TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO BÀI THỰC NGHIỆM
HỌC PHẦN: AN TOÀN & BẢO MẬT THÔNG TIN

ĐỀ TÀI

XÂY DỰNG CHƯƠNG TRÌNH MÃ HÓA VÀ GIẢI MÃ RSA SỬ DỤNG NGÔN NGỮ JAVA, C#

Giáo viên hướng dẫn: TS. Phạm Văn Hiệp

Nhóm sinh viên thực hiện:

1. Trương Công Mạnh — Mã SV: 2021601910
2. Nguyễn Văn Nguyên — Mã SV: 2020605636
3. Trần Hồng Nhung — Mã SV: 2020608128
4. Nguyễn Nam Phi — Mã SV: 2020606964
5. Nguyễn Thị Thu Phương — Mã SV: 2020602360

Mã Lớp: 20222IT6001009 Nhóm: 13

Hà Nội - Năm 2023

LỜI NÓI ĐẦU

Trước đây khi công nghệ máy tính chưa phát triển, khi nói đến vấn đề an toàn bảo mật thông tin, chúng ta thường hay nghĩ đến các biện pháp nhằm đảm bảo cho thông tin được trao đổi hay cất giữ một cách an toàn và bí mật, chẳng hạn là các biện pháp như: Đóng dấu và ký niêm phong một bức thư để biết rằng lá thư có được chuyển nguyên vẹn đến người nhận hay không, dùng mật mã mã hóa thông điệp để chỉ có người gửi và người nhận hiểu được thông điệp, lưu giữ tài liệu trong các két sắt có khóa tại nơi được bảo vệ nghiêm ngặt.

Ngày nay với sự phát triển của khoa học công nghệ, đặt biệt là sự phát triển của Internet, việc sử dụng máy tính và điện thoại cá nhân càng trở lên rộng rãi, dẫn đến càng nhiều thông tin được lưu trữ trên máy tính và gửi đi trên mạng Internet. Do đó nhu cầu về an toàn và bảo mật thông tin trên máy tính càng nhiều và việc sử dụng mật mã mã hóa càng được phổ biến. Trong đồ án này, nhóm em thực hiện xây dựng chương trình mã hóa và giải mã mật mã hóa công khai RSAbằng C# và Java. Báo cáo này gồm 3 chương:

Chương 1: Tổng quan.

Chương 2: Kết quả nghiên cứu.

Chương 3: Kiến luận và bài học kinh nghiệm.

Trong quá trình thực hiện đề tài nhóm chúng em xin gửi lời cảm ơn chân thành tới thầy Phạm Văn Hiệp giảng viên hướng dẫn nhóm chúng em thực hiện đề tài này. Trong quá trình nghiên cứu và thực hiện đề tài được sự chỉ bảo tận tình của các thầy cô, nhóm chúng em đã cố gắng hết sức để hoàn thiện đề tài. Tuy nhiên chúng em rất mong nhận được sự góp ý của thầy cô và các bạn

Nhóm 13

MỤC LỤC

DANH MỤC HÌNH ẢNH

CHƯƠNG 1. TỔNG QUAN

1.1 Tổng quan về An toàn bảo mật thông tin

1.1.1 Sự cần thiết của An toàn và bảo mật thông tin

Ngày nay, với sự phát triển mạnh mẽ của Công nghệ thông tin và mạng Internet đã giúp cho việc trao đổi thông tin trở nên nhanh gọn, dễ dàng, ngày càng có nhiều thông tin được số hoá trên máy tính và lưu trữ cũng như truyền đi trên mạng Internet. Chính vì vậy nên nhu cầu về an toàn và bảo mật thông tin trên máy tính là hết sức cấp thiết.

1.1.2 An toàn và bảo mật thông tin

1.1.2.1 Khái niệm

An toàn thông tin (Information Security) là việc bảo vệ chống truy nhập, sử dụng, tiết lộ, sửa đổi hoặc phá hủy thông tin một cách trái phép. An toàn thông tin còn bao gồm cả việc đảm bảo an toàn cho các thành phần hoặc hệ thống được sử dụng để quản lý, lưu trữ, xử lý và trao đổi thông tin.

Hệ thống thông tin an toàn là hệ thống đảm bảo an toàn thông tin, đảm bảo hệ thống có khả năng hoạt động liên tục và đảm bảo khả năng phục hồi.

Bảo mật hệ thống thông tin là đảm bảo tính bí mật, tính toàn vẹn và tính sẵn sàng của hệ thống thông tin.

1.1.2.2 Các loại hình tấn công

Định nghĩa chung: Tấn công là hoạt động có chủ ý của kẻ gian muốn lợi dụng các thuong tổn của hệ thống thông tin và tiến hành phá vỡ tính bí mật, tính toàn vẹn và tính sẵng sàng của hệ thống thông tin.

Các loại hình tấn công phổ biến:

- Xem trộm thông tin: kẻ gian chặn các thông điệp và xem được nội dung của thông tin.
- Thay đổi thông điệp: kẻ gian chặn các thông điệp và thay đổi nội dung của thông tin.
- Mạo danh: kẻ gian giả danh là người gửi để gửi thông tin.

- Phát lại thông điệp: kẻ gian sao chép lại thông tin và gửi đi.

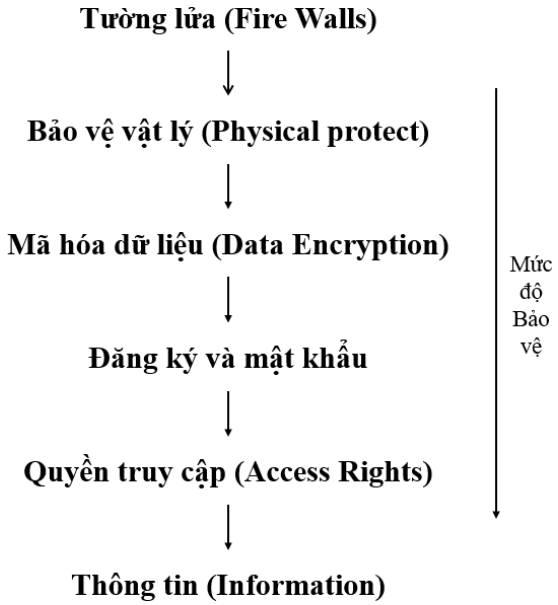
1.1.2.3 Các yêu cầu an toàn bảo mật thông tin

Ngày nay, với sự phát triển rất nhanh của khoa học công nghệ, các biện pháp tấn công ngày càng tinh xảo hơn, độ an toàn của thông tin có thể bị đe dọa từ nhiều nơi, theo nhiều cách khác nhau, chúng ta cần phải đưa ra các chính sách đề phòng thích hợp. Các yêu cầu cần thiết của việc bảo vệ thông tin và tài nguyên:

Hình 1.1 Các yêu cầu an toàn bảo mật thông tin.

- **Đảm bảo tính bảo mật (Confidentiality):** Ngăn ngừa việc tiết lộ trái phép thông tin quan trọng với người không được phép. Thông tin được che giấu với người không được cấp phép.
- **Đảm bảo tính toàn vẹn (Integrity):** Thông tin và tài nguyên không thể bị sửa đổi, bị thay thế bởi những người không có quyền hạn.
- **Đảm bảo tính sẵn sàng (Availability):** Thông tin và tài nguyên luôn sẵn sàng để đáp ứng sử dụng cho người có quyền hạn.

1.1.2.4 Các mức độ bảo vệ



Hình 1.2 Các mức độ bảo vệ thông tin.

Tường lửa: Ngăn chặn thâm nhập trái phép và lọc bỏ các gói tin không muốn gửi hoặc nhận vì các lý do nào đó để bảo vệ một máy tính hoặc cả mạng nội bộ (intranet).

Bảo vệ vật lý: Ngăn cản các truy nhập vật lý vào hệ thống.

Mã hoá dữ liệu: Dữ liệu bị biến đổi từ dạng nhận thức được sang dạng không nhận thức được theo một thuật toán nào đó và sẽ được biến đổi ngược lại ở trạm nhận (giải mã).

