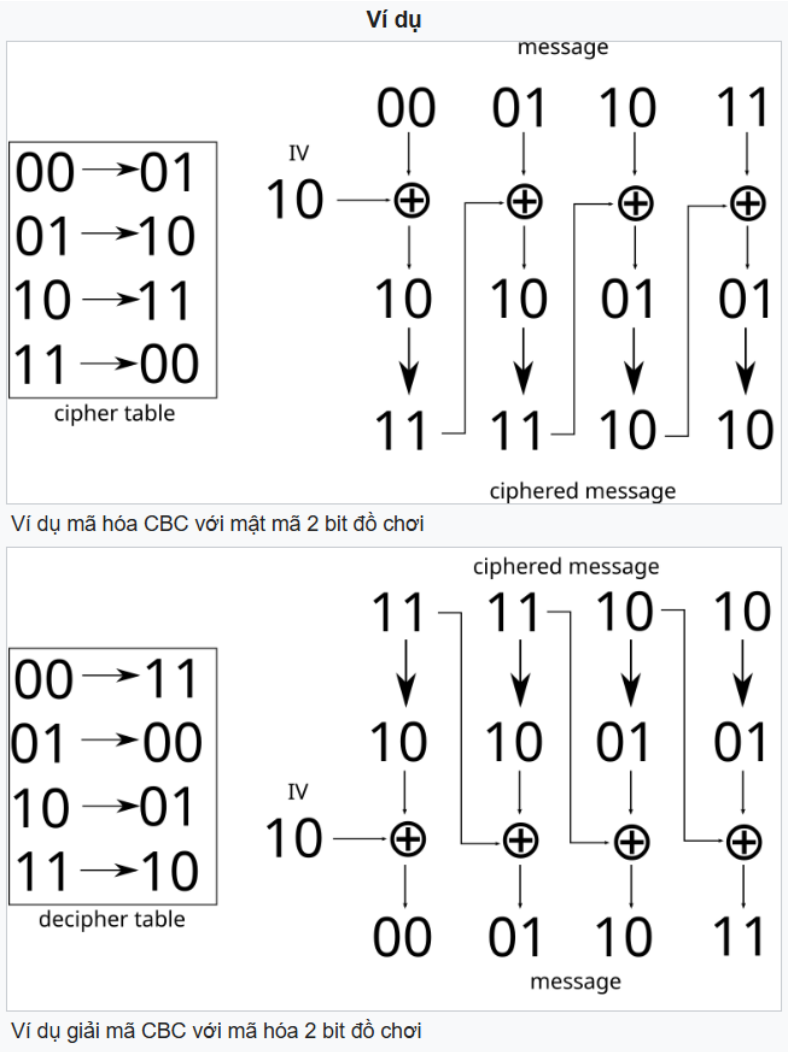
– Nếu khối đầu tiên có chỉ số 1, công thức toán học cho mã hóa CBC là:

$$C_i = E_K(P_i \oplus C_{i-1}), \quad C_0 = IV,$$

trong khi công thức toán học để giải mã CBC là:

$$P_i = D_K(C_i) \oplus C_{i-1}, \quad C_0 = IV. \qquad [1]$$

– Ví dụ mã hóa và giải mã CBC với mật mã 2 bit đồ chơi:



Hình 2.3: Ví dụ minh họa mã hóa và giải mã CBC với mã 2 bit đồ chơi

– Một số đặc điểm bổ sung về CBC:

  * **Không thể song song hóa khi mã hóa:** Trong chế độ CBC, mỗi khối mã hóa C_i phụ thuộc vào kết quả của khối trước đó C_{i-1}. Do đó, toàn bộ quá trình mã hóa phải được thực hiện tuần tự, không thể song song hóa [1].

  * **Yêu cầu đệm dữ liệu (padding):** Vì thuật toán mã hóa khối yêu cầu dữ liệu đầu vào có độ dài là bội số của kích thước khối (ví dụ: 128 bit trong AES), nếu văn bản thuần không đủ độ dài, ta cần thêm padding. Một kỹ thuật thay thế padding là *ciphertext stealing*, cho phép mã hóa mà không cần chèn thêm dữ liệu đệm [1].

  * **Ảnh hưởng khi thay đổi một bit dữ liệu:**

    · Nếu một bit trong **plaintext** hoặc **IV** bị thay đổi, toàn bộ các khối mã hóa phía sau sẽ bị ảnh hưởng. Điều này gây ra lỗi dây chuyền trong mã hóa [1].

· Nếu một bit trong **ciphertext** bị thay đổi:

○ Khối *plaintext* hiện tại (ứng với ciphertext đó) sẽ bị giải mã sai hoàn toàn [1].

○ Khối *plaintext* tiếp theo sẽ bị lỗi tại đúng vị trí bit tương ứng (vì bị XOR với ciphertext sai) [1].

○ Các khối *plaintext* sau đó vẫn được giải mã chính xác [1].

∗ **Giải mã có thể song song hóa:** Mặc dù mã hóa không thể thực hiện song song, nhưng quá trình giải mã thì có thể. Lý do là mỗi khối plaintext P_i chỉ phụ thuộc vào hai khối C_i và C_{i-1} (không cần phải giải mã C_{i-1} trước). Do đó, quá trình giải mã có thể được thực hiện đồng thời cho nhiều cặp khối [1].

∗ **Dễ bị tấn công oracle đệm:** CBC có thể bị khai thác thông qua các cuộc tấn công padding oracle (ví dụ: POODLE). Trong đó, kẻ tấn công lợi dụng việc hệ thống phản hồi lỗi không đúng định dạng padding để từ từ dò ra thông tin về khóa hoặc nội dung được mã hóa [1].

∗ **Sử dụng vectơ khởi tạo rõ ràng (explicit IV):** Một phương pháp khác để thiết lập IV là chèn trực tiếp một khối ngẫu nhiên vào đầu plaintext trước khi mã hóa. Khi giải mã, khối đầu tiên sẽ bị ”hỏng”do không có IV khớp ban đầu, nhưng có thể được loại bỏ an toàn. Phần còn lại của văn bản sẽ được giải mã chính xác mà không cần truyền IV riêng biệt [1].

