BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC ĐẠI NAM

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO BÀI TẬP LỚN

NHẬP MÔN AN TOÀN BẢO MẬT THÔNG TIN

GỬI TÀI LIỆU PHÁP LÝ QUA NHIỀU SERVER TRUNG GIAN

Sinh viên thực hiện : **Nguyễn Hà Phương**
Nguyễn Thị Hiền
Bùi Hữu Tri Phương

Ngành : **Công nghệ thông tin**

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

Lời cảm ơn

Chúng em xin gửi lời cảm ơn chân thành đến ThS. Lê Thị Thùy Trang đã tận tình giảng dạy và truyền đạt kiến thức trong suốt quá trình học tập, tạo nền tảng vững chắc để chúng em thực hiện bài tập lớn này.

Mục lục

Danh sách hình vẽ

Danh sách giải thuật

Chương 1

Giới thiệu

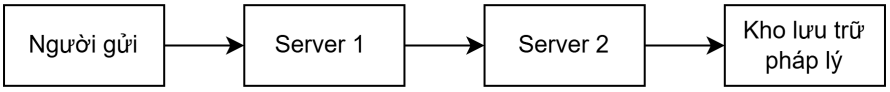
1.1 Đặt vấn đề

Trong kỷ nguyên số hóa, các hệ thống thông tin phân tán đóng vai trò trung tâm trong việc xử lý và lưu trữ khối lượng lớn dữ liệu. Đặc biệt, đối với lĩnh vực pháp lý – nơi các tài liệu, chứng cứ, hồ sơ vụ án và hợp đồng mang tính chất nhạy cảm và có giá trị pháp lý cao – việc truyền tải và lưu trữ an toàn các văn bản này trở thành một yêu cầu tất yếu. Sự phát triển của hạ tầng công nghệ cùng nhu cầu kết nối giữa các tổ chức pháp lý, tòa án, văn phòng luật sư và kho lưu trữ đã làm xuất hiện những mô hình truyền dữ liệu qua nhiều server trung gian. Mục đích của các server trung gian có thể bao gồm cân bằng tải, tăng cường khả năng dự phòng, hoặc phân tán dữ liệu nhằm nâng cao tính sẵn sàng. Tuy nhiên, mô hình này cũng đồng thời làm gia tăng nguy cơ rò rỉ, mất mát hoặc bị tấn công giả mạo dữ liệu trong quá trình truyền tải.

Trên thực tế, các vụ việc rò rỉ thông tin pháp lý không chỉ gây thiệt hại về uy tín mà còn dẫn đến hậu quả pháp lý nghiêm trọng, làm suy giảm niềm tin của tổ chức và cá nhân vào hệ thống số hóa. Bên cạnh đó, các yêu cầu tuân thủ khắt khe trong ngành, như luật bảo mật dữ liệu (GDPR, HIPAA hoặc các quy định trong nước), càng nhấn mạnh tầm quan trọng của việc xây dựng quy trình truyền tải dữ liệu an toàn. Đặc biệt, việc áp dụng các phương pháp mã hóa, xác thực và kiểm chứng toàn vẹn dữ liệu cần được nghiên cứu và triển khai một cách bài bản để phù hợp với kiến trúc phân tán đa tầng, trong đó dữ liệu phải đi qua nhiều nút trung gian trước khi đến đích.

Đề tài này tập trung nghiên cứu và xây dựng một quy trình truyền tải tệp tin nhạy cảm, từ một chuyên viên IT đến kho lưu trữ pháp lý đích thông qua hai server trung gian. Quy trình này nhằm đảm bảo ba yếu tố cốt lõi: (1) tính bảo mật: ngăn chặn việc truy cập trái phép thông tin trong suốt quá trình truyền; (2) tính toàn vẹn: phát hiện kịp thời mọi thay đổi, chỉnh sửa hoặc giả mạo dữ liệu; (3) khả năng xác thực: đảm bảo nguồn gốc và danh tính của bên gửi và bên

nhận. Phương pháp nghiên cứu tập trung áp dụng kết hợp kỹ thuật mã hóa khóa đối xứng và bất đối xứng, hàm băm kiểm tra toàn vẹn, cùng cơ chế chứng thực số để xây dựng giải pháp có tính khả thi và hiệu quả trong môi trường thực tế.



Hình 1.1: Sơ đồ truyền file

Bài toán có các yêu cầu về bảo mật:

- Đảm bảo tính liên tục và sẵn sàng trong truyền tải dữ liệu trong môi trường phân tán.
- Bảo mật nội dung tài liệu pháp lý thông qua mã hóa.
- Xác thực nguồn gốc tài liệu và đảm bảo không bị thay đổi thông qua ký số.
- Kiểm tra tính toàn vẹn của tài liệu trong suốt quá trình truyền dẫn.

1.2 Mục tiêu của đề tài

Các mục tiêu chính của đề tài bao gồm:

- **Mã hóa (Encryption)**

  Thuật toán: DES (Data Encryption Standard)

  Mục đích: bảo vệ nội dung tài liệu khỏi bị truy cập trái phép.

  Cơ chế: sử dụng thuật toán DES (Data Encryption Standard) để mã hóa tệp legal_doc.txt trước khi truyền đi.

- **Chữ ký số (Authentication and Integrity)**

  Thuật toán: RSA với băm SHA-512

  Mục đích:Xác thực người gửi (ai gửi file) và đảm bảo tính toàn vẹn (nội dung không bị sửa đổi trên đường truyền).

  Cơ chế: Băm tài liệu sử dụng SHA-512 để tạo ra giá trị băm duy nhất. Dùng RSA với khóa riêng để ký lên giá trị băm, tạo thành chữ ký số. Người nhận có thể xác thực chữ ký bằng khóa công khai và kiểm tra tính toàn vẹn của nội dung.

- **Kiểm tra toàn vẹn (Integrity check)**

Dùng SHA-512 để tạo thông điệp băm (message digest), sau đó ký bằng RSA → giúp phát hiện nếu dữ liệu bị thay đổi.

Đề tài tập trung vào việc truyền tải một tệp tin văn bản nhạy cảm qua hai server trung gian. Đề tài không đi sâu vào vấn đề lưu trữ dữ liệu lâu dài, không xử lý trường hợp tệp bị chia nhỏ thành nhiều gói để truyền song song, và cũng không mở rộng sang các loại dữ liệu phi cấu trúc (ví dụ: hình ảnh, video). Tuy vậy, phương pháp và quy trình xây dựng hoàn toàn có thể mở rộng để áp dụng cho các trường hợp truyền tải nhiều tệp hoặc dữ liệu có cấu trúc phức tạp hơn.

1.3 Cấu trúc của báo cáo

Báo cáo gồm các phần như sau:

- Chương 1: Giới thiệu.
- Chương 2: Phân tích yêu cầu và thiết kế ứng dụng
- Chương 3: Kết luận và hướng phát triển

Chương 2

Phân tích yêu cầu và thiết kế ứng dụng

2.1 Mô tả thuật toán

2.1.1 Handshake

Giao thức bắt tay (Handshake) là một giai đoạn khởi tạo quan trọng trong truyền thông mạng, đặc biệt là trong các giao thức bảo mật như SSL/TLS (Secure Sockets Layer/Transport Layer Security). Về cơ bản, bắt tay SSL/TLS là một "cuộc trò chuyện" ban đầu giữa hai bên – thường là máy khách (ví dụ: trình duyệt web của người dùng) và máy chủ (ví dụ: máy chủ web) – nhằm thiết lập một kênh liên lạc an toàn và được mã hóa. Mục đích chính của quá trình này là để các bên xác thực danh tính của nhau, đồng ý về các thuật toán mật mã sẽ được sử dụng, và tạo ra các khóa phiên bí mật dùng chung để mã hóa dữ liệu sau đó. Quá trình bắt tay này diễn ra mỗi khi người dùng truy cập một trang web qua HTTPS, hoặc bất cứ khi nào có giao tiếp khác sử dụng HTTPS, bao gồm các cuộc gọi API và truy vấn DNS qua HTTPS. Nếu quá trình bắt tay không thành công, kết nối an toàn sẽ không được thiết lập, dẫn đến lỗi như "SSL handshake failed".

Điều quan trọng là phải phân biệt "giao thức bắt tay" trong ngữ cảnh bảo mật mạng (như SSL/TLS) với "Handshake (HNS)" là một giao thức đặt tên miền ngang hàng phi tập trung. HNS được thiết kế để phục vụ như một chuỗi DNS thay thế, cho phép tạo các miền cấp cao nhất (TLD) mới mà không phụ thuộc vào Tập đoàn Internet cấp số và tên miền (ICANN). Mặc dù HNS cũng liên quan đến giao thức mạng, nhưng chức năng và mục đích của nó hoàn toàn khác biệt so với giao thức bắt tay mật mã được thảo luận trong báo cáo này. Để tránh nhầm lẫn, báo cáo này tập trung vào lý thuyết và hoạt động của giao thức bắt tay trong ngữ cảnh thiết lập các kênh truyền thông an toàn và mã hóa.

Quy trình Bắt tay TLS:

1. **Client Hello**: Quá trình bắt tay bắt đầu khi máy khách gửi một thông điệp "Client Hello"

tới máy chủ. Thông điệp này bao gồm phiên bản TLS cao nhất mà máy khách hỗ trợ, một danh sách các bộ mã hóa (cipher suites) mà máy khách có thể sử dụng (được xếp hạng theo thứ tự ưu tiên), và một chuỗi byte ngẫu nhiên được tạo ra bởi máy khách (gọi là "client random"). Thông điệp này cũng có thể bao gồm các phương thức nén dữ liệu được hỗ trợ.