Đăng ký và mật khẩu: Thực ra đây cũng là kiểm soát quyền truy nhập, nhưng không phải truy nhập ở mức thông tin mà ở mức hệ thống.

Quyền truy nhập: Là lớp bảo vệ trong cùng nhằm kiểm soát các tài nguyên của mạng và quyền hạn trên tài nguyên đó.

1.1.3 Các phương pháp mã hóa cổ điển

1.1.3.1 An toàn thông tin bằng mật mã

Mật mã là một ngành khoa học chuyên nghiên cứu các phương pháp truyền tin bí mật. Mật mã bao gồm lập mã và phá mã.

- **Lập mã** bao gồm mã hóa và giải mã. Sản phẩm là các hệ mã mật, các hàm băm, các hệ chữ ký điện tử, các cơ chế phân phối, quản lý khóa và các giao thức mật mã.
- **Phá mã** bao gồm phá mã hoặc tạo mã giả. Sản phẩm là các phương pháp phá mã, các phương pháp giả mạo chữ ký, các phương pháp tấn công các hàm băm và các giao thức mật mã.

1.1.3.2 Hệ mật mã

❖ **Vai trò của hệ mật mã:**

- Hệ mật mã phải che dấu được nội dung của văn bản rõ (PlainText).
- Tạo các yếu tố xác thực thông tin, đảm bảo thông tin lưu hành trong hệ thông đến người nhận hợp pháp là xác thực (Authenticity).
- Tổ chức các sơ đồ chữ ký điện tử, đảm bảo không có hiện tượng giả mạo, mạo danh để gửi thông tin trên mạng.

❖ **Khái niệm cơ bản:**

- Bản rõ X được gọi là là bản tin gốc. Bản rõ có thể được chia nhỏ có kích thước phù hợp.
- Bản mã Y là bản tin gốc đã được mã hoá. Ở đây ta thường xét phương pháp mã hóa mà không làm thay đổi kích thước của bản rõ, tức là chúng có cùng độ dài.
- Mã là thuật toán E chuyển bản rõ thành bản mã. Thông thường chúng ta cần thuật toán mã hóa mạnh, cho dù kẻ thù biết được thuật toán, nhưng không biết thông tin về khóa cũng không tìm được bản rõ.
- Khoá K là thông tin tham số dùng để mã hoá, chỉ có người gửi và người nhận biết. Khóa là độc lập với bản rõ và có độ dài phù hợp với yêu cầu bảo mật.

- Mã hoá là quá trình chuyển bản rõ thành bản mã, thông thường bao gồm việc áp dụng thuật toán mã hóa và một số quá trình xử lý thông tin kèm theo.
- Giải mã là quá trình chuyển bản mã thành bản rõ, đây là quá trình ngược lại của mã hóa.

❖ **Một hệ mã mật là bộ 5 (P, C, K, E, D) thoả mãn các điều kiện sau:**

- P là không gian bản rõ: là tập hữu hạn các bản rõ có thể có.
- C là không gian bản mã: là tập hữu hạn các bản mã có thể có.
- K là kkhông gian khoá: là tập hữu hạn các khoá có thế có.
    - Đối với mỗi k ∈ K có một quy tắc mã: eK: P → C và một quy tắc giải mã tương ứng dK: C → P.
    - Với mỗi eK: P → C và dK: C → P là những hàm mà dK(eK(x)) = x với mọi bản rõ x ∈ P.
    - Hàm giải mã dK chính là ánh xạ ngược của hàm mã hóa eK.

❖ **Phân loại hệ mật mã:**

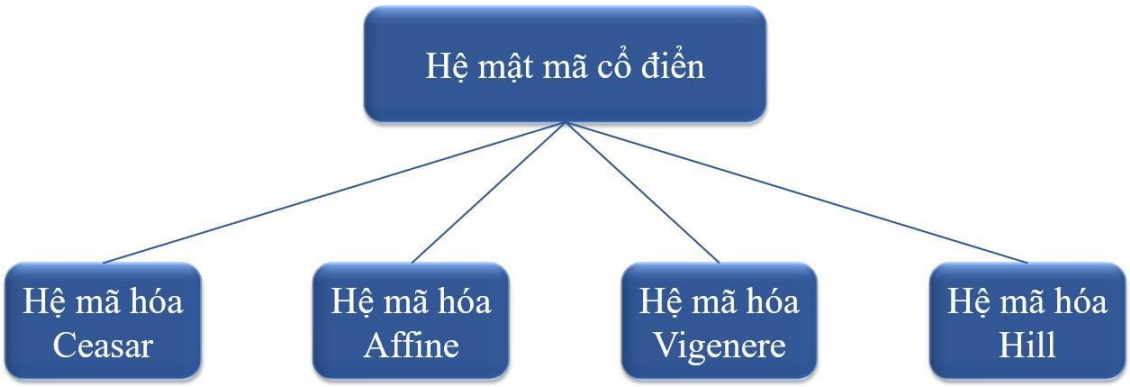
- Hệ mật mã đối xứng (hay còn gọi là mật mã khóa bí mật): là những hệ mật dùng chung một khoá cả trong quá trình mã hoá dữ liệu và giải mã dữ liệu. Do đó khoá phải được giữ bí mật tuyệt đối. Một số thuật toát nổi tiếng trong mã hoá đối xứng là: DES, Triple DES(3DES), RC4, AES...
- Hệ mật mã bất đối xứng (hay còn gọi là mật mã khóa công khai):
    - Các hệ mật này dùng một khoá để mã hoá sau đó dùng một khoá khác để giải mã, nghĩa là khoá để mã hoá và giải mã là khác nhau.
    - Các khoá này tạo nên từng cặp chuyển đổi ngược nhau và không có khoá nào có thể suy được từ khoá kia. Khoá dùng để mã hoá có thể công khai - Public Key nhưng khoá dùng

để giải mã phải giữ bí mật - Private Key. Một số thuật toán mã hoá công khai nổi tiếng: Diffle-Hellman, RSA...

❖ **Có ba phương pháp chính cho việc mã hoá và giải mã:**

- Sử dụng khóa đối xứng.
- Sử dụng khóa bất đối xứng.
- Sử dụng hàm băm một chiều.

❖ **Các hệ mật mã cổ điển:**



Hình 1.3 Các hệ mật mã cổ điển

1.2 Lý do chọn đề tài

Hiện nay, việc mã hóa dữ liệu là một yếu tố vô cùng quan trọng, có nhiều phương pháp mã hóa như Mã hóa cổ điển, mã hóa một chiều (hash), mã hóa đối xứng (symmetric key encryption), mã hóa bất đối xứng (public key encryption). Trong phương pháp mã hóa bất đối xứng nổi bật là phương pháp mã hóa RSA- một thuật toán mật mã hóa khóa công khai. Đây là thuật toán đầu tiên phù hợp với việc tạo ra chữ ký điện tử đồng thời với việc mã hóa. Nó đánh dấu một sự tiến bộ vượt bậc của lĩnh vực mật mã học trong việc sử dụng khóa công cộng. Nhận thấy tính ứng dụng cao của phương pháp này, nhóm 13 chúng em quyết định lựa chọn tìm hiểu đề tài "Xây dựng chương trình mã hóa và giải mã RSA bằng C#, Java".

Thông qua đề tài, chúng em mong muốn:

- Tạo cơ hội để hiểu sâu hơn về các thuật toán mã hóa bất đối xứng và các ứng dụng thực tế của chúng.

- Củng cố sự hiểu biết về các thành phần chính, thuật toán và quy trình liên quan đến mã hóa và giải mã RSA.
- Chương trình có thể đóng góp vào sự phát triển của các hệ thống liên lạc an toàn và giải quyết những lo ngại về mã hóa dữ liệu ngày nay.

Ngoài ra, nhóm hy vọng sẽ tích lũy được thêm kiến thức về bảo mật dữ liệu, cũng như tinh thần làm việc nhóm, các kĩ năng sử dụng công cụ hỗ trợ và các ngôn ngữ khác nhau.