- **Mã hóa khóa công khai và ký số (RSA-OAEP)**: Sử dụng thuật toán RSA 1024-bit để mã hóa khóa phiên và ký metadata. Chế độ OAEP (Optimal Asymmetric Encryption Padding) được chọn vì khả năng chống tấn công phân tích và định thời.

– Các thành phần cơ bản của hệ mã công khai bao gồm:

∗ **Bản rõ (plaintext)**: Là dữ liệu gốc chưa được mã hóa [3].

∗ **Bản mã (ciphertext)**: Là dữ liệu đã được mã hóa từ bản rõ bằng thuật toán và khóa thích hợp [3].

∗ **Thuật toán mã hóa**: Là phương pháp toán học dùng để chuyển đổi bản rõ thành bản mã và ngược lại [3].

∗ **Khóa công khai (public key) và khóa riêng tư (private key)**: Là một cặp khóa có liên hệ toán học với nhau. Nếu một khóa được sử dụng để mã hóa thì khóa còn lại được dùng để giải mã. Cơ chế này đảm bảo chỉ người giữ khóa riêng mới có thể giải mã dữ liệu được mã hóa bởi khóa công khai tương ứng

[3].

- Thuật toán OAEP (Optimal Asymmetric Encryption Padding) là một kỹ thuật tiền xử lý bản rõ trước khi áp dụng mã hóa bất đối xứng (như RSA), giúp cải thiện độ an toàn cho hệ mã công khai truyền thống. Cụ thể, OAEP hoạt động như một mạng Feistel, sử dụng hai hàm băm giả lập như các oracle ngẫu nhiên là G và H để chuyển đổi bản rõ thành một chuỗi đã được ”xáo trộn”trước khi mã hóa [4].
- Khi kết hợp với bất kỳ hoán vị một chiều có bẫy an toàn nào f (ví dụ: RSA), OAEP được chứng minh là an toàn theo nghĩa mật mã học, trong mô hình oracle ngẫu nhiên, chống lại tấn công văn bản thuần được chọn (IND-CPA). Ngoài ra, với một số hoán vị bẫy như RSA, OAEP còn có thể chống lại tấn công văn bản mã được chọn (IND-CCA) [4].
- OAEP đạt được hai mục tiêu quan trọng:
  - Thêm một thành phần ngẫu nhiên vào bản rõ, giúp biến hệ mã hóa xác định (deterministic) như RSA thành hệ mã xác suất (probabilistic). Nhờ đó, cùng một bản rõ khi mã hóa nhiều lần sẽ tạo ra các bản mã khác nhau, tăng độ an toàn [4].
  - Ngăn chặn khả năng giải mã từng phần bản mã hoặc khai thác thông tin phụ từ bản mã. OAEP đảm bảo rằng kẻ tấn công không thể phục hồi bất kỳ phần nào của bản rõ nếu không thể đảo ngược được hoán vị một chiều bẫy f (như giải RSA khi không có khóa riêng) [4].
- Mã hóa bằng OAEP không chỉ giúp tăng tính bảo mật cho RSA mà còn hỗ trợ tính toàn vẹn và khả năng chống lại các cuộc tấn công kênh phụ (side-channel attacks) bằng cách làm cho bản mã trở nên không thể đoán trước nếu thiếu đầy đủ thông tin. Nhờ đó, OAEP giúp loại bỏ khả năng kẻ tấn công phân tích cấu trúc bản mã để suy ra thông tin về bản rõ [4].
- Cụ thể, quá trình mã hóa OAEP thực hiện theo các bước sau:
  - Sinh một chuỗi ngẫu nhiên r có độ dài cố định (ví dụ: 128 bit).
  - Sử dụng hàm băm giả lập G để tính: $X = M \oplus G(r)$, với M là bản rõ cần mã hóa.
  - Sử dụng hàm H để tính: $Y = r \oplus H(X)$.
  - Ghép cặp (X, Y) lại thành thông điệp đầu vào cho RSA:

$$\text{OAEP}(M) = RSA\_Encrypt(X \parallel Y)$$

- Khi giải mã, các bước được thực hiện ngược lại:
  - Tính lại $r = Y \oplus H(X)$.
  - Khôi phục bản rõ $M = X \oplus G(r)$.
- Quá trình này đảm bảo rằng nếu một trong hai thành phần X hoặc Y bị thay đổi (ví dụ: do tấn công hoặc lỗi truyền dữ liệu), việc giải mã sẽ thất bại hoàn toàn và không cho ra bất kỳ thông tin nào hữu ích [4].
- Ngoài ra, OAEP còn có đặc tính quan trọng: nó có thể được mở rộng để hỗ trợ các thuật toán mã hóa bất đối xứng khác ngoài RSA, miễn là thuật toán đó là một hoán vị một chiều có bẫy (trapdoor permutation). Tuy nhiên, trong thực tế, OAEP chủ yếu được kết hợp với RSA và đã được tiêu chuẩn hóa trong PKCS #1 v2.2 bởi RSA Laboratories [4].
- Mặc dù OAEP rất hiệu quả và an toàn trong mô hình lý tưởng, các nhà nghiên cứu vẫn khuyến nghị khi triển khai thực tế cần lưu ý các yếu tố sau:
  - Cần sử dụng các thư viện mã hóa uy tín, đã được kiểm chứng để tránh lỗi lập trình, đặc biệt là lỗi gây rò rỉ thông tin qua tấn công thời gian (timing attack).
  - Kích thước khóa RSA phải đủ lớn (tối thiểu 2048-bit) để chống lại tấn công dò khóa thô (brute-force).
  - Các hàm băm G và H nên là các hàm có tính ngẫu nhiên mạnh như SHA-256 hoặc SHA-512 để đảm bảo tính không thể đảo ngược.
- **Chữ ký số (Digital Signature)** là một hình thức chữ ký điện tử sử dụng thuật toán mã hóa khóa không đối xứng, bao gồm một cặp khóa: **khóa riêng tư** và **khóa công khai**. Trong đó, **khóa riêng tư** được sử dụng để ký số, còn **khóa công khai** được sử dụng để kiểm tra tính hợp lệ của chữ ký số [3].
- Chữ ký số bảo đảm ba thuộc tính quan trọng của an toàn thông tin:
  - **Tính xác thực** (authentication): xác minh người tạo chữ ký đúng là chủ sở hữu khóa riêng [3].
  - **Tính toàn vẹn** (integrity): đảm bảo thông tin không bị thay đổi kể từ khi được ký [3].
  - **Tính chống chối bỏ** (non-repudiation): người gửi không thể phủ nhận việc đã ký hoặc gửi dữ liệu [3].
- Tuy nhiên, chữ ký số **không đảm bảo tính bí mật** (confidentiality) của thông điệp

dữ liệu, do thông tin có thể vẫn bị đọc nếu không đi kèm quá trình mã hóa [3].

