TRƯỜNG ĐẠI HỌC ĐẠI NAM

KHOA CÔNG NGHỆ THÔNG TIN

BÀI TẬP LỚN

HỌC PHẦN:

ĐỀ TÀI: ỨNG DỤNG BẢO MẬT TIN NHẮN VĂN BẢN VỚI MÃ HOÁ TRIPLEDES VÀ XÁC THỰC RSA

Giảng viên: TS. Trần Đăng Công

Sinh Viên thực hiện:

TT	*Mã SV*	*Họ và Tên*	*Ngày Sinh*	*Lớp*
1	**1771020474**	**Nguyễn Ngọc Minh**	**22/09/2005**	**CNTT 17-05**
2	**1771020083**	**Phạm Văn Ngọc Biên**	**24/06/2005**	**CNTT 17-05**
3	**1771020037**	**Lê Đức Anh**	**28/12/2005**	**CNTT 17-05**
4	**1771020001**	**Nguyễn Văn An**	**24/10/2005**	**CNTT 17-05**

Hà Nội, năm 2025

TRƯỜNG ĐẠI HỌC ĐẠI NAM

KHOA CÔNG NGHỆ THÔNG TIN

BÀI TẬP LỚN

HỌC PHẦN:

ĐỀ TÀI: ỨNG DỤNG BẢO MẬT TIN NHẮN VĂN BẢN VỚI MÃ HOÁ TRIPLEDES VÀ XÁC THỰC RSA

TT	*Mã SV*	*Họ và Tên*	*Ngày Sinh*	*Điểm*	
				Bằng Số	*Bằng Chữ*
1	**1771020474**	**Nguyễn Ngọc Minh**	**22/09/2005**		
2	**1771020083**	**Phạm Văn Ngọc Biên**	**24/06/2005**		
3	**1771020037**	**Lê Đức Anh**	**28/12/2005**		
4	**1771020001**	**Nguyễn Văn An**	**24/10/2005**		

CÁN BỘ CHẤM THI 1 **CÁN BỘ CHẤM THI 2**

Hà Nội, năm 2025

MỤC LỤC

DANH MỤC HÌNH ẢNH

CHƯƠNG 1. GIỚI THIỆU CHUNG

1.1. Lý do chọn đề tài

1.1.1. Tầm quan trọng của bảo mật nhắn tin

Trong thời đại công nghệ số, các ứng dụng nhắn tin trực tuyến như WhatsApp, Telegram, và các nền tảng giao tiếp nội bộ đã trở thành phương tiện chính để trao đổi thông tin cá nhân, kinh doanh, và thậm chí là dữ liệu nhạy cảm. Tuy nhiên, sự phổ biến của các nền tảng này cũng đi kèm với rủi ro bảo mật nghiêm trọng, bao gồm nghe lén (eavesdropping), giả mạo danh tính (impersonation), và sửa đổi dữ liệu (data tampering). Các cuộc tấn công mạng ngày càng tinh vi, với các kỹ thuật như tấn công man-in-the-middle (MITM) hoặc khai thác lỗ hổng giao thức, khiến việc bảo vệ nội dung tin nhắn trở thành yêu cầu cấp thiết.z

Bảo mật nhắn tin không chỉ đảm bảo quyền riêng tư của người dùng mà còn duy trì lòng tin trong các giao dịch trực tuyến, đặc biệt trong các lĩnh vực như tài chính, y tế, và quản lý doanh nghiệp. Một hệ thống nhắn tin thiếu bảo mật có thể dẫn đến rò rỉ thông tin nhạy cảm, gây thiệt hại nghiêm trọng về tài chính và danh tiếng. Do đó, việc nghiên cứu và triển khai các giải pháp mã hóa mạnh mẽ để bảo vệ nội dung tin nhắn và xác thực danh tính người dùng là một ưu tiên hàng đầu trong lĩnh vực an ninh mạng.

1.1.2. Lợi ích của TRIPLEDES và RSA

Sự kết hợp giữa mã hóa TRIPLEDES (Data Encryption Standard) và RSA (Rivest-Shamir-Adleman) trong ứng dụng nhắn tin mang lại hiệu quả cao về cả tốc độ và độ an toàn, phù hợp với yêu cầu bảo mật hiện đại.

- TRIPLEDES là một thuật toán mã hóa đối xứng sử dụng khóa 56-bit, hoạt động hiệu quả trong chế độ CFB (Cipher Feedback). Với cơ chế mã hóa khối, TRIPLEDES có khả năng xử lý nhanh dữ liệu văn bản, đặc biệt phù hợp cho các ứng dụng nhắn tin yêu cầu thời gian thực. Mặc dù độ dài khóa 56-bit của TRIPLEDES không còn đủ mạnh để chống lại các cuộc tấn công brute-force hiện đại, việc sử dụng TRIPLEDES trong các ứng dụng nhỏ, kết hợp với RSA để bảo vệ khóa, vẫn đảm bảo hiệu suất và tính thực tiễn. Chế độ CFB cho phép mã hóa dữ liệu theo luồng, giảm độ trễ và tăng tính linh hoạt khi xử lý các tin nhắn có độ dài thay đổi.z

- RSA là một thuật toán mã hóa bất đối xứng sử dụng cặp khóa công khai và khóa bí mật, với độ dài khóa 2048-bit trong đề tài này. RSA không chỉ hỗ trợ mã hóa khóa TRIPLEDES để trao đổi an toàn mà còn cung cấp cơ chế ký số để xác minh danh tính người gửi, ngăn chặn giả mạo. Sử dụng padding OAEP (Optimal Asymmetric Encryption Padding) cùng hàm băm SHA-256, RSA đảm bảo tính bảo mật cao và khả năng chống lại các tấn công như chosen-ciphertext attack. Việc trao đổi khóa công khai qua kết nối P2P (Peer-to-Peer) giúp hai bên thiết lập kênh an toàn mà không cần cơ sở hạ tầng phức tạp, phù hợp cho các ứng dụng nhắn tin quy mô nhỏ hoặc nội bộ.

Sự kết hợp này tận dụng ưu điểm của cả hai thuật toán: TRIPLEDES đảm bảo tốc độ mã hóa, còn RSA cung cấp lớp bảo vệ cho khóa và xác thực, tạo nên một hệ thống bảo mật toàn diện.

1.1.3. Ứng dụng thực tiễn

Đề tài đáp ứng nhu cầu thực tiễn trong việc xây dựng các hệ thống nhắn tin bảo mật, đặc biệt trong bối cảnh các tổ chức và cá nhân ngày càng phụ thuộc vào giao tiếp trực tuyến. Các ứng dụng thực tiễn bao gồm:

- Hệ thống nhắn tin nội bộ doanh nghiệp: Đảm bảo thông tin kinh doanh, kế hoạch chiến lược, và dữ liệu khách hàng không bị rò rỉ hoặc sửa đổi.

- Ứng dụng giao tiếp cá nhân: Bảo vệ quyền riêng tư của người dùng trong các cuộc trò chuyện nhạy cảm.

- Hệ thống giao dịch tài chính: Xác thực danh tính và bảo mật nội dung trao đổi giữa các bên tham gia.

- Chống giả mạo và sửa đổi dữ liệu: Sử dụng SHA-256 để kiểm tra tính toàn vẹn, đảm bảo tin nhắn không bị can thiệp trong quá trình truyền tải.

Việc triển khai một hệ thống nhắn tin sử dụng TRIPLEDES và RSA không chỉ giải quyết các vấn đề bảo mật mà còn cung cấp một mô hình tham khảo cho các ứng dụng yêu cầu bảo mật cao, đóng góp vào sự phát triển của các giải pháp an ninh mạng hiện đại.