1.3 Nội dung nghiên cứu

Nội dung nghiên cứu bao gồm:

- Tìm hiểu mã hóa công khai
- Nghiên cứu, tìm hiểu về hệ mật mã RSA
- Tìm hiểu nội dung các thuật toán được ứng dụng trong chương trình
- Thiết kế chương trình, cài đặt thuật toán

1.4 Các kiến thức cơ sở

Để thực hiện được đề tài này nhóm chúng em đã phải dựa trên nhiều kiến thức về thuật toán như:

- Định lý Euclid,
- Thuật toán Euclid mở rộng
- Định lí Fermat
- Hàm số Euler
- Thuật toán Miller-Rabin
- Thuật toán Bình phương và nhân
- Sử dụng một số ngôn ngữ khá phổ biến hiện nay như C#, Java

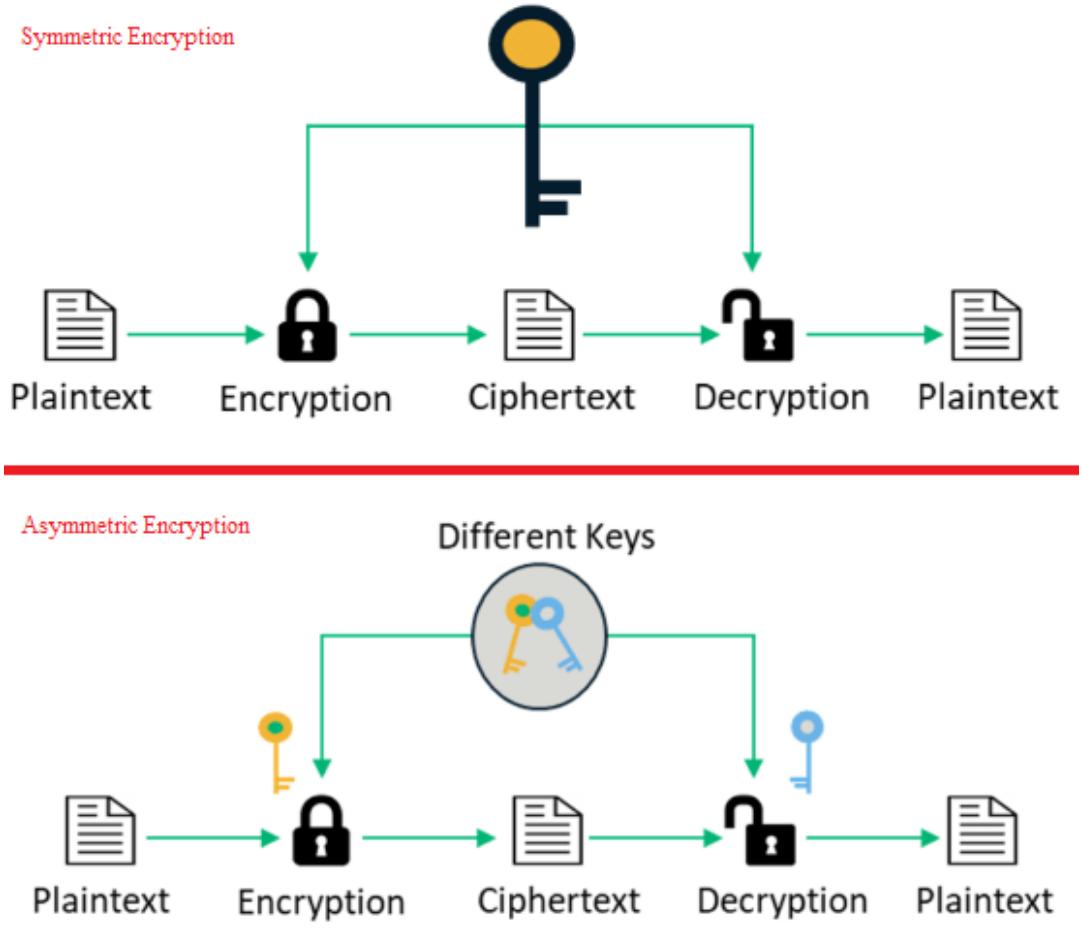
Ngoài ra chúng em còn phải nghiên cứu về phương pháp tạo khóa, mã hóa và giải mã RSA một cách thành thạo (cả tiếng việt và tiếng anh). Bên cạnh đó chúng em cũng nỗ lực xây dựng chương trình có tính sát với thực tiễn nhất nhằm phục vụ cho cuộc sống.

CHƯƠNG 2. KẾT QUẢ NGHIÊN CỨU

2.1 Nghiên cứu, tìm hiểu hệ mã hóa công khai

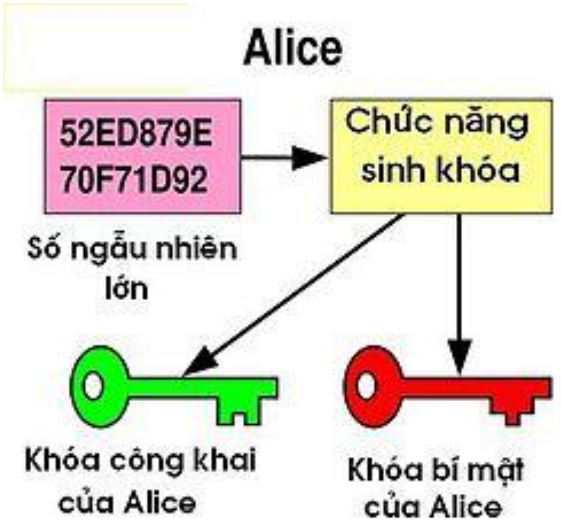
2.1.1 Giới thiệu về hệ mã hóa công khai

Trong hầu hết lịch sử mật mã học, khóa dùng trong các quá trình mã hóa và giải mã phải được giữ bí mật và cần được trao đổi bằng một phương pháp an toàn khác (không dùng mật mã) như gặp nhau trực tiếp hay thông qua một người đưa thư tin cậy. Vì vậy quá trình phân phối khóa trong thực tế gặp rất nhiều khó khăn, đặc biệt là khi số lượng người sử dụng rất lớn. Mật mã hóa khóa công khai đã giải quyết được vấn đề này vì nó cho phép người dùng gửi thông tin mật trên đường truyền không an toàn mà không cần thỏa thuận khóa từ trước.



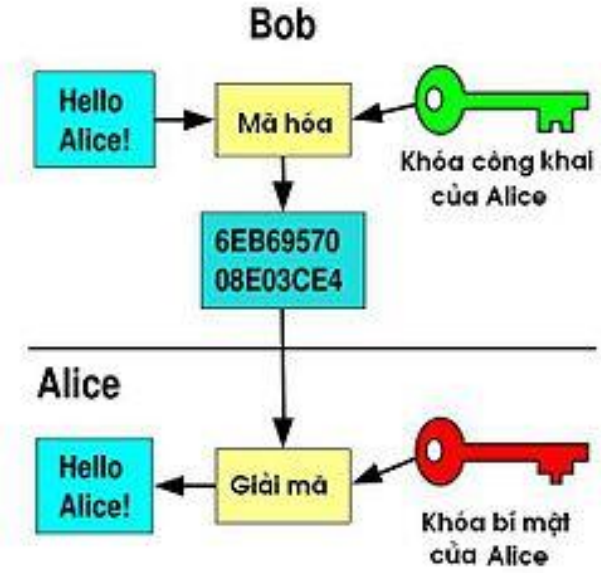
Hình 2.1 Mã khóa đối xứng và mã hóa bất đối xứng

Mật mã hóa khóa công khai là một dạng mật mã hóa cho phép người sử dụng trao đổi các thông tin mật mà không cần phải trao đổi các khóa chung bí mật trước đó. Điều này được thực hiện bằng cách sử dụng một cặp khóa có quan hệ toán học với nhau là khóa công khai và khóa cá nhân (hay khóa bí mật).



Hình 2.2 Chọn một số ngẫu nhiên lớn để sinh cặp khóa

Trong mật mã hóa khóa công khai, khóa cá nhân phải được giữ bí mật trong khi khóa công khai được phổ biến công khai. Trong 2 khóa, một dùng để mã hóa và khóa còn lại dùng để giải mã. Điều quan trọng đối với hệ thống là không thể tìm ra khóa bí mật nếu chỉ biết khóa công khai.