- Trong hệ thống RSA, chữ ký số được tạo bằng cách sử dụng khóa riêng tư để ký, và được xác minh bằng khóa công khai tương ứng [3].

- **Kiểm tra toàn vẹn bằng băm SHA-512**: Sử dụng hàm băm SHA-512 để phát hiện bất kỳ thay đổi nào trong quá trình truyền dữ liệu.
  - SHA-512 (Secure Hash Algorithm 512-bit) là một hàm băm mật mã thuộc họ SHA-2, được phát triển bởi NSA và chuẩn hóa bởi NIST. Nó tạo ra chuỗi băm dài 512 bit bất kể độ dài bản rõ, phù hợp để bảo vệ tính toàn vẹn dữ liệu.
  - Mỗi khi dữ liệu (gồm IV và ciphertext) bị thay đổi dù chỉ một bit, giá trị băm sẽ thay đổi hoàn toàn. Điều này giúp nhanh chóng phát hiện các trường hợp can thiệp hoặc lỗi trong quá trình truyền.
  - SHA-512 là hàm một chiều — nghĩa là không thể đảo ngược về bản rõ từ giá trị băm — và có khả năng chống trùng lặp rất cao, do đó rất an toàn để sử dụng trong các ứng dụng bảo mật như ký số, xác thực và kiểm tra file.
  - Ở thực tế, SHA-512 thường được sử dụng để bảo vệ mật khẩu, kiểm tra tính toàn vẹn của file tải xuống, xác thực tin nhắn và dữ liệu trong các giao thức bảo mật như SSL/TLS và blockchain.
  - Như vậy, việc so sánh giá trị băm được tạo ra phía người gửi và người nhận cho phép phát hiện lỗi hoặc giả mạo hiệu quả, góp phần bảo đảm tính toàn vẹn dữ liệu trong hệ thống gửi CV an toàn.

Toàn bộ quá trình xử lý được mô tả chi tiết như sau:

Bước 1: Handshake & Kiểm tra IP

1. Sender (ứng viên) mở kết nối TCP tới Receiver (hệ thống tuyển dụng) tại địa chỉ và cổng đã định (mặc định `127.0.0.1:12345` hoặc IP thật của máy B).
2. Ngay sau khi kết nối thành công, Sender gửi một chuỗi đơn giản:

   `Hello|<Sender_IP>`

   Trong đó `<Sender_IP>` là IP của chính máy A (ví dụ `"192.168.0.10"` hoặc `"127.0.0.1"`).
3. Receiver nhận chuỗi này, tách theo ký tự `|` để trích ra IP gửi, rồi so sánh với `whitelist` đã cấu hình:

- Nếu IP nằm trong `whitelist` → Receiver trả về `"Ready!"` và tiến tới bước kế tiếp.
- Nếu IP không hợp lệ → Receiver trả về `"NACK (IP)"` và đóng kết nối ngay lập tức.

4. Nhờ cơ chế này, bất kỳ kết nối từ IP không được phép đều bị chặn ngay từ đầu, giảm thiểu nguy cơ tấn công.

Bước 2: Trao đổi khóa công khai & Ký metadata

1. Receiver sinh cặp RSA-1024 (demo) hoặc RSA-2048 (thực tế), lưu lại `priv_recv` và `pub_recv`.
2. Receiver gửi `pub_recv` (PEM) cho Sender.
3. Sender sinh cặp RSA-1024, lưu lại `priv_s` và `pub_s`, rồi gửi `pub_s` (PEM) lại cho Receiver.
4. Sender chuẩn bị metadata theo bytes:

   ```
   metadata = b"<filename>|<timestamp>|<Sender_IP>"
   ```

   Trong đó:

   - `<filename>`: tên file, ví dụ `"cv.pdf"`.
   - `<timestamp>`: thời gian hiện tại (Unix time).
   - `<Sender_IP>`: IP đã dùng ở bước 1.

5. Ký metadata bằng RSA PKCS#1 v1.5 + SHA-512:

   ```
   signature = sign_data(priv_s, metadata)
   ```

   Mục đích: đảm bảo xác thực và toàn vẹn metadata.

Bước 3: Mã hóa dữ liệu & Đóng gói JSON

1. Sinh SessionKey 32 bytes ngẫu nhiên (AES-256):

   ```
   session_key = generate_aes_key()
   ```

2. Mã hóa SessionKey bằng RSA-OAEP sử dụng SHA-256:

   ```
   enc_session_key = rsa_encrypt(pub_recv, session_key)
   ```

3. Sinh IV 16 bytes ngẫu nhiên và đọc nội dung file:

```
iv = os.urandom(16)
with open("cv.pdf", "rb") as f:
    plaintext = f.read()
```

4. Mã hóa nội dung bằng AES-CBC với padding PKCS#7:

```
ciphertext = aes_encrypt_cbc(session_key, iv, plaintext)
```

5. Tính hash toàn vẹn SHA-512 trên `iv || ciphertext`:

```
hash_value = compute_hash(iv, ciphertext)
```

6. Tạo JSON chứa toàn bộ thông tin:

```
{
  "key":   "<Base64 của enc_session_key>",
  "meta":  "<Base64 của metadata>",
  "iv":    "<Base64 của iv>",
  "cipher":"<Base64 của ciphertext>",
  "hash":  "<Hex của hash_value>",
  "sig":   "<Base64 của signature>"
}
```