2. **Server Hello**: Máy chủ phản hồi bằng thông điệp "Server Hello". Trong thông điệp này, máy chủ chọn phiên bản TLS và bộ mã hóa từ danh sách mà máy khách đã cung cấp, gửi một ID phiên, và một chuỗi byte ngẫu nhiên khác ("server random"). Đồng thời, máy chủ gửi chứng chỉ số SSL/TLS của mình cho máy khách. Nếu máy chủ yêu cầu xác thực máy khách, nó cũng sẽ gửi một yêu cầu chứng chỉ máy khách.

3. **Xác thực Chứng chỉ Máy chủ**: Máy khách nhận chứng chỉ số của máy chủ và tiến hành xác minh nó với tổ chức cấp chứng chỉ (CA) đã phát hành. Bước này đảm bảo rằng máy chủ là thực thể mà nó tuyên bố, và máy khách đang giao tiếp với chủ sở hữu hợp pháp của miền.

4. **Trao đổi Khóa (Key Exchange)**: Đây là một bước quan trọng để thiết lập khóa bí mật chung. Máy khách gửi một chuỗi byte ngẫu nhiên khác, gọi là "premaster secret", được mã hóa bằng khóa công khai của máy chủ (thu được từ chứng chỉ SSL của máy chủ). Chỉ có máy chủ mới có thể giải mã "premaster secret" này bằng khóa riêng tư tương ứng của mình. Sau khi cả hai bên có "premaster secret" cùng với "client random" và "server random", cả máy khách và máy chủ độc lập tính toán và tạo ra các khóa phiên đối xứng (session keys) dùng chung.

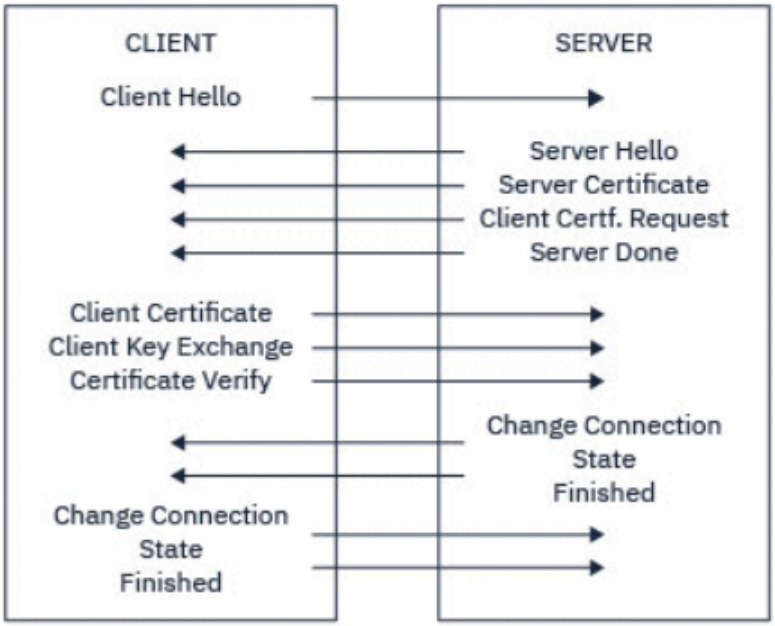
5. **Xác thực Máy khách (Tùy chọn)**: Nếu máy chủ đã yêu cầu chứng chỉ máy khách, máy khách sẽ gửi chứng chỉ số của mình cùng với một chuỗi byte ngẫu nhiên được mã hóa bằng khóa riêng tư của nó. Máy chủ sau đó xác minh chứng chỉ này để xác thực danh tính của máy khách.

6. **Thông điệp Finished**: Để hoàn tất quá trình bắt tay, cả máy khách và máy chủ gửi các thông điệp "Finished" được mã hóa bằng các khóa phiên đối xứng mới tạo. Những thông điệp này xác nhận rằng quá trình bắt tay đã hoàn tất và các bên đã sẵn sàng cho giao tiếp được mã hóa đối xứng.

7. **Giao tiếp An toàn**: Sau khi bắt tay hoàn tất thành công, máy khách và máy chủ có thể bắt đầu trao đổi dữ liệu ứng dụng một cách an toàn. Tất cả các thông điệp tiếp theo sẽ được mã hóa đối xứng bằng các khóa phiên dùng chung đã được thiết lập trong quá trình bắt tay.

Các cơ chế trao đổi khóa trong quá trình bắt tay

Trong quá trình bắt tay TLS, việc trao đổi khóa là một bước thiết yếu để thiết lập một khóa bí mật chung, sau đó sẽ được sử dụng cho mã hóa đối xứng hiệu quả hơn. Hai cơ chế trao đổi



Hình 2.1: Quy trình bắt tay TLS.

khóa phổ biến là RSA và Diffie-Hellman (DH).

- **Trao đổi khóa RSA**: Trong các phiên bản TLS cũ hơn (trước TLS 1.3), RSA là một phương pháp trao đổi khóa phổ biến. Trong phương pháp này, máy chủ gửi chứng chỉ số của mình chứa khóa công khai RSA. Máy khách xác thực chứng chỉ này và sử dụng khóa công khai của máy chủ để mã hóa "premaster secret" (một chuỗi byte ngẫu nhiên) và gửi nó cho máy chủ. Máy chủ sau đó giải mã "premaster secret" bằng khóa riêng tư của mình. Việc máy chủ có thể giải mã thành công thông điệp này ngụ ý xác thực danh tính của máy chủ.

- **Trao đổi khóa Diffie-Hellman (DH) / Ephemeral Diffie-Hellman (DHE)**: Cơ chế DH hoạt động khác biệt; thay vì một bên mã hóa một bí mật và gửi nó, cả máy khách và máy chủ trao đổi các tham số công khai và độc lập tính toán "premaster secret" chung. Một lợi thế đáng kể của DHE là nó cung cấp "bảo mật chuyển tiếp" (Forward Secrecy). Điều này có nghĩa là ngay cả khi khóa riêng tư dài hạn của máy chủ bị lộ trong tương lai, các phiên giao tiếp cũ đã được mã hóa bằng khóa phiên được tạo qua DHE vẫn an toàn và không thể bị giải mã. Điều này là do khóa phiên không được mã hóa trực tiếp bằng khóa riêng tư của máy chủ mà được tạo ra từ các tham số tạm thời chỉ tồn tại trong suốt phiên giao tiếp đó.

Xác thực và Đàm phán Bộ mã hóa (Cipher Suite)

Bộ mã hóa (Cipher Suite) là một tập hợp các thuật toán mật mã được sử dụng để thiết lập một kết nối an toàn giữa máy khách và máy chủ. Một bộ mã hóa điển hình bao gồm:

- Thuật toán trao đổi khóa (ví dụ: RSA, Diffie-Hellman).

- Thuật toán xác thực (ví dụ: RSA, DSA).
- Thuật toán mã hóa đối xứng (ví dụ: AES, DES) để bảo mật dữ liệu phiên.
- Hàm băm (ví dụ: SHA) để kiểm tra tính toàn vẹn của dữ liệu.

Trong quá trình bắt tay, máy khách gửi một danh sách các bộ mã hóa mà nó hỗ trợ, theo thứ tự ưu tiên. Máy chủ sau đó xem xét danh sách này và chọn một bộ mã hóa mà cả hai bên đều hỗ trợ và đồng ý sử dụng cho phiên giao tiếp. Quá trình đàm phán này đảm bảo rằng các bên sử dụng một tập hợp các thuật toán tương thích và an toàn cho kết nối của họ.

2.1.2 Mã hóa đối xứng DES

Tiêu chuẩn mã hóa dữ liệu DES (Data Encryption Standard) là một phương pháp mật mã hóa được FIPS (Federal Information Processing Standard: Tiêu chuẩn xử lý thông tin Liên bang Hoa Kỳ) chọn làm chuẩn chính thức vào năm 1976. Thuật toán mã hóa theo tiêu chuẩn DES gọi là DEA (Data Encryption Algorithm). (Người ta cũng thường gọi lẫn lộn DEA và DES trong khi sử dụng). DES là một mã khối, mỗi khối gồm 64 bit trong đó dành 8 bit để kiểm tra lỗi (Parity checking) còn lại 56 bit khóa.

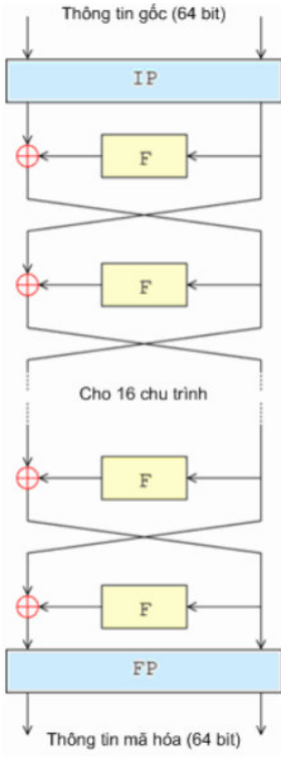
Giải thuật 1: Thuật toán DES (Data Encryption Standard)

Input: Bản rõ **P**, khóa **K** (56 bit)
Output: Bản mã **C**

```
1:  function DES(P, K)
2:      Thực hiện hoán vị ban đầu (Initial Permutation) trên P
3:      Chia P thành hai nửa trái (L) và phải (R)
4:      for 16 vòng lặp do
5:          Sinh khóa con K_i từ K
6:          Thực hiện hàm F với R và K_i, rồi XOR với L
7:          Hoán đổi L và R
8:      end for
9:      Gộp L và R, thực hiện hoán vị cuối (Final Permutation)
10:     return C
11: end function
```

Mô tả thuật toán DES

Có 16 chu trình giống nhau trong quá trình xử lý. Ngoài ra còn có hai lần hoán vị đầu và cuối (*Initial and final permutation:* IP and FP). Hai quá trình này có tính chất đối nhau (trong quá trình mã hóa thì IP trước FP, khi giải mã thì ngược lại). IP và FP, được sử dụng từ thập niên 1970, không có vai trò xét về mật mã học và việc sử dụng chúng chỉ có ý nghĩa đáp ứng cho quá trình đưa thông tin vào và lấy thông tin ra từ các khối phần cứng.