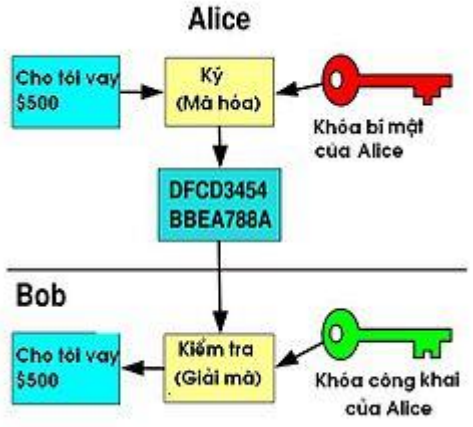


Hình 2.3 Dùng khóa công khai để mã hóa, khóa bí mật để giải mã

2.1.2 Ứng dụng của hệ mã hóa khóa công khai

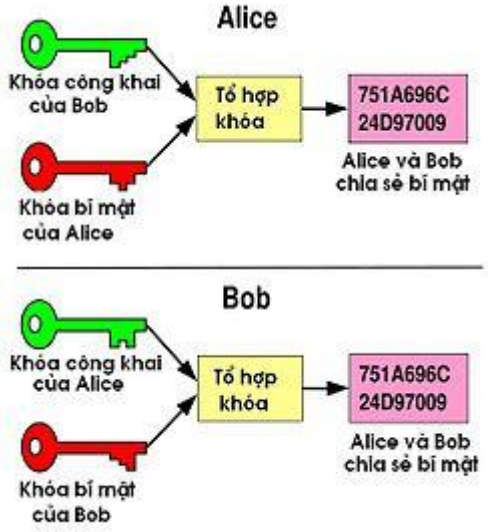
Hệ thống mật mã hóa khóa công khai có thể sử dụng với các mục đích:

- **Mã hóa**: giữ bí mật thông tin và chỉ có người có khóa bí mật mới giải mã được.
- **Tạo chữ ký số**: cho phép kiểm tra một văn bản có phải đã được tạo với một khóa bí mật nào đó hay không.



Hình 2.4 Dùng khoá bí mật để ký một thông báo; khoá công khai để xác minh chữ ký.

- **Thỏa thuận khóa**: cho phép thiết lập khóa dùng để trao đổi thông tin mật giữa 2 bên.



Hình 2.5 Tổ hợp khoá bí mật mình với khoá công khai của người khác tạo ra khoá dùng chung

Thông thường, các kỹ thuật mật mã hóa khóa công khai đòi hỏi khối lượng tính toán nhiều hơn các kỹ thuật mã hóa khóa đối xứng nhưng những lợi điểm mà chúng mang lại khiến cho chúng được áp dụng trong nhiều ứng dụng.

2.1.3 Ưu điểm và sự khác biệt so với mã hóa đối xứng

Ưu điểm chính của mật mã khóa công khai nằm ở khả năng giải quyết vấn đề phân phối khóa cố hữu trong mật mã khóa đối xứng. Trong các hệ thống khóa đối xứng, việc phân phối an toàn các khóa bí mật giữa các bên giao tiếp có thể là một thách thức, đặc biệt là trong các mạng quy mô lớn. Mật mã khóa công khai loại bỏ nhu cầu trao đổi khóa an toàn, vì mỗi người dùng sở hữu cặp khóa duy nhất của riêng họ.

Một ưu điểm khác của mật mã khóa công khai là tính linh hoạt của nó trong việc hỗ trợ các hoạt động mật mã khác nhau. Nó không chỉ cho phép liên lạc an toàn thông qua mã hóa và giải mã mà còn hỗ trợ các chức năng thiết yếu khác, chẳng hạn như chữ ký số và giao thức trao đổi khóa.

2.1.4 Vai trò của mã hóa bất đối xứng trong hệ thống khóa công khai

Mã hóa bất đối xứng, một thành phần cơ bản của mật mã khóa công khai, đóng một vai trò quan trọng trong các hệ thống khóa công khai. Mã hóa bằng khóa chung đảm bảo rằng chỉ người nắm giữ khóa riêng tương ứng mới có thể giải mã bản mã. Quá trình này cung cấp tính bảo mật, vì chỉ người nhận dự định mới sở hữu khóa riêng cần thiết để giải mã.

Hơn nữa, bản chất bất đối xứng của mật mã khóa công khai cho phép thực hiện chữ ký số. Bằng cách sử dụng khóa riêng để ký một tin nhắn, người gửi có thể cung cấp khả năng xác thực và không từ chối, vì chữ ký có thể được xác minh bằng khóa chung tương ứng. Điều này đảm bảo tính toàn vẹn và tính xác thực của tin nhắn, vì bất kỳ hành vi giả mạo hoặc giả mạo nào cũng sẽ dẫn đến chữ ký không hợp lệ.

2.2 Nghiên cứu tìm hiểu về hệ mật mã RSA

2.2.1 Sơ lược về hệ mật RSA

Trong mật mã học, RSA là một thuật toán mật mã hóa khóa công khai. Đây là thuật toán đầu tiên phù hợp với việc tạo ra chữ ký điện tử đồng thời với việc mã hóa. Nó đánh dấu một sự tiến bộ vượt bậc của lĩnh vực mật mã học trong việc sử dụng khóa công cộng. RSA đang được sử dụng phổ biến trong thương mại điện tử và được cho là đảm bảo an toàn với điều kiện độ dài khóa đủ lớn.

Hệ mã RSA được đặt tên dựa theo các chữ cái đầu của 3 tác giả của hệ mã là Rivest, Shamir và Adleman. Đây là thuật toán mã hóa nổi tiếng nhất và cũng là thuật toán được ứng dụng thực tế nhất.

RSA được xây dựng dựa trên tính khó của bài toán phân tích các số lớn ra thừa số nguyên tố: Biết một số nguyên tố nhân chúng với nhau để thu được một hợp số là dễ còn biết hợp số phân tích nó ra thừa số nguyên tố là khó.

2.2.2 Bối cảnh lịch sử

Bối cảnh lịch sử của RSA bắt nguồn từ những nỗ lực ban đầu về mật mã. Trước RSA, hầu hết các hệ thống mật mã đều dựa trên các thuật toán khóa đối xứng, trong đó cùng một khóa được sử dụng cho cả mã hóa và giải mã. Tuy nhiên, việc phân phối và quản lý khóa bí mật một cách an toàn giữa các bên liên lạc đặt ra những thách thức đáng kể.