1.2. Mục tiêu của báo cáo

1.2.1. Giới thiệu cơ chế bảo mật

Báo cáo nhằm trình bày chi tiết các cơ chế mã hóa và xác thực được sử dụng trong hệ thống nhắn tin bảo mật, cụ thể:

- TRIPLEDES (CFB mode): Phân tích cách TRIPLEDES mã hóa tin nhắn văn bản trong chế độ CFB, đảm bảo tính bí mật và hiệu suất. Báo cáo sẽ làm rõ quy trình mã hóa, giải mã, và vai trò của khóa TRIPLEDES trong việc bảo vệ nội dung.

- RSA 2048-bit (OAEP + SHA-256): Giới thiệu cơ chế mã hóa bất đối xứng RSA với độ dài khóa 2048-bit, sử dụng padding OAEP và hàm băm SHA-256 để mã hóa khóa TRIPLEDES và ký số. Báo cáo sẽ giải thích cách RSA hỗ trợ xác thực danh tính và trao đổi khóa an toàn.

- SHA-256: Trình bày vai trò của hàm băm SHA-256 trong việc kiểm tra tính toàn vẹn của tin nhắn, đảm bảo dữ liệu không bị sửa đổi trong quá trình truyền tải.

1.2.2. Xây dựng ứng dụng

Mục tiêu tiếp theo là mô tả quy trình triển khai ứng dụng nhắn tin bảo mật, bao gồm:

- Thiết kế hệ thống: Xây dựng một ứng dụng nhắn tin P2P với giao diện người dùng thân thiện, sử dụng các công nghệ như Node.js, Socket.IO, Vue.js, JSEncrypt (cho RSA), và thư viện hỗ trợ TRIPLEDES.

- Luồng xử lý: Chi tiết hóa các bước thực hiện, từ handshake (trao đổi "Hello!" và "Ready!"), trao đổi khóa công khai RSA, xác thực danh tính, mã hóa tin nhắn bằng TRIPLEDES, đến kiểm tra toàn vẹn và giải mã.

- Tính năng: Ứng dụng hỗ trợ phòng chat riêng tư với định danh duy nhất, thông báo khi người dùng tham gia/rời phòng, và giao diện tự động cuộn (autoscroll) để hiển thị tin nhắn mới nhất.

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT VỀ MÃ HÓA TRIPLEDES VÀ RSA

2.1. Hệ mã hóa TRIPLETRIPLE

2.1.1. Khái niệm và đặc điểmz

Data Encryption Standard (TRIPLEDES) là một thuật toán mã hóa đối xứng được phát triển vào những năm 1970 bởi IBM và được chuẩn hóa bởi Viện Tiêu chuẩn hóa và Công nghệ Quốc gia Hoa Kỳ (NIST) vào năm 1977. TRIPLEDES sử dụng cùng một khóa bí mật cho cả quá trình mã hóa và giải mã, do đó được gọi là mã hóa đối xứng. Thuật toán này hoạt động trên các khối dữ liệu có kích thước cố định và được thiết kế để đảm bảo tính bí mật của thông tin trong các ứng dụng bảo mật.

Đặc điểm chính của TRIPLEDES:

- Khóa 56-bit: TRIPLEDES sử dụng khóa có độ dài 64-bit, nhưng chỉ 56-bit được sử dụng thực sự cho mã hóa, còn 8-bit còn lại dùng để kiểm tra chẵn lẻ (parity check). Khóa này được tạo ngẫu nhiên và phải được chia sẻ an toàn giữa người gửi và người nhận trước khi bắt đầu quá trình mã hóa.

- Mã hóa khối: TRIPLEDES chia dữ liệu đầu vào thành các khối 64-bit và mã hóa từng khối riêng lẻ. Mỗi khối được xử lý qua 16 vòng (rounds) lặp lại, sử dụng các phép toán thay thế (substitution) và hoán vị (permutation) để tạo ra ciphertext.

- Chế độ CFB (Cipher Feedback): Trong đề tài này, TRIPLEDES được sử dụng ở chế độ CFB, một chế độ mã hóa luồng (stream cipher mode). Trong CFB, output của quá trình mã hóa khối trước được sử dụng như một phản hồi (feedback) để mã hóa khối tiếp theo, tạo ra một chuỗi dữ liệu mã hóa liên tục. Điều này giúp CFB phù hợp với dữ liệu có độ dài thay đổi, như tin nhắn văn bản, và giảm thiểu lỗi lan truyền so với các chế độ như ECB (Electronic Codebook).

- Cơ chế hoạt động: TRIPLEDES dựa trên cấu trúc Feistel, chia khối dữ liệu 64-bit thành hai nửa (trái và phải) và thực hiện các phép toán phức tạp qua 16 vòng. Mỗi vòng sử dụng một khóa con (subkey) được tạo từ khóa chính thông qua thuật toán lập lịch khóa (key schedule). Các phép toán bao gồm mở rộng (expansion), thay thế qua S-box, và hoán vị qua P-box, đảm bảo tính phi tuyến và độ phức tạp của thuật toán.

Chế độ CFB được chọn trong đề tài vì nó cho phép mã hóa dữ liệu theo luồng, phù hợp với đặc điểm của tin nhắn văn bản có độ dài không cố định. Ngoài ra, CFB giúp giảm độ trễ trong quá trình truyền tải, một yếu tố quan trọng trong các ứng dụng nhắn tin thời gian thực.

2.1.2. Ưu và nhược điểm

Ưu điểm:

- Tốc độ nhanh: TRIPLEDES được thiết kế để hoạt động hiệu quả trên phần cứng và phần mềm, đặc biệt là trong các hệ thống có tài nguyên hạn chế. Với kích thước khối 64-bit và cơ chế Feistel đơn giản, TRIPLEDES có thể mã hóa và giải mã dữ liệu nhanh chóng, đáp ứng yêu cầu của các ứng dụng nhắn tin thời gian thực. Trong chế độ CFB, TRIPLEDES xử lý dữ liệu theo luồng, giúp giảm độ trễ khi mã hóa các tin nhắn ngắn hoặc dài.

- Phù hợp với dữ liệu văn bản: TRIPLEDES hoạt động tốt với dữ liệu văn bản, đặc biệt khi kết hợp với các kỹ thuật padding để xử lý các khối không đủ 64-bit. Điều này làm cho TRIPLEDES trở thành lựa chọn lý tưởng cho việc mã hóa nội dung tin nhắn trong các hệ thống nhắn tin bảo mật.

- Lịch sử sử dụng lâu dài: TRIPLEDES đã được sử dụng rộng rãi trong nhiều thập kỷ và được kiểm chứng trong các ứng dụng bảo mật, như hệ thống ngân hàng và giao tiếp quân sự. Mặc dù không còn là chuẩn mã hóa chính, TRIPLEDES vẫn hữu ích trong các ứng dụng nhỏ khi kết hợp với các cơ chế bảo vệ khóa mạnh hơn.

Nhược điểm:

- Khóa ngắn: Độ dài khóa 56-bit của TRIPLEDES là một hạn chế lớn trong bối cảnh công nghệ hiện đại. Với sự phát triển của các siêu máy tính và các thuật toán tấn công brute-force, một khóa 56-bit có thể bị bẻ khóa trong thời gian ngắn (khoảng vài giờ đến vài ngày, tùy thuộc vào tài nguyên tính toán). Điều này khiến TRIPLEDES không còn đủ an toàn để sử dụng độc lập trong các hệ thống bảo mật hiện đại.