Hình 2.2: Mô tả thuật toán DES.

Trước khi đi vào 16 chu trình chính, khối thông tin 64 bit được tách làm hai phần 32 bit và mỗi phần sẽ được xử lý tuần tự (quá trình này còn được gọi là mạng Feistel). Cấu trúc của thuật toán (mạng Feistel) đảm bảo rằng quá trình mã hóa và giải mã diễn ra tương tự. Điểm khác nhau chỉ ở chỗ các khóa con được sử dụng theo trình tự ngược nhau. Điều này giúp cho việc thực hiện thuật toán trở nên đơn giản, đặc biệt là khi thực hiện bằng phần cứng.

Ký hiệu (+) thể hiện phép toán XOR (hàm "tuyển ngặt": Exclusive OR) hay là hàm "cộng theo modulo 2". Hàm F làm biến đổi một khối 32 bit đang xử lý với một khóa con.

Đầu ra sau hàm F được kết hợp khối 32 bit còn lại và hai phần được tráo đổi để xử lý trong chu trình kế tiếp. Sau chu trình cuối cùng thì 2 nửa không bị tráo đổi; đây là đặc điểm của cấu trúc Feistel khiến cho quá trình mã hóa và giải mã trở nên giống nhau.

Hàm Feistel (F)

Hàm F, như được miêu tả như hình 2.3, hoạt động trên khối 32 bit và bao gồm bốn giai đoạn:

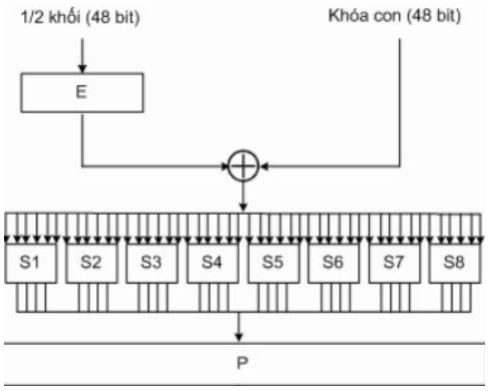
1. Mở rộng: 32 bit đầu vào được mở rộng thành 48 bit sử dụng thuật toán hoán vị mở rộng (expansion permutation) với việc nhân đôi một số bit. Giai đoạn này được ký hiệu là E trong sơ đồ.

2. Trộn khóa: 48 bit thu được sau quá trình mở rộng được XOR với khóa con. Mười sáu khóa con 48 bit được tạo ra từ khóa chính 56 bit theo một chu trình tạo khóa con (key schedule)

miêu tả ở phần sau.

3. Thay thế: 48 bit sau khi trộn được chia làm 8 khối con 6 bit và được xử lý qua hộp thay thế S-box. Đầu ra của mỗi khối 6 bit là một khối 4 bit theo một chuyển đổi phi tuyến được thực hiện bằng một bảng tra. Khối S-box đảm bảo phần quan trọng cho độ an toàn của DES. Nếu không có S-box thì quá trình sẽ là tuyến tính và việc thám mã sẽ rất đơn giản.

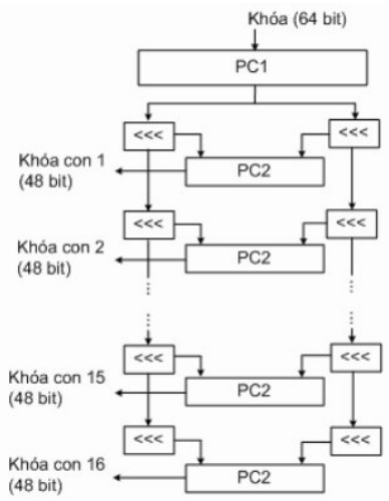
4. Hoán vị: Cuối cùng, 32 bit thu được sau S-box sẽ được sắp xếp lại theo một thứ tự cho trước (còn gọi là P-box).



Hình 2.3: Hàm F trong DES.

Quá trình luân phiên sử dụng S-box và sự hoán vị các bit cũng như quá trình mở rộng đã thực hiện được tính chất gọi là sự xáo trộn và khuếch tán (confusion and diffusion). Đây là yêu cầu cần có của một thuật toán mã hóa được Claude Shannon phát hiện trong những năm 1940.

Quá trình tạo khóa con (Sub-key)



Hình 2.4: Quá trình tạo khóa con trong DES.

Hình 2.3 mô tả thuật toán tạo khóa con cho các chu trình. Đầu tiên, từ 64 bit ban đầu của khóa, 56 bit được chọn (Permuted Choice 1, hay PC-1); 8 bit còn lại bị loại bỏ. 56 bit thu được

được chia làm hai phần bằng nhau, mỗi phần được xử lý độc lập. Sau mỗi chu trình, mỗi phần được dịch đi 1 hoặc 2 bit (tùy thuộc từng chu trình). Các khóa con 48 bit được tạo thành bởi thuật toán lựa chọn 2 (Permuted Choice 2, hay PC-2) gồm 24 bit từ mỗi phần. Quá trình dịch chuyển bit (được ký hiệu là "<<<" trong sơ đồ) khiến cho các khóa con sử dụng các bit khác nhau của khóa chính; mỗi bit được sử dụng trung bình là 14 trong tổng số 16 khóa con.

Quá trình tạo khóa con khi thực hiện giải mã cũng diễn ra tương tự nhưng các khóa con được tạo theo thứ tự ngược lại. Ngoài ra sau mỗi chu trình, khóa sẽ được dịch chuyển phải thay vì dịch chuyển trái như khi mã hóa.

Mặc dù Data Encryption Standard (DES) từng được xem là chuẩn mã hóa đối xứng phổ biến và được tin cậy trong nhiều thập kỷ, sự phát triển nhanh chóng của năng lực tính toán đã khiến DES dần trở nên không còn đủ an toàn. Cụ thể, kích thước khóa ngắn 56 bit của DES khiến thuật toán này dễ bị tấn công vét cạn (*brute-force*) trong thời gian thực tế có thể chấp nhận được. Để khắc phục hạn chế đó mà vẫn tận dụng kiến trúc DES sẵn có, các biến thể mở rộng như Double DES đã được đề xuất.

Double DES (DESx2) áp dụng hai lần thuật toán DES nối tiếp với hai khóa độc lập khác nhau. Cơ chế mã hóa được mô tả như sau:

$$C = E_{K_2}\big(E_{K_1}(P)\big) \tag{2.1}$$

Trong đó:

- P: bản rõ (*plaintext*)
- K_1, K_2: hai khóa DES độc lập (mỗi khóa dài 56 bit)
- $E_K(M)$: hàm mã hóa DES của thông điệp M với khóa K
- C: bản mã (*ciphertext*)

Ý tưởng ban đầu của Double DES là nhằm tăng độ dài khóa từ 56 bit lên 112 bit (vì có hai khóa). Tuy nhiên, Double DES không thực sự an toàn gấp đôi như mong đợi do tồn tại tấn công *meet-in-the-middle* (MITM). Với MITM, kẻ tấn công chỉ cần 2^{56+1} phép tính thay vì 2^{112}, làm giảm đáng kể hiệu quả bảo mật. Vì vậy, trên thực tế, Double DES không được sử dụng rộng rãi.

Để giải quyết nhược điểm của Double DES, Triple DES (3DES) áp dụng ba lần DES thay vì hai lần, kết hợp ba khóa khác nhau. Phổ biến nhất là cơ chế mã hóa – giải mã – mã hóa (EDE):

$$C = E_{K_3}\big(D_{K_2}(E_{K_1}(P))\big) \tag{2.2}$$

Trong đó:

- K_1, K_2, K_3: ba khóa DES, mỗi khóa dài 56 bit
- E: hàm mã hóa DES
- D: hàm giải mã DES

Triple DES có thể hoạt động ở các chế độ khác nhau:

1. Ba khóa khác nhau ($K_1 \neq K_2 \neq K_3$): độ dài khóa hiệu dụng 168 bit.
2. Hai khóa ($K_1 = K_3 \neq K_2$): độ dài khóa hiệu dụng 112 bit.
3. Một khóa ($K_1 = K_2 = K_3$): thực chất chỉ là DES, để tương thích ngược.

Triple DES khắc phục được tấn công meet-in-the-middle, nhờ độ phức tạp tính toán lớn hơn nhiều, và vẫn dựa trên nền tảng thuật toán DES vốn đã được kiểm chứng rộng rãi. Tuy nhiên, nhược điểm lớn nhất của Triple DES là hiệu năng kém so với các thuật toán hiện đại (như AES), do phải thực hiện ba lần mã hóa và/hoặc giải mã, dẫn đến tốc độ chậm và tiêu tốn nhiều tài nguyên tính toán..

2.1.3 Mã hóa bất đối xứng RSA

Mã hóa bất đối xứng (asymmetric encryption), đôi khi được gọi là mã hóa khóa công khai sử dụng một cặp khóa cho quá trình mã hóa và giải mã. Trong cặp khóa, khóa công khai được sử dụng cho mã hóa và khóa riêng được sử dụng cho giải mã. Chỉ khóa riêng cần giữ bí mật, còn khóa công khai có thể phổ biến rộng rãi, nhưng phải đảm bảo tính toàn vẹn và xác thực chủ thể của khóa.