7. Đóng gói và gửi:

```
packet = package_full(enc_session_key, metadata, iv, ciphertext, hash_value, s
s.sendall(packet)
```

Bước 4: Giải mã & Xác minh ở Receiver

1. Đọc toàn bộ JSON payload (vòng lặp `recv()` đến khi hết dữ liệu), sau đó parse:

```
enc_key, metadata, iv, ciphertext, recv_hash, signature = parse_full(data)
```

2. Giải mã SessionKey:

```
session_key = rsa_decrypt(priv_recv, enc_key)
```

3. Kiểm tra toàn vẹn:

```
if compute_hash(iv, ciphertext) != recv_hash:
    conn.sendall(b"NACK (integrity)")
    return
```

4. Xác thực metadata:

```
if not verify_signature(pub_sender, signature, metadata):
    conn.sendall(b"NACK (auth)")
    return
```

5. Giải mã AES-CBC và lưu file:

```
plaintext = aes_decrypt_cbc(session_key, iv, ciphertext)
with open("cv_received.pdf", "wb") as f:
    f.write(plaintext)
```

6. Nếu mọi bước trên đều thành công:

```
conn.sendall(b"ACK")
```

7. Nếu thất bại ở bất kỳ bước nào:

```
conn.sendall(b"NACK (<lý do>)")
```

2.2 Phân tích mã nguồn

Mã nguồn của hệ thống được tổ chức thành các tệp chính: `sender.py`, `receiver.py`, `crypto_utils.py`, và `protocol.py`, được chia thành hai phía: Người gửi (Sender) và Người nhận (Receiver). Dưới đây là mô tả chi tiết cho từng phía.

2.2.1 Phía Người gửi – `sender.py`

1. **Kết nối đến Receiver:** Người gửi khởi tạo kết nối TCP tới địa chỉ IP và cổng mặc định của phía nhận (`127.0.0.1:12345`). Sau khi kết nối thành công, người gửi gửi chuỗi `"Hello|<Sender_IP>"` để kiểm tra địa chỉ IP.

2. **Trao đổi khóa công khai:** Phía nhận gửi khoá công khai RSA 1024-bit (định dạng PEM) cho Sender. Sender cũng sinh cặp khóa RSA của mình và gửi khoá công khai ngược lại cho phía nhận.

3. **Tạo và ký metadata:** Metadata là chuỗi dạng bytes bao gồm tên file, timestamp và địa chỉ IP:

   ```
   metadata = b"<filename>|<timestamp>|<Sender_IP>"
   ```

   Metadata được ký bằng RSA PKCS#1 v1.5 và hàm băm SHA-512 để đảm bảo xác thực và toàn vẹn.

4. **Mã hóa khóa phiên AES:** Khóa phiên AES-256 được sinh ngẫu nhiên bằng `os.urandom(32)` và được mã hóa bằng RSA (OAEP + SHA-256) với khóa công khai của người nhận.

5. **Mã hóa nội dung tệp CV:** File `cv.pdf` được đọc và mã hóa bằng AES-CBC với IV 16 byte ngẫu nhiên và padding PKCS#7.

6. **Tính toàn vẹn:** Tính giá trị băm SHA-512 trên chuỗi `IV || ciphertext` để kiểm tra toàn vẹn dữ liệu.

7. **Đóng gói và gửi:** Tạo một gói JSON gồm:

   ```
   {
     "key": "<Base64 của enc_session_key>",
     "meta": "<Base64 của metadata>",
     "iv": "<Base64 của iv>",
     "cipher": "<Base64 của ciphertext>",
     "hash": "<Hex của hash_value>",
     "sig": "<Base64 của signature>"
   }
   ```

   Sau đó sử dụng `sendall()` để gửi toàn bộ JSON tới phía nhận.

2.2.2 Phía Người nhận – `receiver.py`

1. **Lắng nghe kết nối và xác minh IP:** Receiver mở socket tại địa chỉ `127.0.0.1:12345`, chấp nhận kết nối từ Sender. Ngay khi nhận chuỗi `Hello|<Sender_IP>`, địa chỉ IP sẽ được kiểm tra với danh sách `whitelist`.

2. **Nhận khóa công khai của Sender:** Receiver lưu khóa công khai từ người gửi để phục vụ cho việc xác minh chữ ký số metadata.

3. **Giải mã khóa phiên:** Sử dụng khóa riêng của Receiver để giải mã `enc_session_key` từ gói tin, bằng thuật toán RSA-OAEP + SHA-256.

4. **Kiểm tra toàn vẹn dữ liệu:** Tính lại giá trị SHA-512 trên chuỗi `iv + ciphertext` và so sánh với giá trị `hash` trong JSON.

5. **Xác thực chữ ký metadata:** Dùng khoá công khai của Sender để kiểm tra chữ ký `sig` ứng với metadata. Nếu không khớp, phản hồi lỗi xác thực.

6. **Giải mã nội dung file:** Nếu kiểm tra toàn vẹn và chữ ký hợp lệ, tiến hành giải mã ciphertext bằng AES-CBC và lưu kết quả ra tệp `cv_received.pdf`.

7. **Phản hồi kết quả:** Nếu toàn bộ quá trình xử lý thành công, hệ thống gửi `ACK` trở lại. Nếu phát hiện lỗi tại bất kỳ bước nào, gửi `NACK (<lý do>)` cho phía gửi.

2.2.3 Các mô-đun hỗ trợ – `crypto_utils.py` và `protocol.py`

- `crypto_utils.py`: Cung cấp các hàm sinh khóa RSA, mã hóa/giải mã RSA (OAEP + SHA-256), ký số và xác thực chữ ký (PKCS#1 v1.5 + SHA-512), mã hóa AES-CBC và xử lý padding PKCS#7.

- `protocol.py`: Định nghĩa các hàm đóng gói và phân tích gói JSON, hàm kiểm tra toàn vẹn (SHA-512), xác minh thời gian hết hạn (nếu có).