- Yêu cầu bảo vệ khóa: Vì TRIPLEDES là mã hóa đối xứng, khóa phải được chia sẻ an toàn giữa người gửi và người nhận. Nếu khóa bị lộ, toàn bộ dữ liệu mã hóa có thể bị giải

mã. Trong đề tài này, hạn chế này được khắc phục bằng cách sử dụng RSA để mã hóa và trao đổi khóa TRIPLEDES, đảm bảo rằng chỉ người nhận có private key mới có thể giải mã khóa TRIPLEDES.

- Không chống lại các tấn công nâng cao: TRIPLEDES dễ bị tổn thương trước các tấn công như phân tích vi sai (differential cryptanalysis) và phân tích tuyến tính (linear cryptanalysis), đặc biệt nếu không được triển khai đúng cách. Chế độ CFB, mặc dù cải thiện tính linh hoạt, vẫn yêu cầu vector khởi tạo (initialization vector, IV) được quản lý cẩn thận để tránh các vấn đề bảo mật như tái sử dụng IV.

Để khắc phục các nhược điểm, đề tài sử dụng RSA để mã hóa khóa TRIPLEDES trước khi trao đổi, đảm bảo rằng khóa TRIPLEDES được bảo vệ trong quá trình truyền tải. Ngoài ra, việc kiểm tra toàn vẹn bằng SHA-256, giúp phát hiện bất kỳ sự can thiệp nào vào ciphertext, tăng cường thêm lớp bảo mật cho hệ thống.

2.1.3. Ứng dụng trong mã hóa tin nhắn

Trong các hệ thống nhắn tin bảo mật, thuật toán TRIPLEDES được ứng dụng để mã hóa nội dung tin nhắn văn bản thành ciphertext, giúp đảm bảo chỉ có người nhận sở hữu đúng khóa TRIPLEDES mới có thể giải mã và đọc được nội dung tin nhắn. Quá trình này bắt đầu khi người dùng nhập nội dung tin nhắn cần gửi, chẳng hạn như: "Xin chào, đây là tin nhắn bảo mật!".

Vì TRIPLEDES là thuật toán mã hóa khối với kích thước khối 64-bit nên nếu độ dài của tin nhắn không chia hết cho 64-bit, hệ thống sẽ tự động áp dụng kỹ thuật padding, phổ biến như PKCS7, nhằm đảm bảo dữ liệu phù hợp với yêu cầu về kích thước khối của TRIPLEDES. Trong chế độ hoạt động CFB (Cipher Feedback Mode), TRIPLEDES sử dụng một vector khởi tạo ngẫu nhiên (IV) kết hợp với khóa bí mật để tiến hành mã hóa từng đoạn dữ liệu liên tiếp. Vector IV này sẽ được gửi kèm theo bản mã ciphertext nhằm giúp người nhận có thể giải mã chính xác nội dung khi nhận được tin nhắn.

Kết quả cuối cùng của quá trình mã hóa là một chuỗi ciphertext dưới dạng Base64, giúp dễ dàng truyền tải dữ liệu dưới dạng văn bản qua các kênh truyền thông như hệ thống chat peer-to-peer (P2P).

Để tăng cường mức độ bảo mật, hệ thống sẽ tích hợp thêm các thuật toán RSA và SHA-256. Cụ thể, khóa TRIPLEDES sẽ được mã hóa bằng public key RSA của người nhận trước khi gửi đi, nhằm đảm bảo rằng khóa bí mật không bị lộ trong quá trình truyền tải. Đồng thời, hệ thống tạo ra một mã băm (hash) SHA-256 của ciphertext, sau đó thực hiện ký số bằng thuật toán RSA. Điều này giúp kiểm tra tính toàn vẹn của tin nhắn, đảm bảo rằng nội dung không bị thay đổi hay giả mạo trên đường truyền.

Nhờ vào tốc độ mã hóa nhanh và khả năng xử lý luồng dữ liệu hiệu quả của chế độ CFB, TRIPLEDES đáp ứng tốt các yêu cầu của ứng dụng nhắn tin thời gian thực. Người dùng có thể gửi và nhận tin nhắn liên tục mà không gặp tình trạng trễ đáng kể. Đặc điểm này đặc biệt quan trọng trong các phòng chat riêng tư, nơi hai người cần giao tiếp an toàn và tức thời.

Mặc dù TRIPLEDES hiện không còn được xem là thuật toán mạnh nhất về mặt bảo mật, song việc kết hợp TRIPLEDES với RSA và SHA-256 trong các ứng dụng nhắn tin quy mô nhỏ vẫn mang lại sự cân bằng hợp lý giữa hiệu suất và độ an toàn. Mô hình này đặc biệt phù hợp với các ứng dụng nội bộ như hệ thống nhắn tin trong doanh nghiệp hoặc các nền tảng giao tiếp cá nhân, nơi yêu cầu bảo mật ở mức trung bình nhưng vẫn đảm bảo tính khả thi và hiệu quả khi triển khai.

2.2. Hệ mã hóa RSA

2.2.1. Khái niệmz

RSA là một trong những thuật toán mã hóa bất đối xứng phổ biến nhất hiện nay, được phát minh vào năm 1977 bởi ba nhà khoa học người Mỹ: Rivest, Shamir và Adleman, cũng chính là nguồn gốc của tên gọi RSA. Khác với các hệ mã hóa đối xứng như TRIPLEDES, trong RSA, quá trình mã hóa và giải mã sử dụng hai khóa khác nhau: khóa công khai (public key) và khóa bí mật (private key).

Nguyên lý hoạt động của RSA dựa trên tính khó khăn của bài toán phân tích số nguyên lớn thành các thừa số nguyên tố. Trong hệ thống RSA, người gửi có thể sử dụng khóa công khai của người nhận để mã hóa dữ liệu. Chỉ người nhận, với khóa bí mật tương ứng, mới có thể giải mã dữ liệu đó. Chính vì vậy, RSA được xem là phương pháp hiệu quả để đảm bảo tính bảo mật và xác thực trong quá trình truyền thông tin trên môi trường mạng.

Bên cạnh khả năng mã hóa dữ liệu, RSA còn hỗ trợ ký số, cho phép người gửi gắn chữ ký điện tử vào thông điệp nhằm xác thực nguồn gốc và đảm bảo tính toàn vẹn của dữ liệu. Nhờ đó, RSA ngày càng được ứng dụng rộng rãi trong nhiều lĩnh vực khác nhau, từ bảo mật mạng đến giao dịch điện tử.

2.2.2. Ứng dụngz

RSA đóng vai trò nền tảng trong nhiều lĩnh vực của an ninh thông tin hiện đại, đặc biệt là trong các hệ thống truyền thông cần mức độ bảo mật cao. Một trong những ứng dụng phổ biến nhất của RSA là trong quá trình trao đổi khóa bí mật của các hệ mã đối xứng như TRIPLEDES hoặc AES. Thay vì chia sẻ trực tiếp khóa bí mật qua các kênh truyền dễ bị nghe lén, hệ thống sẽ sử dụng RSA để mã hóa khóa này. Người nhận, với khóa bí mật riêng, sẽ giải mã và nhận được khóa đối xứng để tiếp tục quá trình trao đổi dữ liệu an toàn.z

Ngoài ra, RSA còn được sử dụng rộng rãi trong việc xác thực danh tính và ký số tài liệu điện tử. Khi người gửi muốn đảm bảo thông điệp không bị thay đổi và chứng minh nguồn gốc, họ sẽ tạo ra một chữ ký số bằng cách mã hóa bản băm (hash) của thông điệp bằng khóa bí mật của chính mình. Người nhận chỉ cần sử dụng khóa công khai của người gửi để xác minh tính hợp lệ của chữ ký, từ đó đảm bảo thông điệp chưa bị chỉnh sửa và đúng là do người gửi phát đi.