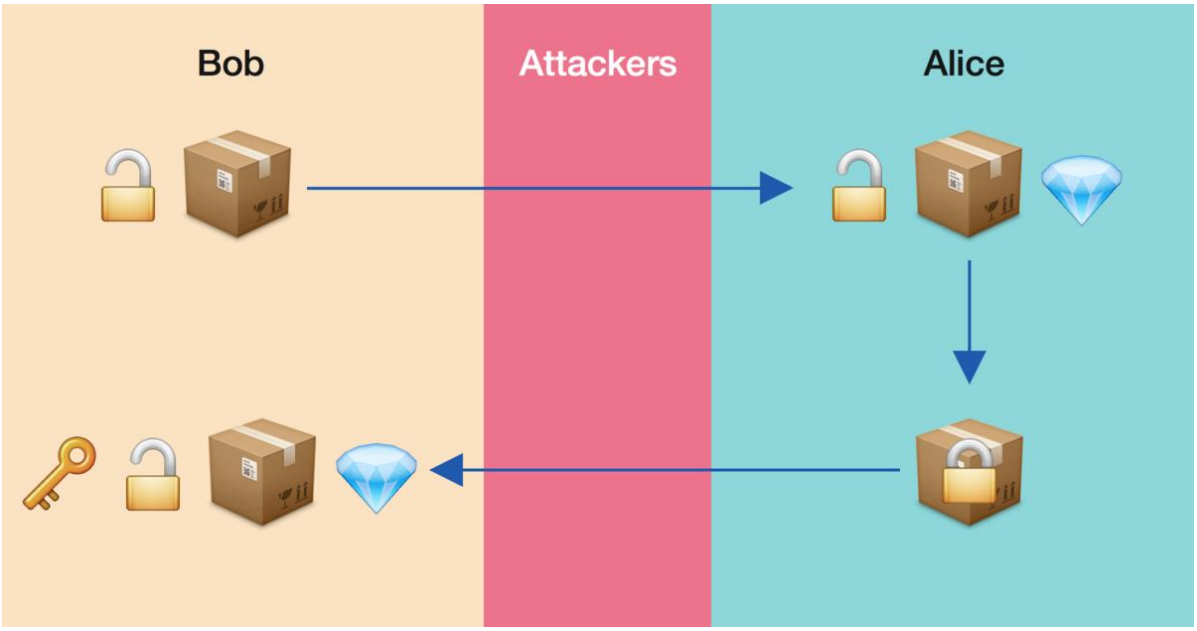
Bước đột phá của hệ thống mật mã RSA là khái niệm mã hóa bất đối xứng, trong đó một cặp khóa liên quan đến toán học được sử dụng. Nó loại bỏ nhu cầu trao đổi khóa an toàn, vì khóa chung có thể được chia sẻ miễn phí trong khi khóa riêng vẫn được giữ bí mật. Khái niệm này không chỉ đơn giản hóa quy trình quản lý khóa mà còn mở đường cho nhiều ứng dụng trong giao tiếp an toàn và chữ ký số.

Kể từ khi được giới thiệu, hệ thống mật mã RSA đã được áp dụng rộng rãi và trở thành nền tảng của mật mã học hiện đại. Nó tạo cơ sở cho nhiều giao

thức và hệ thống bảo mật, từ các giao dịch trực tuyến an toàn đến liên lạc email an toàn. Tính bảo mật và độ tin cậy của RSA phụ thuộc vào độ khó của việc phân tích các số tổng hợp lớn thành các thừa số nguyên tố của chúng, đây vẫn là một bài toán chưa giải được đối với các khóa đủ lớn.

2.2.3 Mô tả sơ lược hoạt động

Thuật toán RSA có hai khóa: khóa công khai (hay khóa công cộng) và khóa bí mật (hay khóa cá nhân). Mỗi khóa là những số cố định sử dụng trong quá trình mã hóa và giải mã. Khóa công khai được công bố rộng rãi cho mọi người và được dùng để mã hóa. Những thông tin được mã hóa bằng khóa công khai chỉ có thể được giải mã bằng khóa bí mật tương ứng. Nói cách khác, mọi người đều có thể mã hóa nhưng chỉ có người biết khóa cá nhân (bí mật) mới có thể giải mã được.



Hình 2.6 Minh họa ý tưởng Hệ mật RSA

Ta có thể mô phỏng trực quan một hệ mật mã khoá công khai như sau: Bình muốn gửi cho An một thông tin mật mà Bình muốn duy nhất An có thể đọc được. Để làm được điều này, An gửi cho Bình một chiếc hộp có khóa đã mở sẵn và giữ lại chìa khóa. Bình nhận chiếc hộp, cho vào đó một tờ giấy viết thư bình thường và khóa lại (như loại khoá thông thường chỉ cần sập chốt lại,

sau khi sập chốt khóa ngay cả Bình cũng không thể mở lại được-không đọc lại hay sửa thông tin trong thư được nữa). Sau đó Bình gửi chiếc hộp lại cho An. An mở hộp với chìa khóa của mình và đọc thông tin trong thư. Trong ví dụ này, chiếc hộp với khóa mở đóng vai trò khóa công khai, chiếc chìa khóa chính là khóa bí mật.

2.2.4 Các thành phần chính của RSA

2.2.4.1 Tạo khóa

Quá trình tạo khóa công khai và khóa riêng trong RSA bao gồm các bước sau:

1. Chọn Hai Số Nguyên Tố Lớn: Hai số nguyên tố khác nhau, p và q, được chọn. Các số nguyên tố này phải lớn, thường có hàng trăm chữ số, để đảm bảo tính bảo mật của mã hóa RSA.
2. Tính toán mô đun: Mô đun, n, được tính bằng cách lấy tích của p và q (n = p * q). Mô-đun đóng vai trò là thành phần chung trong cả khóa chung và khóa riêng.
3. Tính toán hàm tổng của Euler: Hàm tổng của Euler (φ(n)) được tính toán, đại diện cho số lượng các số nguyên dương nhỏ hơn n nguyên tố cùng nhau (tương đối nguyên tố) với n. Trong RSA, φ(n) được sử dụng để đảm bảo quá trình mã hóa và giải mã hoạt động bình thường.
4. Chọn Khóa Công khai: Một số mũ công khai, e, được chọn sao cho nó nguyên tố cùng nhau với φ(n) và nằm trong một phạm vi cụ thể. Thông thường, e được chọn là một số nguyên tố nhỏ hoặc một giá trị cố định như 65537 (2^16 + 1) do tính hiệu quả của nó trong tính toán.
5. Tính khóa riêng: Số mũ riêng, d, được xác định bằng Thuật toán Euclide mở rộng hoặc các phương pháp khác. Khóa riêng (d) được tính là nghịch đảo nhân mô-đun của khóa chung (e) đối với giá trị của φ(n).

Khóa chung được tạo bao gồm mô đun (n) và số mũ chung (e), trong khi khóa riêng bao gồm mô đun (n) và số mũ riêng (d).

2.2.4.2 Mã hóa

Để mã hóa bản rõ bằng RSA, các bước sau được thực hiện:

1. Biểu diễn bản rõ: Thông báo bản rõ trước tiên được chuyển đổi thành biểu diễn số. Điều này có thể được thực hiện bằng cách sử dụng các kỹ thuật như bảng mã ASCII hoặc Unicode.
2. Áp dụng lũy thừa mô-đun: Bản mã được tạo bằng cách thực hiện phép lũy thừa mô-đun trên bản rõ bằng khóa chung. Công thức mã hóa là Ciphertext = (Plaintext^e) mod n.

Bản mã kết quả sau đó có thể được truyền an toàn đến người nhận dự kiến.

2.2.4.3 Giải mã

Người nhận bản mã sử dụng khóa riêng để giải mã tin nhắn. Quá trình giải mã bao gồm các bước sau:

1. Lấy bản mã: Người nhận nhận được bản mã, được tạo bằng khóa công khai của người gửi.
2. Áp dụng lũy thừa mô-đun: Bản mã được giải mã bằng cách thực hiện phép lũy thừa mô-đun trên bản mã bằng khóa riêng. Công thức giải mã là Plaintext = (Ciphertext^d) mod n.
3. Chuyển đổi thành văn bản gốc: Biểu diễn số đã giải mã được chuyển đổi trở lại thành dạng văn bản gốc ban đầu bằng cách sử dụng quy trình ngược lại được áp dụng trong quá trình mã hóa (ví dụ: giải mã ASCII hoặc Unicode).

Bằng cách làm theo các bước này, thông báo văn bản gốc ban đầu được khôi phục và người nhận có thể truy cập được.

2.2.5 Độ an toàn

Độ an toàn của hệ thống RSA dựa trên 2 vấn đề của toán học: bài toán phân tích ra thừa số nguyên tố các số nguyên lớn và bài toán RSA. Nếu 2 bài toán trên là khó (không tìm được thuật toán hiệu quả để giải chúng) thì không thể thực hiện được việc phá mã toàn bộ đối với RSA. Phá mã một phần phải được ngăn chặn bằng các phương pháp chuyển đổi bản rõ an toàn.