2.3 Thử nghiệm

Quá trình thử nghiệm được thực hiện nhằm kiểm tra tính đúng đắn và độ an toàn của hệ thống truyền file CV có xác minh IP, sử dụng mã hóa AES-CBC, trao khóa RSA-OAEP và ký số SHA-512. Toàn bộ các tệp mã nguồn được chạy trực tiếp bằng phần mềm **IDLE (Python GUI)** – công cụ mặc định đi kèm trình cài đặt Python trên Windows.

Quy trình thực hiện trên IDLE

1. Mở tệp `receiver.py` bằng IDLE, sau đó nhấn `F5` để chạy chương trình. Phía người nhận sẽ khởi động một socket, hiển thị thông báo "Đang chờ kết nối..." và lắng nghe kết nối đến từ phía gửi.
2. Tiếp theo, mở tệp `sender.py` trong một cửa sổ IDLE khác và nhấn `F5`. Chương trình sẽ yêu cầu nhập địa chỉ IP của máy người nhận (ví dụ: `127.0.0.1` nếu chạy cùng máy).
3. Chương trình người gửi tiến hành các bước:
    - Gửi chuỗi handshake chứa địa chỉ IP.
    - Nhận khóa công khai của người nhận và gửi lại khóa công khai của mình.
    - Tạo metadata (tên file, thời gian, IP), ký số bằng RSA.
    - Sinh khóa AES, mã hóa tệp `cv.pdf` bằng AES-CBC.
    - Tính hash toàn vẹn SHA-512 và đóng gói vào JSON.
    - Gửi toàn bộ gói tin qua socket.
4. Phía người nhận thực hiện kiểm tra:
    - Kiểm tra IP hợp lệ từ handshake.
    - Giải mã khóa phiên AES.
    - Tính lại hash và so sánh.
    - Xác minh chữ ký metadata.
    - Giải mã nội dung và lưu thành file `cv_received.pdf`.
5. Nếu tất cả các bước đều hợp lệ, hệ thống phản hồi `ACK` về phía người gửi. Ngược lại, nếu xảy ra lỗi tại bất kỳ bước nào, hệ thống phản hồi `NACK (<lý do>)` tương ứng.

Các trường hợp kiểm thử

1. **Trường hợp hợp lệ:**
    - Địa chỉ IP đúng, metadata hợp lệ, dữ liệu không bị thay đổi.
    - Hệ thống lưu file thành công và phản hồi `ACK`.
2. **Sai IP:**

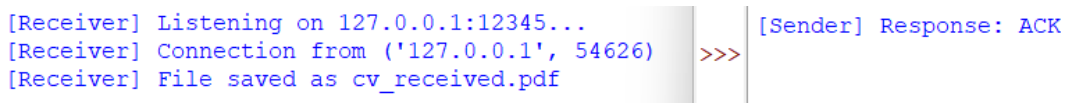
- IP người gửi không nằm trong danh sách cho phép.
- Hệ thống trả về `NACK (IP)` và từ chối kết nối ngay sau handshake.

3. **Sai chữ ký số:**
   - Metadata bị chỉnh sửa hoặc không được ký bằng khóa riêng đúng.
   - Hệ thống trả về `NACK (auth)`.
4. **Sai giá trị hash (toàn vẹn):**
   - Ciphertext hoặc IV bị thay đổi sau mã hóa.
   - Hệ thống trả về `NACK (integrity)`.

Kết quả thử nghiệm

Trong các lần chạy thử nghiệm, hệ thống cho kết quả đúng như kỳ vọng:

- Khi người gửi có IP hợp lệ và file không bị chỉnh sửa, hệ thống phản hồi `ACK` và lưu thành công tệp `cv_received.pdf`.
- Các lỗi sai IP, sai chữ ký hoặc sai hash đều được phát hiện và từ chối ngay tại bước kiểm tra tương ứng, với phản hồi `NACK` kèm lý do rõ ràng.
- Thử nghiệm trên cùng một máy (với IP `127.0.0.1`) cho kết quả ổn định, có thể mở rộng sang mạng LAN với IP nội bộ (ví dụ `192.168.0.4`).
- Ở chương trình này nhóm em thực hiện chương trình trên cùng một máy và đạt được kết quả như mong đợi. Đối với thử nghiệm trên 2 máy khác nhau, nhóm em đang cố gắng hoàn thiện một cách tốt nhất trong tương lai.



Hình 2.4: Kết quả đạt được

2.4 Mã nguồn chương trình

Dưới đây là toàn bộ mã nguồn của chương trình, được chia thành bốn tệp tương ứng với các chức năng chính trong hệ thống.

File: `crypto_utils.py`

```
from cryptography.hazmat.primitives.asymmetric import rsa, padding
from cryptography.hazmat.primitives import serialization, hashes
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes
import os

def generate_rsa_keypair():
    private_key = rsa.generate_private_key(public_exponent=65537, key_size=1024)
    public_key = private_key.public_key()
    return private_key, public_key

def rsa_encrypt(public_key, data):
    return public_key.encrypt(
        data,
        padding.OAEP(
            mgf=padding.MGF1(algorithm=hashes.SHA256()),
            algorithm=hashes.SHA256(),
            label=None
        )
    )

def rsa_decrypt(private_key, ciphertext):
    return private_key.decrypt(
        ciphertext,
        padding.OAEP(
            mgf=padding.MGF1(algorithm=hashes.SHA256()),
            algorithm=hashes.SHA256(),
            label=None
        )
    )

def sign_data(private_key, data):
    return private_key.sign(
        data,
        padding.PKCS1v15(),
        hashes.SHA512()
    )
```

```
def verify_signature(public_key, signature, data):
    try:
        public_key.verify(
            signature,
            data,
            padding.PKCS1v15(),
            hashes.SHA512()
        )
        return True
    except Exception:
        return False

def generate_aes_key():
    return os.urandom(32)

def aes_encrypt_cbc(key, iv, data):
    cipher = Cipher(algorithms.AES(key), modes.CBC(iv))
    encryptor = cipher.encryptor()
    pad_len = 16 - len(data) % 16
    data += bytes([pad_len]) * pad_len
    return encryptor.update(data) + encryptor.finalize()

def aes_decrypt_cbc(key, iv, data):
    cipher = Cipher(algorithms.AES(key), modes.CBC(iv))
    decryptor = cipher.decryptor()
    padded = decryptor.update(data) + decryptor.finalize()
    pad_len = padded[-1]
    return padded[:-pad_len]
```