Quá trình mã hóa (Encrypt) và giải mã (Decrypt) sử dụng mã hóa khóa bất đối xứng. Theo đó, người gửi (Sender) sử dụng khóa công khai (Public key) của người nhận (Recipient) để mã hóa bản rõ (Plaintext) thành bản mã (Ciphertext) và gửi nó cho người nhận. Người nhận nhận được bản mã sử dụng khóa riêng (Private key) của mình để giải mã khôi phục bản rõ.

Đặc điểm nổi bật của các hệ mã hóa khóa bất đối xứng là kích thước khóa lớn, có thể lên đến hàng ngàn bit. Do vậy, các hệ mã hóa dạng này thường có tốc độ thực thi chậm hơn nhiều lần so với các hệ mã hóa khóa đối xứng có độ an toàn tương đương. Mặc dù vậy, các hệ mã hóa khóa bất đối xứng có khả năng đạt độ an toàn cao và ưu điểm nổi bật nhất là việc quản lý và phân phối khóa đơn giản hơn do khóa công khai có thể phân phối rộng rãi.

Các giải thuật mã hóa khóa bất đối xứng điển hình bao gồm: RSA, Rabin, ElGamal, McEliece và Knapsack.

RSA là một hệ mật mã bất đối xứng do ba nhà toán học Rivest–Shamir–Adleman công bố năm 1977. Thuật toán này dựa trên tính chất toán học của bài toán phân tích một số nguyên lớn

thành các thừa số nguyên tố – một bài toán được coi là khó giải trong thời gian thực tế. Nhờ đó, RSA cho phép tạo ra một cặp khóa: khóa công khai để mã hóa và khóa bí mật để giải mã, đảm bảo tính bảo mật mà không cần chia sẻ khóa bí mật qua kênh không an toàn. Với đặc điểm này, RSA đã trở thành nền tảng quan trọng trong nhiều ứng dụng như chữ ký số, giao thức bảo mật SSL/TLS và xác thực điện tử, góp phần to lớn vào sự phát triển của an ninh mạng hiện đại.

Giải thuật 2: Thuật toán RSA (Rivest-Shamir-Adleman)

Input: Bản rõ **M**, cặp khóa công khai (e, n), khóa bí mật d

Output: Bản mã **C** hoặc bản rõ **M** sau giải mã

1: **function** RSA_ENCRYPT(M, e, n)
2: $C \leftarrow M^e \bmod n$
3: **return** C
4: **end function**
5: **function** RSA_DECRYPT(C, d, n)
6: $M \leftarrow C^d \bmod n$
7: **return** M
8: **end function**

Sử dụng hai khóa:

- Khóa công khai: dùng để mã hóa hoặc xác minh.
- Khóa bí mật: dùng để giải mã hoặc ký số.

Tạo cặp khóa RSA

- Sinh 2 số nguyên tố lớn: p, q
- Tính:

$$n = p \times q \quad (modulus) \quad \phi(n) = (p-1)(q-1)$$

- Chọn số nguyên e sao cho :

$$1 < e < \phi(n), \quad \gcd(e, \phi(n)) = 1$$

- Tính:

$$d \equiv e^{-1} \pmod{\phi(n)}$$

Cặp khóa

- Công khai: *(e,n)*
- Bí mật: *(d,n)*

Ưu và nhược điểm của mã RSA

Thuật toán RSA thực hiện một dãy phép tính lũy thừa modulo khá lớn.

Độ phức tạp tính toán

Khóa công khai = O(k2) bước tính toán, Khóa riêng = O(k3), Tổng quát mã RSA có độ phức tạp tính toán là O(k4) – k là số bit của modulo. Vì vậy mã RSA có nhược điểm đầu tiên là tốc độ lập mã và giải mã rất chậm.

Tuy nhiên mã RSA có độ bảo mật cao: hầu như không có thuật toán giải tổng quát mà phải dò thử dần (tấn công bạo lực). Nếu chọn P, Q lớn thì kết quả từ chỗ biết số mũ lập mã E, tìm ngược lại số mũ giải mã D rất phức tạp hầu như không làm được trong thời gian thực. Chẳng hạn ta tạo một khóa mã để mã hóa thông tin cho các thẻ tín dụng chỉ cho phép sử dụng trong 2 năm. Nếu khả năng bị phá khóa là trong thời gian 1000 năm hay lâu hơn nữa thì trong thực tế có thể xem là an toàn.

Một nhược điểm lớn khác của mã RSA là nguy cơ về "tính tin cậy". Khi B dùng khóa công khai nhận từ A để gửi tin, chắc chắn chỉ A đọc được: tin cậy phía người gửi tin. Khi A nhận tin, chưa chắc do B gửi (vì khóa công khai có thể lộ và người thứ ba biết khóa công khai, có thể dùng để mã hóa những thông điệp giả gửi cho A): không tin cậy phía người nhận tin.

Để khắc phục điều đó, phải có phương pháp "phân phối khóa công khai" một cách tin cậy hơn. Trong trường hợp chỉ có 2 đối tác trao đổi với nhau, người ta có thể dùng sơ đồ trao đổi khóa công khai để đảm bảo an toàn và tin cậy cho cả hai phía gửi và nhận tin.

Sơ đồ trao đổi khóa công khai

- A tạo một cặp khóa, khóa công khai (của A) là E_1 cho B và khóa riêng D_1 giữ cho mình.

- B tạo khóa riêng D_2, khóa công khai E_2(của B).

- Dùng E_1 nhận được của A để mã hóa E_2 : $E_1(E_2) = E_2'$, B gửi E_2'cho A và giữ D_2cho riêng mình.

- A nhận được E_2', giải mã bằng D_1 (Chỉ mình A có D_1): Chỉ có A đọc được E_2. Khi đó chỉ có 2 đối tác A và B cùng sở hữu khóa công khai E_2.

- A có thông điệp gốc P, dùng E_2(của B mã hóa thông điệp: $E_2(P)$ = C, gửi thông điệp mã hóa (bằng khóa công khai của B) cho B chắc chắn chỉ có B đọc được.

- B: nhận chắc chắn do A gửi, đọc: $D_2(C)$ = P.

Sử dụng sơ đồ trao đổi khóa công khai, chúng ta tạo được sự tin cậy cả cho hai phía người gửi tin và người nhận tin. Nhưng mặt khác độ phức tạp tính toán tăng lên và tốc độ lập mã, giải

mã càng chậm!

Mức độ an toàn

Về khía cạnh an toàn, các thuật toán mật mã hóa khóa bất đối xứng cũng không khác nhiều với các thuật toán mã hóa khóa đối xứng. Có những thuật toán được dùng rộng rãi, có thuật toán chủ yếu trên lý thuyết; có thuật toán vẫn còn được xem là an toàn, có thuật toán đã bị phá vỡ. Cũng cần lưu ý là những thuật toán được dùng rộng rãi không phải lúc nào cũng đảm bảo an toàn. Một số thuật toán có những chứng minh về độ an toàn với những tiêu chuẩn khác nhau. Nhiều chứng minh gắn việc phá vỡ thuật toán với những bài toán nổi tiếng vẫn được cho là không có lời giải trong thời gian đa thức. Nhìn chung, chưa có thuật toán nào được chứng minh là an toàn tuyệt đối. Vì vậy, cũng giống như tất cả các thuật toán mật mã nói chung, các thuật toán mã hóa khóa công khai cần phải được sử dụng một cách thận trọng.

2.1.4 Hàm băm SHA

Định nghĩa và Vai trò của Hàm băm mật mã

Hàm băm mật mã (Cryptographic Hash Function) là một thuật toán thiết yếu trong an ninh mạng, có khả năng chuyển đổi bất kỳ dữ liệu đầu vào nào (tin nhắn) thành một chuỗi bit có độ dài cố định, được gọi là "message digest" hoặc "hash value". Đặc tính nổi bật của hàm băm mật mã là tính "một chiều" của nó: việc đảo ngược quá trình để tìm lại dữ liệu gốc từ giá trị băm là không khả thi về mặt tính toán. Điều này có nghĩa là từ một giá trị băm, không thể tái tạo lại tin nhắn gốc.

Vai trò của hàm băm mật mã rất đa dạng và quan trọng: **Đảm bảo tính toàn vẹn dữ liệu**: Hàm băm được sử dụng để tạo checksums, giúp xác minh tính toàn vẹn của tệp hoặc khối dữ liệu. Nếu một thay đổi nhỏ nhất được thực hiện đối với dữ liệu gốc, giá trị băm tạo ra sẽ hoàn toàn khác biệt, cho phép phát hiện ngay lập tức bất kỳ sự giả mạo hoặc thay đổi nào trong quá trình truyền tải hoặc lưu trữ.

Tạo chữ ký số: Hàm băm là một thành phần không thể thiếu của chữ ký số. Nó tạo ra một bản tóm tắt duy nhất của tin nhắn, sau đó được mã hóa bằng khóa riêng tư của người gửi.

Lưu trữ mật khẩu an toàn: Thay vì lưu trữ mật khẩu gốc trong cơ sở dữ liệu, các hệ thống thường lưu trữ giá trị băm của mật khẩu. Điều này bảo vệ mật khẩu khỏi bị lộ nếu cơ sở dữ liệu bị tấn công, vì kẻ tấn công sẽ chỉ tìm thấy các giá trị băm chứ không phải mật khẩu thực.