Bài toán RSA là bài toán tính căn bậc e môđun n (với n là hợp số): tìm số m sao cho $m^e=c \bmod n$, trong đó (e, n) chính là khóa công khai và c là bản mã. Hiện nay phương pháp triển vọng nhất giải bài toán này là phân tích n ra thừa số nguyên tố. Khi thực hiện được điều này, kẻ tấn công sẽ tìm ra số mũ bí mật d từ khóa công khai và có thể giải mã theo đúng quy trình của thuật toán. Nếu kẻ tấn công tìm được 2 số nguyên tố p và q sao cho: n = pq thì có thể dễ dàng tìm được giá trị (p-1)(q-1) và qua đó xác định d từ e. Chưa có một phương pháp nào được tìm ra trên máy tính để giải bài toán này trong thời gian đa thức (polynomial-time). Tuy nhiên người ta cũng chưa chứng minh được điều ngược lại (sự không tồn tại của thuật toán).

Tại thời điểm năm 2005, số lớn nhất có thể được phân tích ra thừa số nguyên tố có độ dài 663 bít với phương pháp phân tán trong khi khóa của RSA có độ dài từ 1024 tới 2048 bít. Một số chuyên gia cho rằng khóa 1024 bít có thể sớm bị phá vỡ (cũng có nhiều người phản đối việc này). Với khóa 4096 bít thì hầu như không có khả năng bị phá vỡ trong tương lai gần. Do đó, người ta thường cho rằng RSA đảm bảo an toàn với điều kiện n được chọn đủ lớn. Nếu n có độ dài 256 bít hoặc ngắn hơn, nó có thể bị phân tích trong vài giờ với máy tính cá nhân dùng các phần mềm có sẵn. Nếu n có độ dài 512 bít, nó có thể bị phân tích bởi vài trăm máy tính tại thời điểm năm 1999. Một thiết bị lý thuyết có tên là TWIRL do Shamir và Tromer mô tả năm 2003 đã đặt ra câu hỏi về độ an toàn của khóa 1024 bít. Vì vậy hiện nay người ta khuyến cáo sử dụng khóa có độ dài tối thiểu 2048 bít.

2.2.6 Đánh giá về hệ mật RSA

Một số khía cạnh tiêu biểu khi đánh giá về hệ mật RSA:

- **Tốc độ thực hiện:** RSA có tốc độ thực hiện chậm hơn đáng kể so với các thuật toán mã hóa đối xứng.
- **Chiều dài khóa:** Để đảm bảo an toàn, số n cần phải có kích thước lớn hơn hoặc bằng 512 bit. Năm 2006 hệ mật RSA được cho là hiệu quả với kích thước n phải từ 1024. Trong những năm tiếp theo, chiều dài n phải từ 2048 bits trở nên.
- **Độ phức tạp tính toán**: Độ phức tạp tính toán của RSA tương đối cao so với các thuật toán mã hóa đối xứng. Các hoạt động liên quan đến RSA, chẳng hạn như lũy thừa mô-đun, có thể đòi hỏi nhiều tính toán, đặc biệt đối với các kích thước khóa lớn hơn. Điều này có thể ảnh hưởng đến hiệu suất tổng thể và tốc độ thực thi của các quy trình mã hóa và giải mã dựa trên RSA.
- **Quản lý và lưu trữ khóa**: RSA yêu cầu quản lý và lưu trữ an toàn cả khóa chung và khóa riêng. Khóa riêng phải được giữ bí mật và được bảo vệ khỏi truy cập trái phép, trong khi khóa chung cần được cung cấp cho người nhận dự định. Thực hành quản lý khóa phù hợp, bao gồm lưu trữ khóa an toàn, sao lưu và xoay vòng, là điều cần thiết để đảm bảo tính bảo mật và tính toàn vẹn của hệ thống mật mã RSA.
- **Tiêu thụ tài nguyên**: Các hoạt động RSA, đặc biệt đối với các kích thước khóa lớn hơn, có thể tiêu tốn các tài nguyên tính toán đáng kể, bao gồm cả nguồn CPU và bộ nhớ. Đây có thể là một sự cân nhắc đối với các thiết bị hoặc hệ thống bị hạn chế về tài nguyên với yêu cầu thông lượng cao. Việc sử dụng RSA trong các tình huống như vậy cần tính đến các tài nguyên sẵn có và tác động hiệu suất tiềm năng.

2.2.7 Ứng dụng của hệ mã hóa RSA trong thực tế

Một số ứng dụng của hệ mã hóa RSA:

1. **Chữ ký số và xác thực**: RSA đóng một vai trò quan trọng trong việc cung cấp xác thực và đảm bảo tính toàn vẹn của nội dung số. Chữ ký số được tạo bằng RSA cho phép người nhận xác minh tính xác thực của người gửi và tính toàn vẹn của tin nhắn hoặc tệp. Điều này rất cần thiết trong các tình huống khác nhau, bao gồm cập nhật phần mềm, giao dịch tài chính, tài liệu pháp lý và liên lạc qua email. Chữ ký số dựa trên RSA cung cấp một phương tiện đáng tin cậy và chống giả mạo để xác minh nguồn gốc và tính toàn vẹn của nội dung số.
2. **Trao đổi khóa và quản lý khóa**: Các giao thức trao đổi khóa dựa trên RSA, chẳng hạn như trao đổi khóa Diffie-Hellman, cho phép liên lạc an toàn giữa hai bên qua một kênh không an toàn. Các giao thức này sử dụng RSA để trao đổi khóa mã hóa đối xứng một cách an toàn, đảm bảo rằng chỉ các bên dự kiến mới sở hữu khóa bí mật dùng chung. Trao đổi khóa dựa trên RSA được sử dụng rộng rãi trong các ứng dụng nhắn tin bảo mật, mạng riêng ảo (VPN) và giao thức vỏ bảo mật (SSH).
3. **Xác thực và truy cập từ xa an toàn**: RSA được sử dụng trong các giải pháp truy cập từ xa, chẳng hạn như Mạng riêng ảo (VPN) và Secure Shell (SSH). Nó cung cấp các cơ chế kiểm soát truy cập và xác thực an toàn, cho phép người dùng được ủy quyền kết nối với mạng hoặc hệ thống từ xa một cách an toàn. Các phương thức xác thực dựa trên RSA, như RSA SecurID, sử dụng xác thực hai yếu tố, kết hợp thứ mà người dùng biết (mã PIN hoặc mật khẩu) với thứ họ có (mã thông báo hoặc ứng dụng dành cho thiết bị di động) để tăng cường bảo mật.

4. **Giao tiếp qua email an toàn**: RSA được sử dụng trong mã hóa email để đảm bảo tính bảo mật và quyền riêng tư của email. Bằng cách mã hóa nội dung email và tệp đính kèm bằng khóa chung của người nhận, chỉ người nhận dự định có khóa riêng tương ứng mới có thể giải mã và truy cập thư. Điều này đặc biệt quan trọng đối với việc trao đổi thông tin nhạy cảm, chẳng hạn như trong lĩnh vực chăm sóc sức khỏe, luật pháp và chính phủ.
5. **Giao tiếp an toàn trên Internet vạn vật (IoT):** Với sự phát triển của các thiết bị IoT, RSA được sử dụng để cung cấp giao tiếp an toàn và bảo vệ dữ liệu trong các mạng IoT. Bằng cách mã hóa dữ liệu được truyền giữa các thiết bị IoT và cổng hoặc dịch vụ đám mây bằng RSA, tính riêng tư và tính toàn vẹn của dữ liệu IoT được đảm bảo, giảm thiểu nguy cơ truy cập hoặc thao túng trái phép.