Trong thực tế, RSA xuất hiện ở nhiều hệ thống bảo mật nổi tiếng như SSL/TLS trong giao thức truyền thông Internet, ứng dụng trong các dịch vụ email mã hóa, ví điện tử, chữ ký số, thẻ thông minh và nhiều giải pháp xác thực người dùng khác. Đặc biệt, trong các ứng dụng chat riêng tư hoặc các hệ thống truyền thông tin nhạy cảm, RSA đóng vai trò quan trọng trong việc trao đổi khóa và xác thực, góp phần đảm bảo môi trường giao tiếp an toàn, tin cậy.

2.3. Sinh khóa RSA

Thuật toán RSA (Rivest-Shamir-Adleman) là một hệ mã hóa bất đối xứng dựa trên bài toán nhân các số nguyên tố lớn và bài toán tìm nghịch đảo modulo. Quá trình sinh khóa RSA là bước nền tảng để tạo ra cặp khóa công khai (public key) và khóa bí mật (private key), được sử dụng trong việc mã hóa khóa TRIPLEDES, xác thực danh tính, và ký số trong hệ thống nhắn tin bảo mật. Trong đề tài này, RSA được triển khai với độ dài khóa 2048-bit để đảm bảo tính an toàn trước các cuộc tấn công tính toán hiện đại.

2.3.1. Quy trình sinh khóa

Quy trình sinh khóa RSA là một chuỗi các bước toán học nhằm tạo ra cặp khóa (public key, private key) dựa trên các tính chất của số học modulo và lý thuyết số nguyên tố. Dưới đây là mô tả chi tiết từng bước trong quy trình:

1. **Chọn hai số nguyên tố p và q (2048-bit)**

- Mô tả: Hai số nguyên tố lớn p và q được chọn ngẫu nhiên, mỗi số có độ dài khoảng 2048-bit (tương đương 617 chữ số thập phân). Việc chọn số nguyên tố lớn đảm bảo rằng việc phân tích nhân tử (factorization) của tích p × q là bất khả thi với công nghệ tính toán hiện tại, ngay cả với các siêu máy tính hoặc các thuật toán phân tích nhân tử tiên tiến như General Number Field Sieve (GNFS).

- Cách thực hiện: Sử dụng các thuật toán kiểm tra số nguyên tố như Miller-Rabin hoặc AKS để xác định p và q là số nguyên tố. Trong thực tế, các thư viện mã hóa như JSEncrypt tự động sinh các số nguyên tố ngẫu nhiên với độ dài xác định (2048-bit trong trường hợp này). Ví dụ, p và q có thể là các số lớn như:

$$p \approx 2^{1024}, q \approx 2^{1024}$$

Để đảm bảo tính ngẫu nhiên, các số này thường được tạo bằng các trình sinh số ngẫu nhiên bảo mật (cryptographically secure random number generator).

- Lưu ý: p và q phải được giữ bí mật, vì nếu chúng bị lộ, kẻ tấn công có thể dễ dàng tính toán các khóa RSA.

2. **Tính n = p × q**

- Mô tả: Tính tích n = p × q, gọi là **modulus**, được sử dụng trong cả public key và private key. Giá trị n xác định phạm vi của các phép toán modulo trong quá trình mã hóa và giải mã. Với p và q là số nguyên tố 2048-bit, n sẽ có độ dài khoảng 4096-bit, đảm bảo độ an toàn cao.

- Ví dụ minh họa: Nếu p = 61 và q = 53 (trong thực tế là các số lớn hơn nhiều), thì:

$$n = 61 \times 53 = 3233$$

Trong đề tài, n sẽ là một số rất lớn (khoảng 2^4096), khiến việc phân tích nhân tử trở thành bài toán bất khả thi.

Vai trò: n là thành phần chính trong cả public key và private key, được chia sẻ công khai trong public key và sử dụng trong các phép toán mã hóa/giải mã.

3. **Tính φ(n) = (p − 1)(q − 1)**

Mô tả: Hàm phi Euler φ(n) được tính dựa trên p và q, đại diện cho số lượng số nguyên dương nhỏ hơn n và nguyên tố cùng nhau với n. Với p và q là số nguyên tố, φ(n) được tính đơn giản bằng:

$$\varphi(n) = (p - 1) \times (q - 1)$$

Giá trị φ(n) được sử dụng để chọn số mũ công khai e và tính số mũ bí mật d. φ(n) phải được giữ bí mật để đảm bảo an toàn của thuật toán.

Ví dụ minh họa: Với p = 61, q = 53: $\varphi(n) = (61 - 1) \times (53 - 1) = 60 \times 52 = 3120$

Trong thực tế, với p và q là số 2048-bit, φ(n) sẽ là một số rất lớn, nhưng luôn nhỏ hơn n.

Lưu ý: Trong một số triển khai RSA, có thể sử dụng hàm Carmichael λ(n) = BCNN(p − 1, q − 1) thay vì φ(n). Tuy nhiên, trong đề tài này, φ(n) được sử dụng vì nó đơn giản hơn và phù hợp với các thư viện như JSEncrypt.

4. **Chọn e (thường 65537) sao cho 1 < e < φ(n) và ƯCLN(e, φ(n)) = 1**

Mô tả: Số mũ công khai e (public exponent) được chọn trong khoảng từ 1 đến φ(n), sao cho e và φ(n) nguyên tố cùng nhau (tức là ƯCLN(e, φ(n)) = 1). Giá trị e phổ biến nhất là 65537 (2^16 + 1), vì:

- Nó là một số nguyên tố đủ lớn để đảm bảo an toàn.
- Nó có dạng 2^n + 1, giúp tối ưu hóa hiệu suất tính toán trong quá trình mã hóa (do ít bit 1 trong biểu diễn nhị phân).
- Nó đủ nhỏ để giảm thời gian mã hóa, vì mã hóa RSA liên quan đến phép toán lũy thừa c = m^e mod n.

Cách thực hiện: Sử dụng thuật toán Euclid để kiểm tra ƯCLN(e, φ(n)) = 1. Trong thực tế, các thư viện như JSEncrypt tự động chọn e = 65537 nếu nó thỏa mãn điều kiện. Nếu e = 65537 không phù hợp (hiếm gặp), có thể chọn các số nguyên tố khác như 3, 17, hoặc 257, nhưng điều này có thể làm tăng kích thước của số mũ bí mật d.

Ví dụ: Với φ(n) = 3120, chọn e = 17 (vì ƯCLN(17, 3120) = 1).

5. **Tính d sao cho d × e ≡ 1 (mod φ(n))**

Mô tả: Số mũ bí mật d (private exponent) là nghịch đảo modulo của e theo modulo φ(n), tức là:

$$d \times e = 1 \pmod{\varphi(n)}$$

Điều này có nghĩa là d là số thỏa mãn phương trình:

$$d \times e = k \times \varphi(n) + 1$$

với k là một số nguyên bất kỳ.

Cách thực hiện: Sử dụng thuật toán Euclid mở rộng (Extended Euclidean Algorithm) để tìm d. Thuật toán này tính toán các hệ số Bezout x và y sao cho:

$$x \times e + y \times \varphi(n) = \text{ƯCLN}(e, \varphi(n)) = 1$$

Trong đó, x chính là d (hoặc x + k × φ(n) nếu cần điều chỉnh). Trong các thư viện như JSEncrypt, bước này được thực hiện tự động.