Trong công nghệ blockchain: Hàm băm mật mã, đặc biệt là SHA-256, đóng vai trò bảo mật các khối thông tin và kết nối chúng lại với nhau trong chuỗi khối. *Các thuộc tính thiết yếu của Hàm băm an toàn*

Để được coi là một hàm băm mật mã an toàn, thuật toán phải đáp ứng một số thuộc tính quan

trọng:

- **Kháng tiền ảnh (Pre-image Resistance)**: Với một giá trị băm h(m) cho trước, việc tìm ra tin nhắn gốc m tương ứng là cực kỳ khó khăn về mặt tính toán. Thuộc tính này bảo vệ chống lại các cuộc tấn công vét cạn.
- **Kháng tiền ảnh thứ hai (Second Pre-image Resistance)**: Với một tin nhắn m cho trước, việc tìm một tin nhắn m' khác ($m' \neq m$) nhưng tạo ra cùng một giá trị băm (h(m') = h(m)) là cực kỳ khó khăn.
- **Kháng va chạm (Collision Resistance)**: Việc tìm ra hai tin nhắn m1 và m2 khác nhau ($m1 \neq m2$) nhưng tạo ra cùng một giá trị băm (h(m1) = h(m2)) là cực kỳ khó khăn.
- **Hiệu ứng tuyết lở (Avalanche Effect)**: Một thay đổi rất nhỏ trong dữ liệu đầu vào (ví dụ: thay đổi một bit) sẽ tạo ra một thay đổi lớn và không thể đoán trước trong giá trị băm đầu ra. Điều này đảm bảo rằng giá trị băm không tiết lộ bất kỳ thông tin nào về cấu trúc hoặc nội dung của dữ liệu đầu vào.

Tổng quan về họ SHA

Giải thuật 3: Thuật toán băm SHA (Secure Hash Algorithm)

Input: Thông điệp đầu vào **M**
Output: Giá trị băm **H**

```
1:  function SHA(M)
2:      Bổ sung đệm (padding) để M có độ dài phù hợp
3:      Chia M thành các khối có kích thước cố định
4:      Khởi tạo giá trị băm ban đầu
5:      for mỗi khối do
6:          Áp dụng các phép toán logic, dịch vòng, cộng mô-đun, v.v.
7:          Cập nhật giá trị băm
8:      end for
9:      return H
10: end function
```

Secure Hash Algorithms (SHA) là một họ các hàm băm mật mã được Viện Tiêu chuẩn và Công nghệ Quốc gia Hoa Kỳ (NIST) công bố như một Tiêu chuẩn Xử lý Thông tin Liên bang (FIPS).

SHA-0 và SHA-1:

- Lịch sử: SHA-0 là phiên bản đầu tiên của thuật toán băm an toàn, được NIST công bố vào năm 1993. Do một lỗ hổng lớn được phát hiện, SHA-0 nhanh chóng bị rút lại và được thay

thế bằng SHA-1 vào năm 1995. SHA-1 tạo ra một giá trị băm 160-bit.

- Tuy nhiên, kể từ năm 2005, SHA-1 đã không còn được coi là an toàn trước các đối thủ có nguồn lực tốt. Nó đã được chứng minh là dễ bị tấn công va chạm (collision attacks) và tấn công mở rộng độ dài (length-extension attacks).
- Tấn công SHAttered (2017): Vào tháng 2 năm 2017, CWI Amsterdam và Google đã công bố cuộc tấn công SHAttered, tạo ra hai tệp PDF khác nhau nhưng có cùng giá trị băm SHA-1. Đây là cuộc tấn công va chạm công khai đầu tiên chống lại SHA-1, chứng minh rõ ràng tính không an toàn của nó cho các ứng dụng yêu cầu kháng va chạm.
- Do những lỗ hổng này, NIST đã chính thức loại bỏ SHA-1 vào năm 2011 và cấm sử dụng nó cho chữ ký số từ cuối năm 2013. NIST khuyến nghị các cơ quan liên bang ngừng sử dụng SHA-1 cho tất cả các ứng dụng càng sớm càng tốt và đặt mục tiêu loại bỏ hoàn toàn vào năm 2030.

Họ SHA-2 (SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256):

- SHA-2 được phát triển ngay sau khi phát hiện các cuộc tấn công vét cạn hiệu quả chống lại SHA-1. Đây là một họ các hàm băm tương tự nhau nhưng có kích thước khối và độ dài băm khác nhau (ví dụ: SHA-256 tạo ra băm 256-bit, SHA-512 tạo ra băm 512-bit). SHA-2 sử dụng cấu trúc Merkle–Damgård với hàm nén Davies–Meyer.
- Hiện tại, SHA-2, đặc biệt là SHA-256, là hàm băm được sử dụng phổ biến nhất hiện nay và được coi là an toàn cho hầu hết các ứng dụng mật mã. Nó cung cấp mức độ bảo vệ đáng kể với sức mạnh xử lý máy tính hiện tại.

Họ SHA-3 (Keccak):

- Nguyên lý thiết kế (Cấu trúc Sponge): SHA-3 là thành viên mới nhất của họ SHA, được NIST phát hành vào tháng 8 năm 2015. Nó khác biệt đáng kể so với cấu trúc giống MD5 của SHA-1 và SHA-2, dựa trên hàm băm Keccak và sử dụng cấu trúc "sponge" (hấp thụ và vắt). Cấu trúc này cho phép hấp thụ bất kỳ lượng dữ liệu đầu vào nào và "vắt" ra một đầu ra giả ngẫu nhiên, về lý thuyết là an toàn hơn.
- Cải tiến và Vai trò trong tương lai: SHA-3 không được phát triển để thay thế SHA-2 mà là để cung cấp một lựa chọn thay thế với cấu trúc nội bộ khác biệt. Điều này nhằm tăng cường tính mạnh mẽ của bộ công cụ thuật toán băm của NIST và cung cấp một phương án dự phòng trong trường hợp các lỗ hổng mới được tìm thấy trong cấu trúc Merkle–Damgård của SHA-2. SHA-3 được thiết kế để chống lại các loại tấn công mà SHA-1 và SHA-2 có thể dễ bị tổn thương, chẳng hạn như tấn công mở rộng độ dài.

Việc NIST phát triển SHA-3 không phải để thay thế SHA-2 mà để cung cấp một giải pháp

thay thế với cấu trúc mật mã khác biệt thể hiện một sự thay đổi chiến lược hướng tới sự linh hoạt và đa dạng trong mật mã. Các cuộc tấn công trong quá khứ vào MD5, SHA-0 và SHA-1 (chia sẻ cấu trúc Merkle-Damgård tương tự) đã cho thấy tiềm năng của các lỗ hổng hệ thống nếu tất cả các tiêu chuẩn đều dựa vào các thiết kế tương tự. Do đó, việc phát triển SHA-3 với cấu trúc "sponge" hoàn toàn khác biệt là một biện pháp chủ động để xây dựng khả năng phục hồi trong hệ sinh thái mật mã. Nếu một điểm yếu cơ bản được tìm thấy trong một mô hình thiết kế (ví dụ: Merkle-Damgård), thì vẫn có một giải pháp thay thế rõ ràng (cấu trúc sponge) có sẵn, ngăn chặn một điểm lỗi duy nhất cho các chức năng bảo mật quan trọng. Đây là một cách tiếp cận phòng ngừa chống lại những đột phá trong phân tích mật mã trong tương lai.

Các ứng dụng của SHA

Các thuật toán SHA được sử dụng rộng rãi trong nhiều ứng dụng bảo mật:

- **Chữ ký số**: SHA là thành phần cốt lõi trong chữ ký số, được sử dụng để tạo ra bản tóm tắt tin nhắn trước khi nó được mã hóa bằng khóa riêng tư.
- **Lưu trữ mật khẩu**: Thay vì lưu trữ mật khẩu gốc, các hệ thống lưu trữ giá trị băm của mật khẩu để bảo vệ chúng khỏi bị lộ trong trường hợp cơ sở dữ liệu bị xâm phạm.
- **Kiểm tra tính toàn vẹn dữ liệu**: Bằng cách so sánh giá trị băm của một tệp hoặc khối dữ liệu, có thể xác định liệu dữ liệu có bị thay đổi trong quá trình truyền tải hoặc lưu trữ hay không (checksums).
- **Giao thức bảo mật**: SHA là một phần không thể thiếu của các giao thức như TLS/SSL, PGP, SSH, IPsec và S/MIME.

2.2 Phân tích mã nguồn

2.2.1 Giới thiệu tổng quan mã nguồn hệ thống

Hệ thống truyền tài liệu pháp lý phân tán được xây dựng với mục tiêu đảm bảo an toàn, bảo mật và toàn vẹn dữ liệu trong quá trình truyền tải giữa các thành phần (Sender, Server trung gian, Receiver). Mã nguồn hệ thống được tổ chức thành các module chức năng rõ ràng, bao gồm: xử lý mã hóa/giải mã, kiểm tra tính toàn vẹn, giao tiếp mạng, và giao diện người dùng. Việc phân tách này giúp hệ thống dễ dàng mở rộng, bảo trì và kiểm thử.

Các thành phần chính của hệ thống gồm:

- **Module mã hóa/giải mã (crypto_handler.py):** Được xây dựng dựa trên thư viện Py-cryptodome, thực hiện các chức năng mã hóa file, trao đổi khóa, ký số và kiểm tra toàn vẹn dữ liệu.

- **Module kiểm tra tính hợp lệ (validation.py):** Đảm nhận việc kiểm tra cấu trúc, định dạng và tính hợp lệ của các message, dữ liệu truyền nhận.
- **Module giao tiếp mạng:** Sử dụng giao thức socket TCP để truyền nhận dữ liệu giữa các thành phần.
- **Giao diện người dùng (GUI):** Được xây dựng bằng Tkinter, cung cấp trải nghiệm trực quan cho người dùng khi gửi/nhận tài liệu.
- **Các thành phần xử lý nghiệp vụ cho từng vai trò:** Hệ thống truyền tài liệu pháp lý phân tán được thiết kế theo mô hình client-server, với ba thành phần chính: Sender (người gửi), Server trung gian (server1), và Receiver (người nhận). Mỗi thành phần được tổ chức thành các module (components) riêng biệt.