2.3 Nội dung thuật toán

2.3.1 Kiểm tra số nguyên tố

Để kiểm tra một số nguyên có phải là số nguyên tố hay không, có thể sử dụng nhiều phương pháp khác nhau. Dưới đây là một số phương pháp kiểm tra số nguyên tố phổ biến:

1. Kiểm tra tất cả các ước số: Phương pháp này kiểm tra tất cả các ước số của số đó. Nếu số đó chỉ có hai ước số là 1 và chính nó, thì nó là số nguyên tố. Tuy nhiên, phương pháp này không hiệu quả khi số đó rất lớn.
2. Kiểm tra các ước số nguyên tố: Phương pháp này chỉ kiểm tra các ước số nguyên tố của số đó. Nếu số đó chỉ có hai ước số nguyên tố là 1 và chính nó, thì nó là số nguyên tố. Phương pháp này hiệu quả hơn phương pháp trên, nhưng vẫn không hiệu quả với các số rất lớn.
3. Kiểm tra theo định lý Fermat: Định lý Fermat nói rằng nếu p là số nguyên tố và a là một số nguyên tố cùng nhau với p, thì a^(p-1) ≡ 1 (mod p). Phương pháp này chọn một số nguyên tố a và tính toán a^(p-1) mod p. Nếu kết quả là 1, thì số đó có thể là số nguyên tố. Tuy nhiên, phương pháp này không đảm bảo chắc chắn rằng số đó là số nguyên tố.
4. Kiểm tra theo định lý Miller-Rabin: Định lý Miller-Rabin nói rằng nếu p là số nguyên tố, thì với mọi số nguyên a, 1 ≤ a ≤ p-1, thì entweder a^(p-1) ≡ 1 (mod p) hoặc tồn tại j, 0 ≤ j ≤ s-1, sao cho a^(2^j * (p-1)) ≡ -1 (mod p). Phương pháp này chọn một số nguyên tố a và kiểm tra định lý Miller-Rabin với số đó. Nếu kết quả là đúng, thì số đó có thể là số nguyên tố. Phương pháp này hiệu quả hơn các phương pháp trên và được sử dụng rộng rãi để kiểm tra số nguyên tố.

2.3.2 Thuật toán Miller-Rabin

Thuật toán Miller-Rabin là một thuật toán xác định tính nguyên tố của một số nguyên dương n. Nó là một thuật toán xác suất, có nghĩa là kết quả của nó có thể không chính xác với xác suất rất nhỏ, nhưng trong hầu hết các trường hợp, kết quả sẽ là chính xác.

Thuật toán Miller-Rabin hoạt động dựa trên một định lý trong lý thuyết số, gọi là Định lý Fermat nhỏ. Định lý này nói rằng nếu p là một số nguyên tố và a là một số nguyên không chia hết cho p, thì a^(p-1) ≡ 1 (mod p).

Thuật toán Miller-Rabin sử dụng phương pháp kiểm tra xem một số nguyên dương n có phải là số nguyên tố hay không bằng cách chọn ngẫu nhiên một số a trong khoảng [1, n-1] và kiểm tra xem a có thỏa mãn định lý Fermat nhỏ không. Nếu a không thỏa mãn định lý Fermat nhỏ, thì n không phải là số nguyên tố. Ngược lại, nếu a thỏa mãn định lý Fermat nhỏ, thì thuật toán sẽ tiếp tục kiểm tra với một số a khác. Thực hiện quá trình này tối đa k lần, với k là một số nguyên dương được chọn trước đó. Nếu trong số k lần kiểm tra, tất cả đều cho kết quả là a không phải là số nguyên tố, thì thuật toán sẽ kết luận n không phải là số nguyên tố. Trong trường hợp khác, thuật toán sẽ kết luận rằng n có xác suất rất cao là số nguyên tố.

Thuật toán Miller-Rabin được coi là một trong những thuật toán kiểm tra số nguyên tố nhanh nhất hiện nay và được sử dụng rộng rãi trong các ứng dụng liên quan đến lý thuyết số, mật mã học và khoá công khai.

Giải thuật miller-rabin

Input: số nguyên n cần kiểm tra tính nguyên tố, số lần kiểm tra k

1. Nếu n <= 1 hoặc n == 4, trả về False.
2. Nếu n <= 3, trả về True.
3. Tính các giá trị r và s sao cho n-1 = 2^r * s, với s là số lẻ.
4. Duyệt qua k lần kiểm tra sau:

    4.1. Chọn số nguyên tố ngẫu nhiên a trong khoảng [2, n-2].

4.2. Tính x = a^s mod n.

4.3. Nếu x == 1 hoặc x == n-1:

Chuyển sang kiểm tra lần kiểm tra kế tiếp.

4.4. Với mỗi j từ 1 đến r-1, tính x = x^2 mod n.

Nếu x == n-1, ta chuyển sang kiểm tra lần kiểm tra kế tiếp.

4.5. Nếu không tìm được j nào sao cho x == n-1, trả về False.

5. Trả về True.

2.3.3 Thuật toán Euclid

Thuật toán Euclid là một thuật toán để xác định ước chung lớn nhất (GCD – Greatest Common Divisor) của 2 phần tử thuộc vùng Euclid (ví dụ: các số nguyên). Thuật toán Euclid là một trong những thuật toán cổ nhất được biết đến, từ khi xuất hiện trong cuốn Euclid's Elements khoảng năm 300 trước công nguyên. Euclid khởi đầu đã trình bày rõ ràng vấn đề về phương diện hình học, như vấn đề tìm ra một thước đo chung có độ dài hai đường thẳng, và thuật toán của ông đã xử lý bằng cách lặp lại phép trừ đoạn dài hơn cho đoạn ngắn hơn. Tuy nhiên, thuật toán đã hầu như không được phát hiện ra bởi Euclid và nó đã có thể được biết đến sớm hơn 200 năm. Nó cũng đã được biết đến bởi Eudoxus of Cnidus (khoảng 375 trước công nguyên) và Aristotle (khoảng 330 trước công nguyên).

Thuật toán Euclid là một phương pháp tìm ước chung lớn nhất (UCLN) của hai số nguyên dương. Thuật toán này được đặt tên theo nhà toán học Euclid, người đã phát hiện ra nó vào khoảng thế kỷ thứ 3 trước Công nguyên.

Ý tưởng chính của thuật toán Euclid là sử dụng tính chất UCLN của hai số để giảm độ lớn của chúng dần cho đến khi hai số bằng nhau, khi đó ước chung lớn nhất của chúng chính là số đó.

Bài toán: Cho 2 số nguyên dương a và b. Tìm ước số chung lớn nhất của a và b. Ký hiệu GCD (a, b) =?

- Thuật toán Euclid để tìm UCLN của 2 số a và b:

- Bước 1: Nếu a = 0 thì GCD (a, b) = GCD (0, b) = b
- Bước 2: Nếu b = 0 thì GCD (a, b) = GCD (a, 0) = a
- Bước 3: Nếu a # 0, b # 0. Giả sử a >b

- Viết a dưới dạng a = q*b + r
- Theo thuật toán Euclid, ta có GCD (a, b) = GCD (b, r) = GCD (b, a mod b)
- Cài đặt thuật toán:

```
If (a == 0)    return b
If (b == 0)    return a
        While(b>0)
                r = a mod b
                a = b, b = r
return a
```

Nhận xét: Như vậy, để tìm UCLN của 1 cặp số cho trước ta đưa về bài toán tìm UCLN của cặp số gồm số nhỏ hơn trong 2 số đó và phần dư của số lớn hơn khi chia cho số nhỏ hơn. Thuật toán Euclid tạo nên vòng lặp, ở mỗi bước ta áp dụng tính chất trên khi phần dư đó còn khác 0.

Chứng minh thuật toán

Cho r_1, r_0. Tìm UCLN(GCD)

$r_0 = q_1*r_1 + r_2$ GCD (r_0, r_1) = GCD (r_1, r_2)

$r_1 = q_2*r_2 + 0$

…

$r_{m-2} = q_{m-1}*r_{m-1} + r_m$

$r_{m-1}= q_m*r_m+ 0$

Giả sử a là UCLN của r_0, r_1. Theo định nghĩa, a là số lớn nhất r_0/a và r_2/a

$(q_1*r_1 + r_2)/a$ và r_1/a

⇨ r_1/a, r_2/a

Theo định nghĩa a = GCD (r_1, r_2)

Ví dụ minh họa:

Áp dụng thuật toán Euclid tìm GCD (814, 187)

Ta có: 814 = 187*4 + 66

187 = 66*2 + 55

66 = 55*1 + 11

55 = 11*5 + 0

Vậy GCD (814, 187) = 11

2.3.4 Thuật toán Euclid mở rộng

Thuật toán Euclid sử dụng để giải một phương trình vô định nguyên (còn được gọi là phương trình Đi-ô-phăng) có dạng: a ×

x + b × y = c, trong đó a, b, c là các hệ số nguyên, x, y là các ẩn nhận giá trị nguyên. Điều kiện cần và đủ để phương trình này có nghiệm (nguyên) là gcd(a, b) là ước của c. Khẳng định này dựa trên một mệnh đề sau:

Trong số học đã biết rằng nếu d = gcd(a, b) thì tồn tại các số nguyên x, y sao cho a × x + b × y = d

Giải thuật Euclid mở rộng kết hợp quá trình tìm gcd(a, b) trong thuật toán Euclid với việc tìm một cặp số x, y thoản mãn phương trình Đi-ô-phăng. Giả sử cho hai số tự nhiên a, b, ngoài ra a >b>0. Đặt r_0 = a, r_1 = b, chia r_0 cho r_1 được số dư r_2 và thương số nguyên q_1. Nếu r_2 = 0 thì dừng lại, nếu r_2 khác không, chia r_1 cho r_2 được số dư r_3 .... Vì dãy các ri là giảm thực sự nến sau hữu hạn bước ta được số dư r_m+2 = 0.