Ví dụ minh họa: Với e = 17, φ(n) = 3120:

- Tìm d sao cho 17 × d ≡ 1 (mod 3120).
- Sử dụng thuật toán Euclid mở rộng, ta được d = 2753 (vì 17 × 2753 = 46801 = 15 × 3120 + 1).

Vai trò: d là thành phần cốt lõi của private key, được sử dụng để giải mã hoặc ký số.

2.3.2. Kết quả

Kết quả của quy trình sinh khóa RSA là hai cặp khóa:

- **Public key: (n, e)**
    - Gồm modulus n và số mũ công khai e (thường là 65537).
    - Public key được chia sẻ công khai với tất cả các bên trong hệ thống nhắn tin, dùng để:
        - Mã hóa khóa TRIPLEDES trước khi gửi cho người nhận.
        - Xác minh chữ ký số (RSA/SHA-256) để đảm bảo danh tính người gửi.
    - Ví dụ: Với n = 3233, e = 17, public key là (3233, 17).
- **Private key: (n, d)**
    - Gồm modulus n (giống public key) và số mũ bí mật d.
    - Private key được giữ bí mật bởi người sở hữu, dùng để:
        - Giải mã khóa TRIPLEDES được mã hóa bởi public key tương ứng.
        - Tạo chữ ký số để xác thực danh tính hoặc tin nhắn.
    - Ví dụ: Với n = 3233, d = 2753, private key là (3233, 2753).

Trong đề tài, cặp khóa RSA được sinh ra cho mỗi người dùng khi tham gia phòng chat. Public key được trao đổi qua kết nối P2P trong giai đoạn handshake, trong khi private key được lưu trữ an toàn tại client để sử dụng trong quá trình giải mã và ký số.

2.3.3. Bảo mật

Bảo mật của thuật toán RSA phụ thuộc vào việc giữ bí mật các thành phần nhạy cảm trong quá trình sinh khóa và sử dụng khóa. Các khía cạnh bảo mật quan trọng bao gồm:

- **Giữ bí mật private key (n, d)**:
    - Private key, đặc biệt là số mũ bí mật d, phải được lưu trữ an toàn tại client (ví dụ: trong bộ nhớ tạm hoặc file được mã hóa). Nếu d bị lộ, kẻ tấn công có thể

giải mã tất cả các tin nhắn được mã hóa bằng public key tương ứng hoặc giả mạo chữ ký số.

- Trong hệ thống nhắn tin, private key được tạo và lưu tại thiết bị của người dùng, không bao giờ được truyền qua mạng. Các thư viện như JSEncrypt hỗ trợ lưu trữ private key trong bộ nhớ trình duyệt hoặc thiết bị của người dùng.

- **Giữ bí mật p và q**:
  - Hai số nguyên tố p và q là nền tảng của thuật toán RSA. Nếu kẻ tấn công biết được p và q, họ có thể dễ dàng tính toán:
    - $n = p \times q$ (modulus).
    - $\varphi(n) = (p - 1) \times (q - 1)$.
    - Từ e và $\varphi(n)$, tính d bằng thuật toán Euclid mở rộng.
  - Do đó, p và q phải được xóa ngay sau khi sinh khóa, chỉ giữ lại n, e, và d. Trong thực tế, các thư viện RSA tự động xử lý việc này để giảm thiểu rủi ro.
- **Khó khăn của bài toán phân tích nhân tử**:
  - Tính bảo mật của RSA dựa trên bài toán phân tích nhân tử (factorization problem). Với n có độ dài 4096-bit (từ p và q 2048-bit), việc tìm p và q từ n là bất khả thi với công nghệ hiện tại. Các thuật toán như GNFS hoặc máy tính lượng tử (với thuật toán Shor) vẫn chưa đủ mạnh để phá vỡ RSA 2048-bit trong thời gian hợp lý.
  - Tuy nhiên, để tăng cường bảo mật, hệ thống cần sử dụng trình sinh số ngẫu nhiên bảo mật để chọn p và q, tránh các số nguyên tố yếu hoặc dễ đoán.
- **Bảo vệ trong quá trình trao đổi khóa**:
  - Trong hệ thống nhắn tin, public key (n, e) được trao đổi công khai qua kết nối P2P trong giai đoạn handshake. Để ngăn chặn tấn công man-in-the-middle (MITM), hệ thống sử dụng cơ chế xác thực danh tính bằng chữ ký số RSA/SHA-256, đảm bảo rằng public key nhận được thực sự thuộc về người dùng đích thực.

- Khóa TRIPLEDES, được mã hóa bằng public key RSA của người nhận, cũng được bảo vệ để tránh bị chặn hoặc sửa đổi trong quá trình truyền tải.

- **Các biện pháp bổ sung**:
  - Sử dụng padding OAEP (Optimal Asymmetric Encryption Padding) khi mã hóa bằng RSA để chống lại các cuộc tấn công như chosen-ciphertext attack.
  - Kết hợp hàm băm SHA-256 để kiểm tra tính toàn vẹn của dữ liệu, đảm bảo rằng public key hoặc tin nhắn không bị sửa đổi trong quá trình truyền tải.
  - Triển khai các biện pháp bảo mật vật lý và phần mềm, như lưu trữ private key trong môi trường bảo mật (ví dụ: Web Crypto API hoặc secure storage) và sử dụng giao thức bảo mật như TLS/SSL cho kết nối P2P nếu cần.

CHƯƠNG 3. XÂY DỰNG ỨNG DỤNG NHẮN TIN BẢO MẬT

3.1. Ý tưởng và cấu trúc

3.1.1. Mục tiêu

Mục tiêu chính của ứng dụng là xây dựng một hệ thống nhắn tin trực tuyến bảo mật, sử dụng mã hóa TRIPLEDES và xác thực RSA để đảm bảo tính bí mật, xác thực danh tính, và toàn vẹn dữ liệu. Cụ thể, các mục tiêu bao gồm:

- Bảo mật nội dung tin nhắn: Sử dụng thuật toán TRIPLEDES ở chế độ CFB (Cipher Feedback) để mã hóa tin nhắn văn bản, đảm bảo rằng chỉ người nhận có khóa TRIPLEDES đúng mới có thể giải mã và đọc được nội dung. Điều này ngăn chặn các hành vi nghe lén (eavesdropping) trong quá trình truyền tải qua mạng.

- Xác thực danh tính người dùng: Áp dụng RSA 2048-bit với padding OAEP và hàm băm SHA-256 để ký số ID người gửi và xác minh danh tính, giúp ngăn chặn giả mạo (impersonation) trong các phòng chat riêng tư.

- Trao đổi khóa an toàn: Sử dụng RSA để mã hóa khóa TRIPLEDES trước khi gửi qua kết nối P2P, đảm bảo rằng khóa TRIPLEDES chỉ có thể được giải mã bởi người nhận sở hữu private key tương ứng.

- Kiểm tra toàn vẹn dữ liệu: Tích hợp hàm băm SHA-256 để tạo hash của ciphertext, cho phép người nhận kiểm tra xem tin nhắn có bị sửa đổi trong quá trình truyền tải hay không.

- Tính riêng tư và tiện lợi: Thiết kế các phòng chat riêng tư cho hai người dùng, với định danh phòng ngẫu nhiên hoặc tùy chọn, nhằm đảm bảo tính riêng tư và cung cấp trải nghiệm giao tiếp mượt mà, trực quan thông qua giao diện người dùng thân thiện.