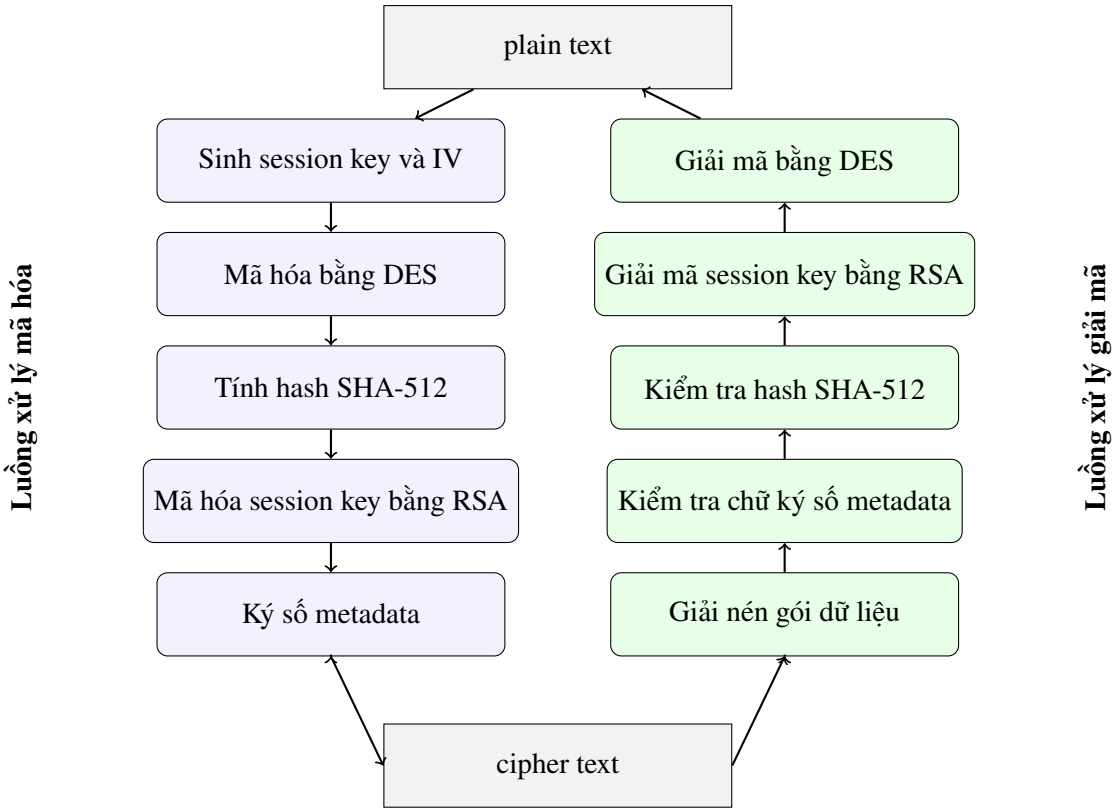
2.2.2 Hàm mã hóa và giải mã

Hệ thống được xây dựng nhằm bảo mật và đảm bảo toàn vẹn dữ liệu trong quá trình truyền nhận, thông qua việc kết hợp nhiều kỹ thuật mật mã hiện đại. Cụ thể, dữ liệu tập tin được mã hóa bằng thuật toán DES (Data Encryption Standard) hoạt động ở chế độ CBC (Cipher Block Chaining), với khóa phiên (session key) 8 byte được sinh ngẫu nhiên cho mỗi giao dịch nhằm tăng cường tính bảo mật và giảm thiểu rủi ro từ việc tái sử dụng khóa. Để đảm bảo chỉ bên nhận (Receiver) mới có thể giải mã và sử dụng khóa phiên này, hệ thống thực hiện mã hóa khóa phiên bằng thuật toán RSA 2048-bit theo chuẩn PKCS#1 v1.5 trong quá trình trao đổi. Bên cạnh đó, nhằm xác thực nguồn gốc dữ liệu và phòng chống hành vi giả mạo, metadata chứa các thông tin quan trọng như: tên tập tin, thời gian tạo, và mã định danh giao dịch (transaction ID) sẽ được ký số bằng RSA 2048-bit kết hợp hàm băm SHA-512. Cuối cùng, để đảm bảo tính toàn vẹn của dữ liệu trong suốt quá trình lưu trữ và truyền tải, hệ thống tính toán giá trị băm SHA-512 trên chuỗi bao gồm IV và ciphertext, từ đó giúp phát hiện kịp thời các thay đổi không mong muốn đối với dữ liệu.

2.2.3 Giao tiếp mạng

Chức năng chính Hệ thống đảm nhiệm việc quản lý kết nối TCP giữa các thành phần, đồng thời thực hiện truyền và nhận cả thông điệp (message) lẫn tập tin (file) một cách hiệu quả và đáng tin cậy. Mỗi thông điệp được kiểm tra và xử lý để đảm bảo tuân thủ đúng định dạng quy định, đồng thời hệ thống cung cấp cơ chế callback log hỗ trợ giao diện người dùng (GUI) trong việc hiển thị thông tin hoạt động theo thời gian thực. Toàn bộ quá trình truyền nhận đều được ghi lại một cách chi tiết nhằm phục vụ mục đích phát hiện lỗi, phân tích sự cố và truy vết các sự kiện đã xảy ra.

Luồng xử lý: Khi bên gửi (Sender) khởi tạo yêu cầu truyền tập tin, thành phần xử lý socket



Hình 2.5: Sơ đồ luồng mã hóa và giải mã

(socket_handler) sẽ chịu trách nhiệm đóng gói dữ liệu, gửi tuần tự từng thông điệp, nhận và xử lý phản hồi từ Server hoặc Receiver, đồng thời ghi log chi tiết ở từng bước. Ngược lại, khi bên nhận (Receiver) tiếp nhận tập tin, socket_handler sẽ thực hiện giải mã dữ liệu, kiểm tra tính toàn vẹn của nội dung, lưu tập tin xuống hệ thống và ghi lại kết quả xử lý, từ đó đảm bảo tính tin cậy và an toàn của quá trình truyền nhận dữ liệu.

2.2.4 Giao diện người dùng (GUI)

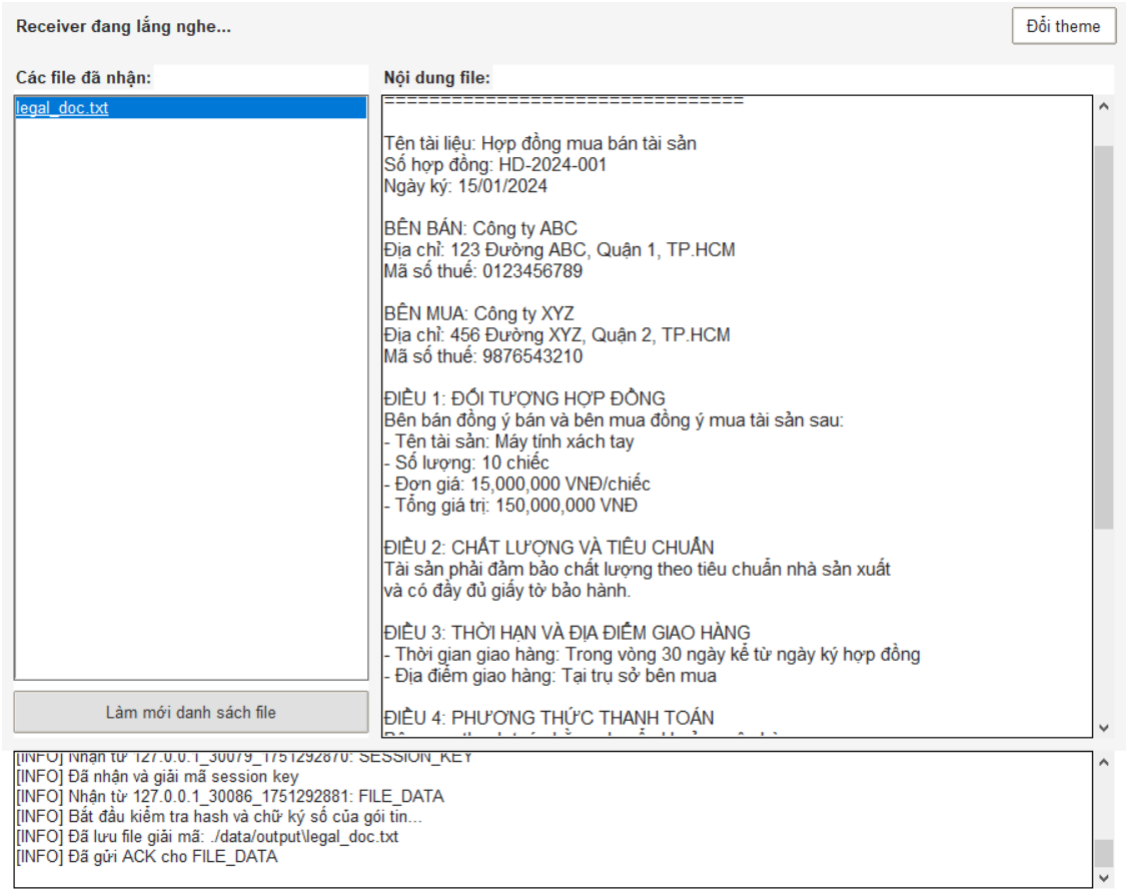
Module giao diện người dùng (GUI) được xây dựng nhằm cung cấp một môi trường trực quan, dễ sử dụng để hỗ trợ người dùng trong việc gửi và nhận tập tin, đồng thời theo dõi toàn bộ quá trình truyền nhận thông qua các thành phần hiển thị trạng thái và log chi tiết. Giao diện cho phép người dùng thực hiện các thao tác cơ bản như: lựa chọn tập tin cần gửi, thiết lập kết nối (handshake), theo dõi tiến trình truyền nhận, xem danh sách và nội dung các tập tin đã nhận. Bên cạnh đó, GUI còn cung cấp thông tin chi tiết về trạng thái kết nối, thông tin server hoặc receiver, và ghi lại log toàn bộ quá trình để phục vụ mục đích kiểm tra và truy vết.