File: `protocol.py`

```
import base64
import hashlib
import json

def compute_hash(iv, ciphertext):
    return hashlib.sha512(iv + ciphertext).hexdigest()

def package_full(enc_key, metadata, iv, ciphertext, hash_value, signature):
```

```
    return json.dumps({
        "key": base64.b64encode(enc_key).decode(),
        "meta": base64.b64encode(metadata).decode(),
        "iv": base64.b64encode(iv).decode(),
        "cipher": base64.b64encode(ciphertext).decode(),
        "hash": hash_value,
        "sig": base64.b64encode(signature).decode()
    }).encode()

def parse_full(msg):
    obj = json.loads(msg.decode())
    return (
        base64.b64decode(obj["key"]),
        base64.b64decode(obj["meta"]),
        base64.b64decode(obj["iv"]),
        base64.b64decode(obj["cipher"]),
        obj["hash"],
        base64.b64decode(obj["sig"])
    )
```

File: `sender.py`

```
import socket, time
from crypto_utils import *
from protocol import compute_hash, package_full
from cryptography.hazmat.primitives import serialization

def main():
    host, port = "127.0.0.1", 12345
    #Nếu chạy 2 máy thì sửa thành:
    # host = "<IP của máy B>"
    # Gửi "Hello|<IP của máy A>" trong handshake.
    time.sleep(1)  # đảm bảo Receiver đã listen

    # 1) Kết nối + Handshake
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect((host, port))
    s.sendall(f"Hello|{host}".encode())
    if s.recv(1024) != b"Ready!":
```

```
    print("Refused"); return

# 2) Nhận Receiver's public key
pem_recv = b""
while b"-----END PUBLIC KEY-----" not in pem_recv:
    pem_recv += s.recv(1024)
pub_recv = serialization.load_pem_public_key(pem_recv)

# 3) Gửi Sender's public key
priv_s, pub_s = generate_rsa_keypair()
pem_sender = pub_s.public_bytes(
    serialization.Encoding.PEM,
    serialization.PublicFormat.SubjectPublicKeyInfo
)
s.sendall(pem_sender)

# 4) Chuẩn bị metadata + ký
filename, ts = "cv.pdf", str(time.time())
metadata = f"{filename}|{ts}|{host}".encode()
sig = sign_data(priv_s, metadata)

# 5) AES encrypt
session_key = generate_aes_key()
iv = os.urandom(16)
with open("cv.pdf","rb") as f: plain = f.read()
cipher = aes_encrypt_cbc(session_key, iv, plain)
hval = compute_hash(iv, cipher)

# 6) RSA encrypt session key
enc_key = rsa_encrypt(pub_recv, session_key)

# 7) Gói & gửi 1 lần
packet = package_full(enc_key, metadata, iv, cipher, hval, sig)
s.sendall(packet)
s.shutdown(socket.SHUT_WR)

# 8) Chờ ACK
resp = s.recv(1024)
print("[Sender] Response:", resp.decode())
```

```
if __name__=="__main__":
    main()
```

File: `receiver.py`

```
import socket
from crypto_utils import *
from protocol import compute_hash, parse_full
from cryptography.hazmat.primitives import serialization

def main():
    host, port = "127.0.0.1", 12345
    whitelist = {"127.0.0.1"}
    # Nếu chạy 2 máy thì đổi thành:
    #host = "0.0.0.0" hoặc IP cụ thể của máy B.
    #whitelist = {"<IP của máy A>"}.
    priv, pub = generate_rsa_keypair()

    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.bind((host, port))
    s.listen(1)
    print(f"[Receiver] Listening on {host}:{port}...")

    conn, addr = s.accept()
    print(f"[Receiver] Connection from {addr}")

    # 1) Handshake + IP check
    hello = conn.recv(1024).decode()
    if not hello.startswith("Hello|"):
        conn.sendall(b"NACK"); return
    sender_ip = hello.split("|",1)[1]
    if sender_ip not in whitelist:
        conn.sendall(b"NACK (IP)"); return
    conn.sendall(b"Ready!")

    # 2) Gửi Receiver's public key
    pem_recv = pub.public_bytes(
        serialization.Encoding.PEM,
```

```
        serialization.PublicFormat.SubjectPublicKeyInfo
    )
    conn.sendall(pem_recv)

    # 3) Nhận Sender's public key
    pem_sender = b""
    while b"-----END PUBLIC KEY-----" not in pem_sender:
        pem_sender += conn.recv(1024)
    pub_sender = serialization.load_pem_public_key(pem_sender)

    # 4) Nhận gói dữ liệu duy nhất
    data = b""
    while True:
        chunk = conn.recv(4096)
        if not chunk: break
        data += chunk

    enc_key, metadata, iv, cipher, hval, sig = parse_full(data)

    # 5) Giải khóa phiên
    session_key = rsa_decrypt(priv, enc_key)

    # 6) Kiểm toàn vẹn
    if compute_hash(iv, cipher) != hval:
        conn.sendall(b"NACK (integrity)"); print("[Receiver] integrity fail"); retu

    # 7) Kiểm chữ ký với Sender's public key
    if not verify_signature(pub_sender, sig, metadata):
        conn.sendall(b"NACK (auth)"); print("[Receiver] auth fail"); return

    # 8) Giải mã file và lưu
    plain = aes_decrypt_cbc(session_key, iv, cipher)
    with open("cv_received.pdf", "wb") as f:
        f.write(plain)

    conn.sendall(b"ACK")
    print("[Receiver] File saved as cv_received.pdf")

if __name__=="__main__":
```

```
main()
```

2.5 Biểu diễn giải thuật

Hệ thống truyền file CV an toàn được chia thành hai vai trò: Người gửi và Người nhận. Mỗi bên thực hiện một chuỗi hành động cụ thể theo thứ tự, đảm bảo các mục tiêu bảo mật: bảo mật nội dung (confidentiality), toàn vẹn dữ liệu (integrity), và xác thực nguồn gửi (authentication). Dưới đây là biểu diễn giải thuật chi tiết.