Ứng dụng hướng đến việc cung cấp một giải pháp bảo mật hiệu quả cho các giao tiếp cá nhân hoặc nội bộ, phù hợp với các kịch bản như trao đổi thông tin nhạy cảm trong doanh nghiệp hoặc trò chuyện riêng tư giữa các cá nhân.

3.1.2. Công nghệz

Để triển khai ứng dụng nhắn tin bảo mật, các công nghệ và thư viện sau được sử dụng, dựa trên cấu trúc từ báo cáo cũ và yêu cầu của đề tài:

- Node.js: Nền tảng phía server để xử lý các kết nối thời gian thực và quản lý truyền tải dữ liệu giữa các client. Node.js cung cấp hiệu suất cao và khả năng mở rộng, phù hợp cho các ứng dụng nhắn tin yêu cầu xử lý đồng thời nhiều kết nối.

- Socket.IO: Thư viện hỗ trợ kết nối P2P thời gian thực, được sử dụng để thực hiện các luồng xử lý như handshake ("Hello!" và "Ready!"), trao đổi khóa công khai RSA, và chuyển tiếp tin nhắn mã hóa. Socket.IO đảm bảo giao tiếp hai chiều nhanh chóng và đáng tin cậy giữa server và client.

- Vue.js: Khung giao diện người dùng (front-end framework) để xây dựng giao diện client trực quan, cho phép hiển thị tin nhắn, quản lý phòng chat, và tương tác với các chức năng mã hóa/giải mã. Vue.js được chọn vì tính đơn giản và khả năng tích hợp tốt với các thư viện mã hóa.

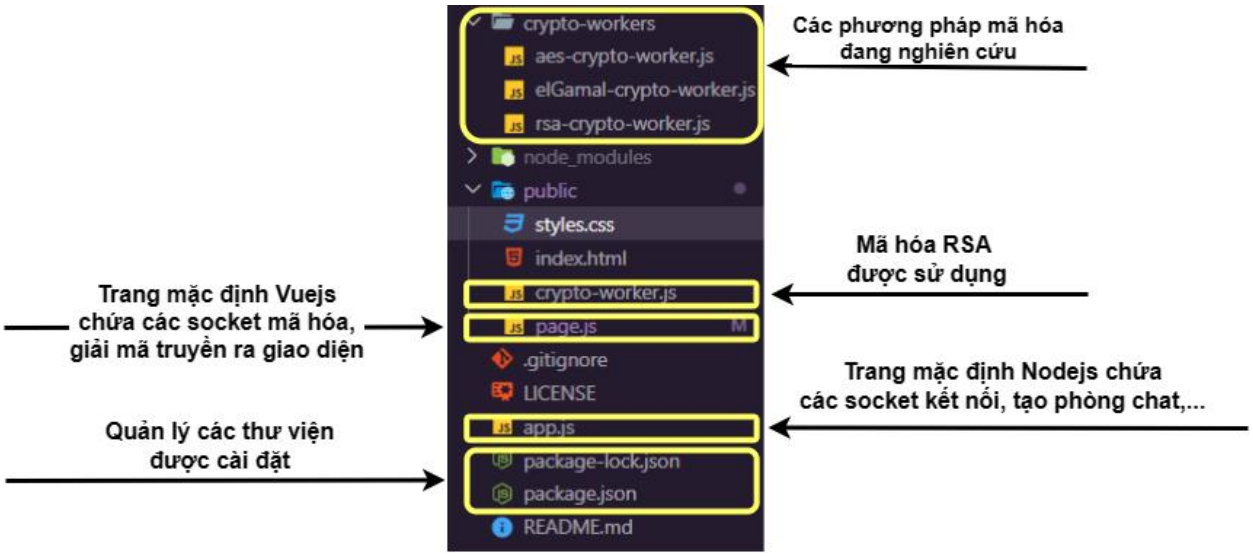
- JSEncrypt: Thư viện JavaScript hỗ trợ triển khai RSA 2048-bit, bao gồm sinh cặp khóa, mã hóa khóa TRIPLEDES bằng public key, ký số với SHA-256, và giải mã bằng private key. JSEncrypt được tích hợp trong một web worker để tối ưu hóa hiệu suất và tránh làm chậm giao diện người dùng.

- Thư viện TRIPLEDES: Một thư viện JavaScript (như crypto-js) được sử dụng để thực hiện mã hóa và giải mã TRIPLEDES ở chế độ CFB. Thư viện này hỗ trợ xử lý khóa 56-bit và vector khởi tạo (IV), đảm bảo tính tương thích với yêu cầu của đề tài.

- SHA-256: Được tích hợp thông qua các thư viện như crypto-js hoặc Web Crypto API để tạo hash của ciphertext, đảm bảo kiểm tra tính toàn vẹn của tin nhắn. SHA-256 cung cấp một hàm băm mạnh, giúp phát hiện bất kỳ thay đổi nào trong dữ liệu truyền tải.

Các công nghệ này được chọn để tận dụng tính tương thích cao với môi trường web, cho phép ứng dụng chạy trực tiếp trên trình duyệt mà không yêu cầu cài đặt phần mềm bổ sung, đồng thời đảm bảo hiệu suất và bảo mật.

3.1.3. Cấu trúc thư mục chương trình



Hình 1. Cấu trúc thư mục chương trình

3.2. Luồng xử lý

Mấu chốt cơ bản của việc sinh khóa trong RSA là tìm được bộ 3 số tự nhiên e, d và n sao cho:

$$m^{ed} \equiv m \mod n$$

và một điểm không thể bỏ qua là cần bảo mật cho d sao cho dù biết e, n hay thậm chí cả m cũng không thể tìm ra d được.

Cụ thể, khóa của RSA được sinh như sau:

- Chọn 2 số nguyên tố p và q
- Tính n = pq. Sau này, n sẽ được dùng làm modulus trong cả public key và private key.
- Tính một số giả nguyên tố bằng phi hàm Carmichael như sau: λ(n) = BCNN(λ(p), λ(q)) = BCNN(p − 1, q − 1). Giá trị này sẽ được giữ bí mật.
- Chọn một số tự nhiên e trong khoảng (1, λ(n)) sao cho ƯCLN(e, λ(n)) = 1, tức là e và λ(n) nguyên tố cùng nhau.
- Tính toán số d sao cho d ≡ 1/e (mod λ(n)) hay viết dễ hiểu hơn thì de ≡ 1 (mod λ(n)). Số d được gọi là số nghịch đảo modulo của e (theo modulo mod λ(n)).

- Public key sẽ là bộ số (n, e), và private key sẽ là bộ số (n, d). Chúng ta cần giữ private key thật cẩn thận cũng như các số nguyên tố p và q vì từ đó có thể tính toán các khóa rất dễ dàng.

Trong thực hành, chúng ta thường chọn e tương đối nhỏ để việc mã hóa và giải mã nhanh chóng hơn. Giá trị thường được sử dụng là e = 65537. Ngoài ra, chúng ta có thể tính số giả nguyên tố bằng phi hàm Euler φ(n) = (p − 1)(q − 1) và dùng nó như λ(n). Vì φ(n) là bội của λ(n) nên các số d thỏa mãn de ≡ 1 (mod φ(n)) cũng sẽ thỏa mãn d ≡ 1/e (mod λ(n)). Tuy nhiên, một số tác dụng phụ của việc này là d thường sẽ trở nên lớn hơn mức cần thiết.