Hệ thống giao diện được tổ chức thành nhiều thành phần riêng biệt nhằm nâng cao tính tái sử dụng và khả năng mở rộng. Cụ thể: tệp sender_gui.py chịu trách nhiệm xây dựng giao diện gửi tập tin, hiển thị log và trạng thái kết nối; tệp receiver_gui.py cung cấp giao diện quản lý các tập tin đã nhận, cho phép người dùng xem nội dung tập tin và log tương ứng. Ngoài ra, các thành

phần giao diện phụ trợ như components/file_selector.py, progressvà log_viewer.py lần lượt đảm nhận chức năng chọn tập tin, hiển thị tiến trình truyền nhận và trình bày log chi tiết theo thời gian thực, góp phần hoàn thiện trải nghiệm người dùng.

Hình 2.6: Giao diện gửi file

Hình 2.7: Giao diện một server trung gian



Hình 2.8: Giao diện nhận file

2.2.5 Các thành phần chính của hệ thống

Hệ thống truyền tài liệu pháp lý phân tán được thiết kế với ba thành phần cốt lõi: Sender, Server trung gian (Server Intermediate), và Receiver. Mỗi thành phần đảm nhiệm một vai trò riêng biệt trong quy trình truyền nhận, mã hóa và xác thực tài liệu, đồng thời phối hợp chặt chẽ để đảm bảo tính bảo mật, toàn vẹn và xác thực của dữ liệu trong suốt quá trình truyền tải. **Thành phần Sender (sender.py):** Sender là thành phần khởi tạo quá trình truyền tài liệu. Vai trò chính của Sender bao gồm:

- Khởi tạo kết nối với Server trung gian thông qua giao thức socket TCP.
- Thực hiện handshake với Receiver (thông qua Server trung gian) để xác nhận sẵn sàng truyền nhận dữ liệu và đồng bộ thông tin giao dịch.
- Yêu cầu và nhận public key của Receiver cho mỗi lần gửi file, đảm bảo mỗi phiên truyền file đều sử dụng khóa xác thực mới.
- Trao đổi khóa phiên (session key): Sinh khóa phiên ngẫu nhiên cho thuật toán mã hóa đối xứng (DES), sau đó mã hóa khóa này bằng public key của Receiver (RSA 2048-bit).

- Mã hóa, ký số và gửi file: Mã hóa nội dung file bằng DES, ký số metadata bằng RSA+SHA-512, đóng gói và gửi dữ liệu qua Server trung gian.
- Nhận phản hồi (ACK/NACK) từ Receiver để xác nhận kết quả truyền file.

Sender được thiết kế module hóa, dễ mở rộng, và tích hợp chặt chẽ với các thành phần kiểm tra hợp lệ, mã hóa, và giao tiếp mạng.

Thành phần Server trung gian (server_intermediate.py): Server trung gian đóng vai trò là cầu nối giữa Sender và Receiver, đảm bảo việc truyền tải dữ liệu được thực hiện an toàn, kiểm soát và có thể mở rộng. Các chức năng chính của Server trung gian bao gồm:

- Tiếp nhận kết nối từ cả Sender và Receiver, quản lý nhiều phiên giao dịch đồng thời.
- Chuyển tiếp các message: Nhận các message từ Sender (ví dụ: handshake, yêu cầu public key, dữ liệu file) và chuyển tiếp đến Receiver, đồng thời chuyển các phản hồi từ Receiver về lại Sender.
- Kiểm tra và ghi log các sự kiện truyền nhận, hỗ trợ phát hiện lỗi, truy vết và kiểm thử hệ thống.
- Đảm bảo tính toàn vẹn luồng dữ liệu: Server trung gian không giải mã nội dung, nhưng đảm bảo các message được chuyển tiếp đúng định dạng, đúng trình tự, và không bị thất lạc.

Thiết kế của Server trung gian giúp hệ thống dễ dàng mở rộng theo mô hình nhiều tầng, tăng tính linh hoạt và bảo mật cho quá trình truyền tài liệu. **3. Thành phần Receiver (receiver.py):** Receiver là thành phần cuối cùng trong chuỗi truyền nhận, chịu trách nhiệm nhận, xác thực và giải mã tài liệu. Các chức năng chính của Receiver bao gồm:

- Thiết lập kết nối với Server trung gian, sẵn sàng nhận các message và dữ liệu từ Sender.
- Thực hiện handshake để xác nhận sẵn sàng nhận file và đồng bộ thông tin giao dịch.
- Cung cấp public key cho Sender khi được yêu cầu, đảm bảo mỗi phiên truyền file đều sử dụng khóa xác thực mới.
- Nhận và giải mã session key: Nhận khóa phiên đã được mã hóa bằng RSA, giải mã bằng private key của Receiver.
- Giải mã file và kiểm tra toàn vẹn: Giải mã nội dung file bằng DES, kiểm tra hash SHA-512 để đảm bảo dữ liệu không bị thay đổi, xác thực chữ ký số metadata để đảm bảo nguồn gốc và tính xác thực của tài liệu.
- Gửi phản hồi (ACK/NACK) về cho Sender thông qua Server trung gian, xác nhận kết quả

nhận và giải mã file.

Receiver được thiết kế để đảm bảo an toàn tuyệt đối cho dữ liệu nhận được, đồng thời cung cấp giao diện trực quan cho người dùng kiểm tra, xem và quản lý các tài liệu đã nhận.

2.3 Thử nghiệm hệ thống

2.3.1 Thiết kế thử nghiệm

Mục tiêu thử nghiệm Mục tiêu của thử nghiệm là đánh giá khả năng xử lý, hiệu suất và độ ổn định của hệ thống truyền tài liệu pháp lý phân tán khi truyền các file có kích thước khác nhau. Thử nghiệm tập trung vào các tiêu chí:

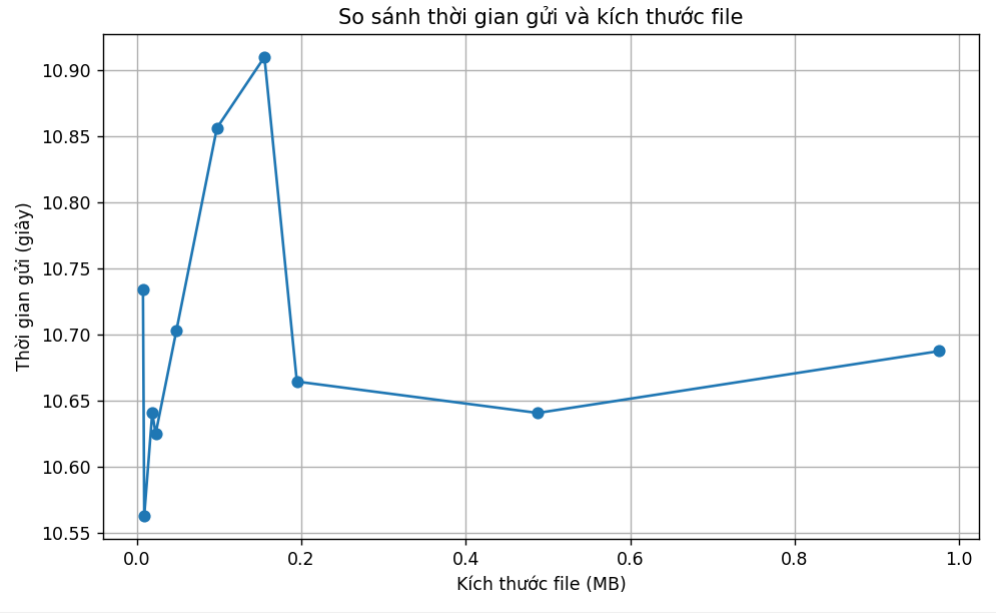
- Độ chính xác của quá trình mã hóa, giải mã và truyền nhận file.
- Hiệu suất (tốc độ) truyền file theo kích thước dữ liệu.
- Ổn định của hệ thống khi truyền nhiều file liên tiếp.

Chuẩn bị dữ liệu thử nghiệm Để đảm bảo tính khách quan và đa dạng, bộ dữ liệu thử nghiệm bao gồm các file với kích thước khác nhau, từ nhỏ đến lớn:

- File văn bản nhỏ: 10 KB, 20 KB, 50 KB, 100 KB, 160 KB, 200 KB.
- File văn bản vừa: 500 KB.
- File văn bản lớn: 1 MB.

Các file này được đặt trong thư mục data/ của hệ thống.

2.3.2 Kết quả thử nghiệm



Hình 2.9: Kết quả thử nghiệm hệ thống

Đánh giá về độ chính xác và ổn định

- Tính đúng đắn: Tất cả các file được gửi từ Sender đều được Receiver nhận và giải mã thành công. Nội dung file nhận được hoàn toàn trùng khớp với file gốc, xác nhận qua kiểm tra hash SHA-512.
- Tính ổn định: Hệ thống hoạt động ổn định trong suốt quá trình thử nghiệm, không xuất hiện lỗi kết nối, lỗi mã hóa/giải mã hoặc mất dữ liệu. Các phản hồi ACK/NACK được ghi nhận đầy đủ, giúp dễ dàng kiểm soát trạng thái truyền file.