Thuật toán phía Người gửi (`sender.py`)

1. Nhập địa chỉ IP của người nhận (Receiver).
2. Khởi tạo kết nối TCP tới IP và cổng 12345.
3. Gửi chuỗi `"Hello|<Sender_IP>"` để xác minh IP.
4. Nếu nhận `"Ready!"` thì tiếp tục, ngược lại dừng kết nối.
5. Nhận khóa công khai RSA từ phía Receiver.
6. Sinh cặp khóa RSA cho người gửi; gửi lại khóa công khai cho Receiver.
7. Tạo metadata: `<filename>|<timestamp>|<Sender_IP>`.
8. Ký metadata bằng khóa riêng RSA-SHA512.
9. Sinh khóa phiên AES ngẫu nhiên (256-bit) và IV (16 byte).
10. Mã hóa nội dung tệp `cv.pdf` bằng AES-CBC.
11. Tính hash toàn vẹn bằng SHA-512 trên chuỗi `IV || ciphertext`.
12. Mã hóa khóa AES bằng RSA-OAEP (SHA-256).
13. Đóng gói toàn bộ vào một gói JSON gồm:
    - `key` – khóa AES đã mã hóa (Base64)
    - `meta` – metadata (Base64)
    - `iv` – IV AES (Base64)
    - `cipher` – ciphertext (Base64)
    - `hash` – giá trị SHA-512 (hex)

- `sig` – chữ ký metadata (Base64)

14. Gửi toàn bộ gói JSON qua socket tới Receiver.
15. Nhận phản hồi: `ACK` nếu thành công, `NACK` nếu lỗi.

Thuật toán phía Người nhận (`receiver.py`)

1. Khởi động máy chủ tại địa chỉ IP và cổng 12345, chờ kết nối.
2. Khi có kết nối đến, nhận chuỗi handshake: `"Hello|<Sender_IP>"`.
3. Kiểm tra IP có nằm trong danh sách `whitelist` không:
    - Nếu hợp lệ → gửi `"Ready!"` và tiếp tục.
    - Nếu không → gửi `"NACK (IP)"` và kết thúc kết nối.
4. Gửi khóa công khai RSA (PEM) cho người gửi.
5. Nhận khóa công khai từ Sender.
6. Nhận gói JSON chứa: `enc_session_key, metadata, iv, ciphertext, hash, sig`.
7. Giải mã khóa phiên AES bằng khóa riêng RSA-OAEP (SHA-256).
8. Tính lại hash SHA-512 trên chuỗi `IV || ciphertext`:
    - Nếu không khớp → gửi `NACK (integrity)` và dừng.
9. Xác thực chữ ký số trên metadata bằng RSA-SHA512:
    - Nếu không đúng → gửi `NACK (auth)` và dừng.
10. Giải mã ciphertext bằng AES-CBC (với IV và khóa phiên).
11. Ghi dữ liệu thành tệp `cv_received.pdf`.
12. Gửi phản hồi `ACK` về cho phía Sender.

Chương 3

Kết luận và hướng phát triển

3.1 Phân tích hiệu quả

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

Hệ thống "Gửi CV an toàn có kiểm tra IP" được xây dựng nhằm giải quyết một vấn đề thực tiễn trong quá trình tuyển dụng hiện đại: truyền tải hồ sơ ứng viên qua mạng một cách an toàn, xác thực và toàn vẹn. Trong bối cảnh thông tin cá nhân có thể bị đánh cắp, giả mạo hoặc sửa đổi, việc tích hợp các thuật toán mã hóa và kiểm tra an toàn là lựa chọn tất yếu.

Mô hình hiện tại vận dụng tổng hợp nhiều kỹ thuật bảo mật phổ biến, bao gồm:

- **Thuật toán RSA-OAEP 1024-bit:** Được dùng để mã hóa khóa phiên AES trước khi truyền. Mặc dù độ dài 1024-bit còn giới hạn về mức độ bảo mật (so với chuẩn khuyến nghị 2048-bit), nhưng việc sử dụng chế độ OAEP giúp chống lại các tấn công dựa trên bản rõ đã biết (known-plaintext) và làm cho hệ thống mạnh hơn so với việc sử dụng RSA với padding PKCS#1 cổ điển. Đây là sự kết hợp phù hợp giữa bảo mật và hiệu năng trong môi trường mô phỏng.
- **Mã hóa đối xứng AES-CBC với khóa 256-bit:** Đây là một trong những thuật toán được công nhận và chuẩn hóa bởi NIST. Chế độ hoạt động CBC tăng tính bảo mật bằng cách đưa yếu tố ngẫu nhiên (IV) vào mỗi phiên mã hóa. Mỗi bản mã phụ thuộc không chỉ vào dữ liệu gốc mà còn vào các khối trước đó, giúp ngăn chặn các tấn công lặp lại hoặc phân tích mẫu dữ liệu.
- **Thuật toán băm SHA-512:** Được sử dụng để kiểm tra toàn vẹn của dữ liệu mã hóa. Với độ dài kết quả lên tới 512 bit, SHA-512 gần như không thể bị đảo ngược hoặc tạo ra xung đột (collision) giả mạo, đảm bảo phát hiện mọi thay đổi dù là nhỏ nhất trong bản mã.

- **Chữ ký số RSA + SHA-512 (PKCS#1 v1.5):** Đảm bảo metadata (bao gồm tên file, timestamp, IP) được xác thực và không bị giả mạo. Việc ký số đảm bảo rằng chỉ có người gửi hợp lệ mới có thể tạo ra chữ ký hợp lệ cho metadata tương ứng, từ đó tăng tính tin cậy trong toàn bộ quy trình truyền file.
- **Cơ chế xác minh địa chỉ IP:** Là một lớp bảo vệ tiền tuyến – thực hiện ngay từ bước bắt tay (handshake). Việc kiểm tra IP giúp hệ thống chủ động từ chối các yêu cầu đến từ nguồn không xác định, giảm thiểu nguy cơ tấn công từ bên ngoài hoặc những truy cập trái phép từ thiết bị không thuộc hệ thống.