Mã hóa và giải mã

Nếu chúng ta có bản rõ M, chúng ta cần chuyển nó thành một số tự nhiên m trong khoảng (0, n) sao cho m, n nguyên tố cùng nhau. Việc này rất dễ dàng thực hiện bằng cách thêm một các kỹ thuật padding. Tiếp theo, chúng ta sẽ má hóa m, thành c như sau:

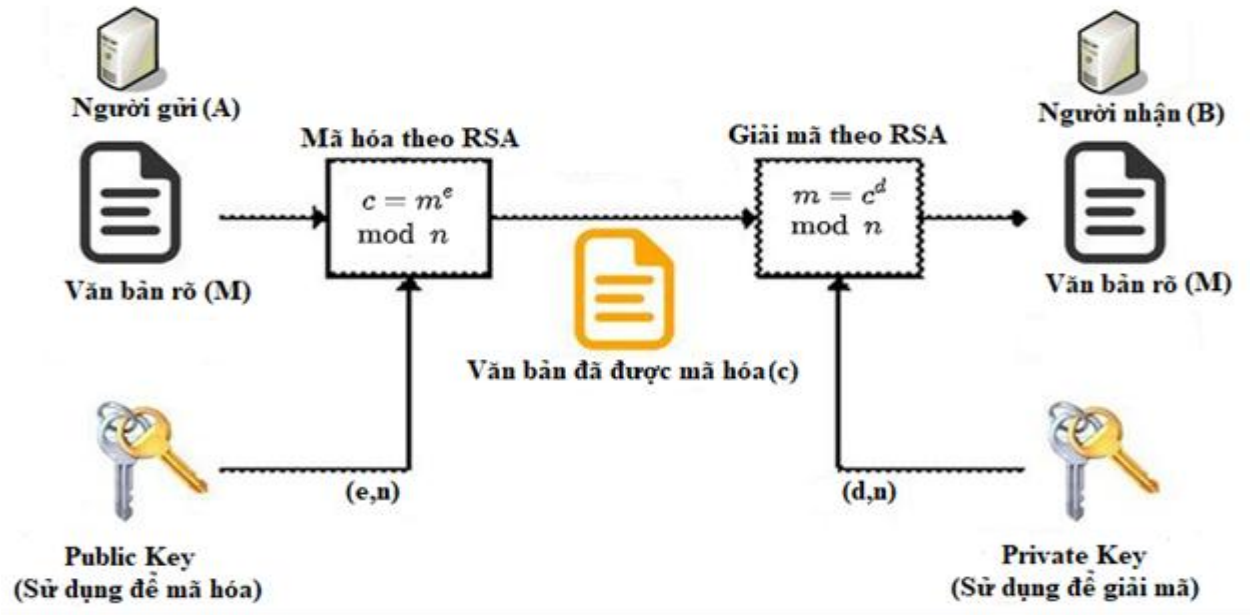
$$c \equiv m^e \mod n$$

Sau đó giá trị c sẽ được chuyển cho người nhận.

Ở phía người nhận, họ sẽ giải mã từ c để lấy được m như sau:

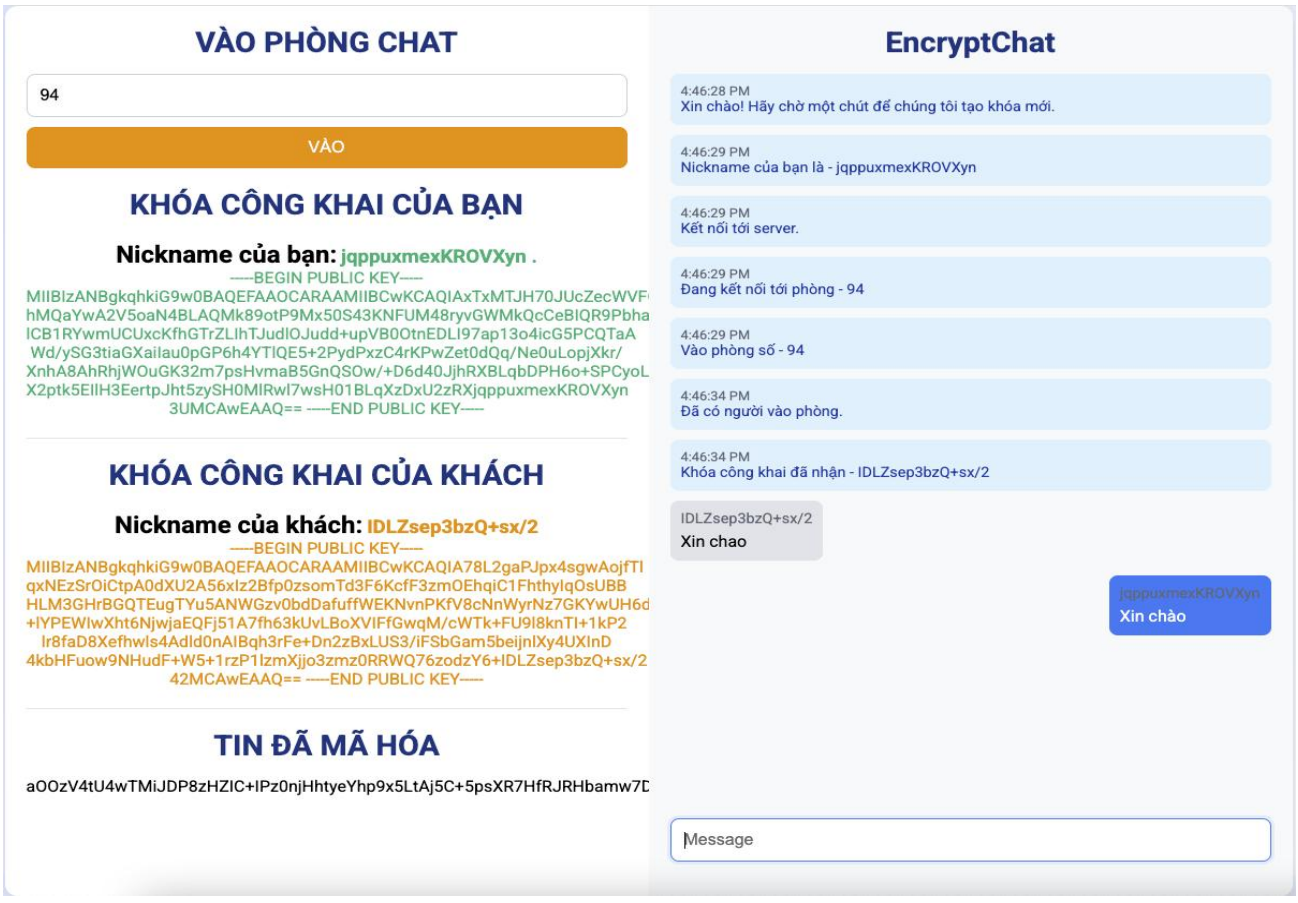
$$c^d \equiv m^{de} \equiv m \mod n$$

Từ m có thể lấy lại được bản tin bằng cách đảo ngược padding



Hình 2. Luồng xử lý

3.3. Kết quả và demo



Hình 3. Giao diện website

KẾT LUẬN

Báo cáo đã trình bày chi tiết quá trình nghiên cứu và triển khai một hệ thống nhắn tin văn bản bảo mật, trong đó sử dụng thuật toán mã hóa TRIPLEDES (Data Encryption Standard) ở chế độ CFB (Cipher Feedback) để đảm bảo tính bí mật của nội dung tin nhắn. Hệ thống kết hợp với thuật toán RSA 2048-bit (sử dụng padding OAEP và hàm băm SHA-256) nhằm xác thực danh tính người dùng và trao đổi khóa an toàn. Bên cạnh đó, việc tích hợp hàm băm SHA-256 giúp kiểm tra tính toàn vẹn của dữ liệu, ngăn chặn các hành vi sửa đổi hoặc giả mạo trong quá trình truyền tải. Hệ thống được thiết kế dựa trên mô hình P2P với các phòng chat riêng tư, đáp ứng nhu cầu giao tiếp an toàn giữa hai người dùng.