Đánh giá về hiệu suất

- Kết quả đo thời gian gửi file cho thấy thời gian gửi không tăng tuyến tính tuyệt đối theo kích thước file, đặc biệt với các file nhỏ. Điều này là do chi phí khởi tạo kết nối, handshake, trao đổi khóa và xác thực chiếm tỷ trọng lớn so với thời gian truyền dữ liệu thực tế khi file nhỏ.
- Với các file lớn hơn, thời gian gửi có xu hướng tăng dần theo kích thước file, tuy nhiên tốc độ tăng không quá đột biến, chứng tỏ hệ thống xử lý tốt các file có dung lượng lớn
- Biểu đồ kết quả cho thấy các điểm dữ liệu được sắp xếp theo kích thước file tăng dần, với trục hoành là kích thước file (MB) và trục tung là thời gian gửi (giây). Đường biểu diễn có xu hướng đi lên nhẹ, phản ánh đúng đặc điểm của hệ thống truyền file bảo mật: chi phí cố định (overhead) lớn cho các file nhỏ, chi phí truyền dữ liệu chiếm ưu thế với file lớn.

Chương 3

Kết luận và hướng phát triển

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

Hệ thống truyền tải tài liệu pháp lý phân tán được xây dựng trên nền tảng các thuật toán mật mã hiện đại, đảm bảo các yêu cầu về bảo mật, toàn vẹn và xác thực dữ liệu trong quá trình truyền tải. Các thuật toán chính được sử dụng bao gồm: DES cho mã hóa đối xứng, RSA 2048-bit cho mã hóa bất đối xứng và ký số, cùng SHA-512 cho kiểm tra toàn vẹn dữ liệu.

Thuật toán mã hóa đối xứng DES (Data Encryption Standard): là thuật toán mã hóa khối đối xứng, sử dụng khóa bí mật có độ dài 56 bit để mã hóa và giải mã dữ liệu. Trong hệ thống, DES được sử dụng ở chế độ CBC (Cipher Block Chaining), kết hợp với một vector khởi tạo (IV) ngẫu nhiên cho mỗi phiên truyền file. Việc sử dụng DES giúp đảm bảo dữ liệu được mã hóa trước khi truyền qua mạng, ngăn chặn các đối tượng không được phép truy cập nội dung file. Đặc điểm nổi bật của DES là tốc độ mã hóa/giải mã nhanh, dễ triển khai, phù hợp với các ứng dụng truyền file vừa và nhỏ.

Thuật toán mã hóa bất đối xứng và ký số: RSA 2048-bit: RSA là thuật toán mã hóa bất đối xứng phổ biến, sử dụng cặp khóa công khai và bí mật với độ dài khóa 2048 bit trong hệ thống này. RSA được ứng dụng trong hai giai đoạn quan trọng:

- **Trao đổi khóa phiên (session key):** Mỗi lần gửi file, Sender sinh một session key ngẫu nhiên cho DES, sau đó mã hóa khóa này bằng public key của Receiver. Điều này đảm bảo chỉ Receiver mới có thể giải mã và sử dụng session key để giải mã nội dung file.
- **Ký số metadata:** Metadata của file (bao gồm tên file, thời gian, transaction id, v.v.) được ký số bằng private key của Sender kết hợp với hàm băm SHA-512. Việc này giúp xác thực nguồn gốc file và đảm bảo dữ liệu không bị giả mạo trong quá trình truyền tải.

RSA 2048-bit mang lại mức độ bảo mật cao, chống lại các tấn công brute-force và đảm bảo an toàn cho quá trình trao đổi khóa và xác thực dữ liệu.

Thuật toán kiểm tra toàn vẹn: SHA-512: là thuật toán băm một chiều thuộc họ SHA-2, cho ra kết quả băm có độ dài 512 bit. Trong hệ thống, SHA-512 được sử dụng để kiểm tra tính toàn vẹn của dữ liệu bằng cách băm chuỗi kết hợp giữa IV và ciphertext. Khi Receiver nhận được file, hệ thống sẽ tính lại giá trị băm này và so sánh với giá trị băm gửi kèm để phát hiện mọi thay đổi hoặc lỗi trong quá trình truyền tải. Đặc điểm của SHA-512 là khả năng chống va chạm rất cao, tốc độ băm nhanh trên các hệ thống hiện đại, đảm bảo dữ liệu không bị thay đổi ngoài ý muốn.

Việc kết hợp ba thuật toán trên trong hệ thống đã tạo nên một quy trình bảo mật toàn diện: dữ liệu được mã hóa an toàn, khóa phiên được trao đổi bảo mật, metadata được xác thực nguồn gốc, và toàn vẹn dữ liệu luôn được kiểm tra nghiêm ngặt. Các thuật toán được lựa chọn đều là các tiêu chuẩn công nghiệp, đã được kiểm chứng về độ an toàn và hiệu quả trong thực tiễn.

3.2 Đề xuất cải tiến

3.2.1 Cải tiến về bảo mật

Thay thế DES bằng AES-GCM: AES (Advanced Encryption Standard) với khóa 128 hoặc 256 bit là tiêu chuẩn quốc tế (FIPS197, ISO/IEC18033-3), ở chế độ Galois/Counter Mode (GCM), cho phép mã hóa đồng thời kết hợp xác thực dữ liệu. Nhờ tính song song hóa tốt, AES-GCM có thông lượng cao hơn đáng kể so với DES-CBC, đồng thời hỗ trợ cả xác thực mã (integrity) không cần MAC riêng.

Bổ sung xác thực hai chiều: Hiện tại hệ thống chỉ xác thực từ Sender đến Receiver. Việc thêm xác thực từ Receiver về Sender (mutual authentication) — ví dụ bằng chứng thư số hoặc ECDSA — sẽ tăng cường khả năng ngăn chặn giả danh và đảm bảo tính toàn vẹn hai chiều giữa các bên.

Ngăn chặn replay attack bằng nonce/timestamp: Metadata nên chứa *timestamp* hoặc *nonce* để hệ thống tự động phát hiện và bác bỏ các gói tin đã được truyền trước đó, phòng chống các cuộc tấn công replay. Đây là phương pháp bảo mật được sử dụng phổ biến trong TLS và IPsec.

Triển khai Elliptic Curve Cryptography (ECC): ECC 256-bit cung cấp độ bảo mật tương đương RSA 3072-bit nhưng với kích thước khóa nhỏ hơn nhiều, giúp tăng tốc độ sinh khóa và ký số, tiết kiệm băng thông và tài nguyên thiết bị, đặc biệt là trên thiết bị IoT và mobile.

Áp dụng tính toán lượng tử: RSA và ECC đều có nguy cơ bị bẻ khóa khi máy tính lượng tử thực sự lớn xuất hiện. Do vậy, nên khảo sát và tích hợp các thuật toán hậu lượng tử như NTRU

hoặc Kyber, theo lộ trình của NIST, để hệ thống có thể thích ứng dài hạn.

3.2.2 Cải tiến về hiệu suất

Truyền bất đồng bộ hoặc đa luồng: Áp dụng I/O không chặn (asynchronous I/O) hoặc đa luồng để truyền nhiều file song song, giúp tận dụng băng thông và đa nhân CPU, giảm thiểu thời gian tổng thể cho các file lớn.

Tái sử dụng session key: Trong các phiên truyền nhiều file nhỏ, sử dụng chung session key thay vì tạo mới cho mỗi file giúp giảm overhead do mã hóa khóa RSA và thiết lập kết nối, nhờ đó nâng cao tốc độ xử lý.

Nén dữ liệu trước khi mã hóa: Sử dụng thư viện như zlib hoặc gzip để giảm đáng kể kích thước file văn bản trước khi mã hóa, giúp tiết kiệm băng thông và giảm thời gian mã hóa – một chiến lược hiệu quả với dữ liệu văn bản thuần.

Tối ưu hóa ghi log: Chuyển sang chế độ ghi log bất đồng bộ hoặc giảm mức độ chi tiết khi ở môi trường sản xuất nhằm tránh bottleneck I/O, trong khi vẫn giữ đủ dữ liệu cần thiết cho truy vết và kiểm thử khi cần.

Cải thiện giao diện người dùng (UI): Thêm thanh tiến trình, chức năng tạm dừng/tiếp tục, hiển thị lỗi và tự động retry khi truyền file lớn hoặc mạng không ổn định, giúp tăng chất lượng trải nghiệm người dùng và tính bền vững ứng dụng trong thực tế.

Tài liệu tham khảo

[1] Giáo trình Mật mã học và An toàn thông tin - TS Thái Thanh Tùng, 2011, NXB Thông tin và Truyền thông.

[2] "What Is a SSL/TLS Handshake?"[https://sematext.com/glossary/ssl-tls-handshake/]

[3] "RFC 2246: The TLS Protocol Version 1.0" DataSheet

[4] "What happens in a TLS handshake? | SSL handshake"[https://www.cloudflare.com/learning/ssl/what-happens-in-a-tls-handshake/]

[5] Mật mã đối xứng: Giải thuật DES - Phạm Nguyên Khang.

[6] "RSA Algorithm in Cryptography: Rivest Shamir Adleman Explained"[https://www.splunk.com/en_us algorithm-cryptography.html]

[7] "FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION:Secure Hash Standard (SHS)"

[8] "Hash Functions" [https://csrc.nist.gov/projects/hash-functions]