Thông qua việc tích hợp các thuật toán kể trên, hệ thống đáp ứng đồng thời cả ba yêu cầu quan trọng nhất trong bảo mật: **bảo mật dữ liệu, tính toàn vẹn và xác thực**. Cách thiết kế chia thành các giai đoạn rõ ràng (xác minh IP, trao đổi khóa, mã hóa, xác thực) giúp kiểm soát tốt tiến trình và dễ dàng mở rộng trong tương lai.

Tuy nhiên, hệ thống hiện tại vẫn mang tính chất mô phỏng và có một số điểm hạn chế, đặc biệt là trong khả năng ứng dụng thực tế – chẳng hạn độ dài khóa RSA, bảo vệ kết nối truyền hoặc khả năng quản lý khóa lâu dài.

3.1.2 Đề xuất cải tiến

Để nâng cấp hệ thống từ bản thử nghiệm thành giải pháp có thể áp dụng trong thực tế, cần cân nhắc các hướng cải tiến sau:

- **Tăng độ dài khóa RSA:** Sử dụng khóa 2048-bit hoặc cao hơn là bắt buộc trong hầu hết các tiêu chuẩn bảo mật hiện đại (FIPS, NIST, ISO/IEC 27001). Khi dùng RSA 2048-bit, hệ thống có thể kết hợp được với OAEP + SHA-512 đúng tiêu chuẩn mà không bị giới hạn kích thước bản rõ.
- **Thay thế AES-CBC bằng AES-GCM:** AES-GCM là chế độ mã hóa hiện đại hơn, tích hợp khả năng xác thực (Authenticated Encryption with Associated Data – AEAD). Điều này giúp đơn giản hóa luồng xử lý (không cần tính hash SHA-512 riêng), đồng thời đảm bảo tính toàn vẹn ở mức thuật toán mà không cần cơ chế kiểm tra phụ trợ.
- **Thay thế ký số PKCS#1 v1.5 bằng chuẩn RSA-PSS:** RSA-PSS là phương pháp ký số hiện đại hơn, hỗ trợ chống tấn công chọn bản mã (chosen-ciphertext attack), được khuyến nghị bởi nhiều tổ chức tiêu chuẩn hóa và được sử dụng trong các hệ thống như TLS 1.3.
- **Bổ sung mã hóa luồng truyền (TLS/SSL):** Việc truyền dữ liệu qua socket thô như hiện tại dễ bị nghe lén hoặc bị kẻ tấn công chen ngang. Bổ sung lớp bảo mật TLS giúp mã hóa toàn bộ luồng giao tiếp, nâng cao tính riêng tư và tránh tấn công Man-in-the-Middle (MitM).

- **Tích hợp xác thực hai yếu tố (2FA) hoặc mã token OTP:** Cơ chế IP tĩnh không phải lúc nào cũng đủ an toàn trong môi trường mạng thay đổi. Việc thêm lớp xác thực OTP hoặc xác thực bằng mã hóa đối xứng trước khi gửi có thể nâng cao độ tin cậy mà không ảnh hưởng lớn đến trải nghiệm người dùng.
- **Phân quyền người gửi/nhận qua mã định danh:** Thay vì chỉ xác thực IP, có thể cấp mã định danh cho từng ứng viên hoặc nhân viên, quản lý bằng cơ sở dữ liệu nội bộ, từ đó tăng độ kiểm soát và khả năng mở rộng hệ thống.
- **Xây dựng giao diện người dùng (GUI/Web):** Việc vận hành hệ thống thông qua dòng lệnh tương đối bất tiện với người không chuyên. Giao diện đơn giản (Tkinter hoặc web Flask) sẽ giúp mở rộng người dùng và cải thiện trải nghiệm.
- **Ghi log đầy đủ và có hệ thống:** Việc lưu lại thông tin mỗi phiên truyền (thời gian, IP, trạng thái xác thực) không chỉ hỗ trợ kiểm toán mà còn cần thiết khi có tranh chấp, sự cố bảo mật hoặc phân tích hành vi bất thường.
- **Tối ưu hiệu năng và kiểm thử tải:** Trong tình huống hệ thống tiếp nhận nhiều hồ sơ cùng lúc (ví dụ tuyển sinh, nộp hồ sơ online), cần thử nghiệm khả năng chịu tải và mở rộng bằng đa luồng hoặc hàng đợi xử lý.

Tổng kết lại, hệ thống hiện tại là một mô hình mẫu đầy đủ chức năng, dễ hiểu và đáp ứng yêu cầu học thuật, tuy nhiên để ứng dụng thực tế trong môi trường doanh nghiệp hoặc chính phủ, cần nâng cấp về cả mặt kỹ thuật và trải nghiệm người dùng. Việc kết hợp giữa các kỹ thuật mật mã hiện đại, xác thực định danh và giao diện thân thiện sẽ là hướng phát triển bền vững cho hệ thống này trong tương lai.

Tài liệu tham khảo

[1] Wikipedia contributors, *Block cipher mode of operation – Cipher Block Chaining (CBC)*, Wikipedia, The Free Encyclopedia. Đường dẫn: https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation#Cipher_block_chaining_(CBC)

[2] Lê Thị Thùy Trang, *Bài giảng Nhập môn An toàn, Bảo mật Thông tin*, Thuật toán AES, Trường Đại học Đại Nam. Đường dẫn: https://fitdnu.net/pluginfile.php/14289/mod_resource/content/2/Ba%CC%80i%206.%20Thua%CC%A3%CC%82t%20toa%CC%81n%20AES.pdf

[3] Lê Thị Thùy Trang, *Bài giảng Nhập môn An toàn, Bảo mật Thông tin*, Thuật toán RSA, Trường Đại học Đại Nam. Đường dẫn: https://fitdnu.net/pluginfile.php/14380/mod_resource/content/1/Ba%CC%80i%209.%20Ma%CC%83%20hoa%CC%81%20khoa%CC%81%20co%CC%82ng%20khai.pdf

[4] Wikipedia contributors, *Optimal asymmetric encryption padding*, Wikipedia, The Free Encyclopedia. Đường dẫn: https://en.wikipedia.org/wiki/Optimal_asymmetric_encryption_padding