Về mặt thành tựu, báo cáo đã xây dựng được cơ sở lý thuyết vững chắc cho hệ thống. Nội dung báo cáo đã cung cấp nền tảng lý thuyết đầy đủ về hai thuật toán mã hóa TRIPLEDES và RSA, làm rõ các đặc điểm, quy trình sinh khóa và cách kết hợp hai thuật toán này để tạo nên một hệ thống bảo mật toàn diện. Việc sử dụng TRIPLEDES ở chế độ CFB giúp đảm bảo tốc độ mã hóa nhanh, thích hợp cho nhắn tin thời gian thực, trong khi RSA đóng vai trò bảo vệ khóa và xác thực người dùng hiệu quả.

Bên cạnh cơ sở lý thuyết, hệ thống nhắn tin bảo mật đã được triển khai thành công trong thực tế. Quá trình xây dựng sử dụng các công nghệ hiện đại như Node.js, Socket.IO, Vue.js, JSEncrypt và thư viện mã hóa TRIPLEDES. Các quy trình xử lý quan trọng như handshake, trao đổi khóa, mã hóa, xác thực và giải mã đều vận hành ổn định, đảm bảo các yếu tố quan trọng gồm tính bí mật, tính xác thực và tính toàn vẹn của dữ liệu.

Hệ thống còn được tích hợp những tính năng nổi bật như hỗ trợ các phòng chat riêng tư với định danh ngẫu nhiên hoặc do người dùng tùy chọn, giao diện đơn giản, dễ sử dụng với chức năng autoscroll, và khả năng phát hiện lỗi trong quá trình xác minh chữ ký hoặc mã băm. Ứng dụng vận hành hiệu quả trên môi trường localhost tại cổng 3000, cho phép hai người dùng giao tiếp bảo mật thông qua các trình duyệt khác nhau mà không gặp sự cố.

Về phương diện yêu cầu kỹ thuật, hệ thống đã hoàn toàn đáp ứng các tiêu chí của đề tài. Cụ thể, hệ thống sử dụng mã hóa TRIPLEDES với chế độ CFB, RSA 2048-bit kết hợp padding OAEP và SHA-256, đồng thời thực hiện kiểm tra toàn vẹn dữ liệu bằng hàm băm SHA-256. Kết quả thử nghiệm cho thấy hệ thống hoạt động đúng như thiết kế, bảo vệ tốt nội

dung tin nhắn khỏi các nguy cơ như nghe lén, giả mạo hoặc bị chỉnh sửa trong quá trình truyền tải.

Đánh giá về hiệu quả, hệ thống đã chứng minh được khả năng bảo mật cao và cung cấp trải nghiệm người dùng mượt mà. TRIPLEDES ở chế độ CFB cho phép mã hóa nhanh, phù hợp với tin nhắn có độ dài linh hoạt. RSA 2048-bit đảm bảo an toàn cho quá trình trao đổi khóa và xác thực danh tính người dùng, trong khi SHA-256 giúp phát hiện kịp thời các thay đổi không mong muốn của dữ liệu, nâng cao độ tin cậy cho hệ thống. Việc sử dụng Vue.js kết hợp Socket.IO tạo nên một nền tảng giao tiếp thời gian thực với độ trễ thấp, mang lại trải nghiệm liên lạc ổn định cho người dùng.

Tuy nhiên, hệ thống vẫn còn một số hạn chế. Thuật toán TRIPLEDES với độ dài khóa 56-bit hiện không còn đảm bảo an toàn trước các cuộc tấn công brute-force hiện đại, mặc dù khóa đã được bảo vệ bằng RSA. Ngoài ra, hệ thống hiện chỉ hỗ trợ nhắn tin P2P trong môi trường localhost, chưa được tối ưu để triển khai trên quy mô lớn hoặc hoạt động qua mạng Internet. Một điểm hạn chế khác là hệ thống chưa tích hợp các giao thức bảo mật mạng như TLS/SSL, vốn rất cần thiết để chống lại các tấn công man-in-the-middle (MITM) trong môi trường thực tế.

Trong tương lai, để nâng cao hiệu quả và mở rộng khả năng ứng dụng của hệ thống, một số hướng phát triển quan trọng được đề xuất. Trước tiên, việc thay thế thuật toán TRIPLEDES bằng các thuật toán mã hóa đối xứng mạnh hơn như AES với độ dài khóa 128-bit hoặc 256-bit sẽ giúp tăng cường khả năng bảo mật trước các cuộc tấn công hiện đại. Bên cạnh đó, tích hợp các giao thức bảo mật mạng như TLS/SSL vào kết nối P2P sẽ bảo vệ dữ liệu truyền tải tốt hơn khi hệ thống được triển khai qua Internet.

Ngoài ra, hệ thống cần được tối ưu hóa để hỗ trợ nhiều phòng chat đồng thời và tăng khả năng chịu tải, mở ra khả năng ứng dụng trong môi trường doanh nghiệp hoặc các giao dịch tài chính. Việc nâng cấp giao diện và trải nghiệm người dùng cũng nên được chú trọng, bao gồm các tính năng như lưu trữ lịch sử trò chuyện được mã hóa, hỗ trợ đa nền tảng (mobile, TRIPLEDESktop) và thông báo đẩy để tăng sự tiện lợi. Một hướng nghiên cứu quan trọng khác là tìm hiểu và áp dụng các thuật toán mã hóa hậu lượng tử (post-quantum

cryptography), nhằm chuẩn bị cho các thách thức bảo mật trong kỷ nguyên máy tính lượng tử.

TÀI LIỆU THAM KHẢO

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

[2] Bruce Schneier, Applied Cryptography: Protocols, Algorithms, and Source Code in C, 2nd Edition, Wiley, 1996.

[3] National Institute of Standards and Technology (NIST), Data Encryption Standard (TRIPLEDES), FIPS Publication 46-3, 1999. https://csrc.nist.gov/publications/detail/fips/46/3/final

[4] Rivest, R. L., Shamir, A., & Adleman, L., "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," Communications of the ACM, vol. 21, no. 2, pp. 120-126, 1978.

[5] RFC 3447, Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1, Internet Engineering Task Force (IETF), 2003. https://www.rfc-editor.org/rfc/rfc3447

[6] Eastlake, D., & Jones, P., "US Secure Hash Algorithm 2 (SHA-256, SHA-384, SHA-512)," RFC 4634, IETF, 2006. https://www.rfc-editor.org/rfc/rfc4634

[7] Vue.js Official Documentation, https://vuejs.org/

[8] Node.js Official Documentation, https://nodejs.org/

[9] Socket.IO Official Documentation, https://socket.io/

[10] JSEncrypt: RSA Encryption in JavaScript, https://github.com/travist/jsencrypt

[11] Trần Văn Tùng, An toàn và bảo mật hệ thống mạng, NXB Bưu điện, 2020.

[12] Nguyễn Văn Minh, Ứng dụng mật mã trong bảo mật dữ liệu truyền thông, Luận văn Thạc sĩ, Trường Đại học Bách Khoa Hà Nội, 2021.