$r_0 = q_1 \times r_1 + r_2, 0 < r_2 < r_1$; $r_1 = q_2 \times r_2 + r_3, 0 < r_3 < r_2$;

....;

$r_m - 1 = q_m \times r_m + r_m + 10 < r_m + 1 < r_m$, $r_m = q_m + 1 \times r_m + 1$

trong đó số dư cuối cùng khác 0 là $r_m + 1 = d$. Bài toán đặt ra là tìm x, y sao cho

$a \times x + b \times y = r_m + 1 (= d)$

Khi i = m − 1 ta có được $x_m + 1$ và $y_m + 1$. Các công thức (1), (2),

(3) là công thức truy hồi để tính x, y.

Áp dụng giải thuật Euclid mở rộng tìm số nghịch đảo trong vành Z_m Số nghịch đảo trong vành Z_m

Trong lý thuyết số, vành Z_m được định nghĩa là vành thương của với quan hệ đồng dư theo môđun m (là quan hệ tương đương) mà các phần tử của nó là các lớp đồng dư theo mođun m

(m là số nguyên dương lớn hơn 1). Ta cũng có thể xét Z_m chỉ với các đại diện của nó. Khi đó

$Z_m = \{0, 1, ..., m - 1\}$

thông thường được rút gọn theo mođun m:

$$a + b = (a + b) \bmod m \qquad (2.4)$$

$$a \times b = (a \times b) \bmod m \qquad (2.5)$$

Phần tử a của Z_m được gọi là khả nghịch trong Z_m hay khả nghịch theo mođun m nếu tồn tại phần tử a' trong Z_m sao cho $a \times a' = 1$ trong Z_m hay $a \times a' \equiv 1 (\bmod m)$. Khi đó a' được gọi là nghịch đảo modulo m của a. Trong lý thuyết số đã chứng minh rằng, số a là khả nghịch theo mođun m khi và chỉ khi GCD của a và m bằng 1.

Khi đó tồn tại các số nguyên x, y sao cho

$$m \times x + a \times y = 1$$

Đẳng thức này lại chỉ ra x là nghịch đảo của a theo mođun m. Do đó có thể tìm được phần tử nghịch đảo của a theo mođun m nhờ thuật toán Euclid mở rộng khi chia m cho a.

Thuật toán:

Đầu vào: hai số nguyên a và b.

Bước 1: Khởi tạo các giá trị đầu tiên.

$a0 = 1, b0 = 0, a1 = 0, b1 = 1$

Bước 2: Thực hiện vòng lặp để tìm giá trị của d và các hệ số x, y.

while $b \neq 0$ do

q = a div b r = a mod b

a = b b = r

x = a0 - q * a1 y = b0 - q * b1

a0 = a1 a1 = x

b0 = b1 b1 = y

Bước 3: Trả về kết quả.

d = a

x = a0

y = b0

Trong đó, a div b là phép chia lấy phần nguyên của a cho b.

Khi thuật toán kết thúc, giá trị của d là ước số chung lớn nhất của a và b, và giá trị của x và y là các hệ số sao cho ax + by = d.

Các bước tính:

Bước 1: Xây dựng bảng

Dòng	r_0	r_1	r_2	q	t_0	t_1

Trên 1 dòng: $r_0 = r_1*q+t_2$

Bước 2: Điền vào dòng đầu tiên $r_0 = n$, $r_1 = a$, $t_0 = 0$, $t_1 = 1$

Dòng	r_0	r_1	r_2	q	t_0	t_1
0	n	a			0	1

Bước 3: trên dòng i đang xét tính giá trị r_2, q

Dòng	r_0	r_1	r_2	q	t_0	t_1
0	n	a			0	1
…						
i			r_0 mod r_1	r_0/r_1		

Bước 4: Tính giá trị t_1 của dòng tiếp theo

Dòng	r_0	r_1	r_2	q	t_0	t_1

…						
i-1				X	Y	Z
i		r_0 mod r_1		r_0/r_1		(Y-X*Z) mod n

Bước 5: Trên dòng i đang xét

+ Nếu $r_2 = 0$ thì

Nếu $r_1 = 1$ thì giá trị t_1 (của dòng đang xét) là phần tử nghịch đảo của a trong Z_n

Ngược lại ($r_1 \# 1$) thì không tồn tại phần tử nghịch đảo của a trong Z_n chỉ xảy ra khi UCLN (a, n) #1

+ $r_2 \# 0$ => sang bước 6

Bước 6: Sao chép giá trị sang dòng tiếp theo: $r_0 = r_1$, $r_1 = r_2$, $t_0 = t_1$, sau đó trở lại bước 3 theo quy tắc

Ví dụ minh họa: n = 121, a = 39

Dòng	r_0	r_1	r_2	q	t_0	t_1
0	121	39	4	3	0	1
1	39	4	3	9	1	118
2	4	3	1	1	118	28
3	3	1	0	3	28	90

Vậy $39^{-1} = 90$ (trong Z_{121})

2.3.5 Định lý Fermat

Định lý Fermat là một trong những định lý toán học quan trọng nhất của thế kỷ XVII, được đặt tên theo nhà toán học Pierre de Fermat. Định lý nói rằng:

"Nếu n là một số nguyên dương và a là một số nguyên không chia hết cho n, thì a^(n-1) ≡ 1 (mod n)."

Trong đó, a^(n-1) được đọc là "a mũ n trừ một", và ≡ được đọc là "đồng dư với". Cụ thể, định lý Fermat khẳng định rằng nếu chia a^(n-1) cho n thì số dư sẽ luôn bằng 1.

Quy tắc này có thể được diễn giải như sau: Nếu ta lấy một số tự nhiên a không chia hết cho một số nguyên dương n bất kỳ và tính a^(n-1), thì khi ta chia kết quả cho n, số dư sẽ luôn là 1.

Ví dụ: nếu ta lấy a = 2 và n = 5, theo định lý Fermat, ta có:

$$2^{(5-1)} \equiv 1 \pmod 5$$

Tức là khi ta tính 2^(5-1) = 16, và chia kết quả cho 5, số dư sẽ là 1.

Định lý Fermat là một trong những định lý toán học quan trọng nhất và có ứng dụng rộng rãi trong các lĩnh vực khác nhau của toán học, như lý thuyết số và mật mã học: thuật toán kiểm tra số nguyên tố Miller-Rabin hay thuật toán RSA trong mật mã học.

2.3.6 Định lý Euler

Định lý Euler là một trong những định lý cơ bản trong toán học, được đặt theo tên của nhà toán học người Thụy Sĩ Leonhard Euler. Định lý này liên quan đến các hàm số mũ và được phát biểu như sau:

"Cho a và m là hai số nguyên tố cùng nhau (tức là ước chung lớn nhất của a và m là 1). Khi đó, a^φ(m) ≡ 1 (mod m), trong đó φ(m) là hàm số Euler, đếm số lượng số nguyên tố nhỏ hơn hoặc bằng m."

Trong đó, a^φ(m) ≡ 1 (mod m) có nghĩa là a^φ(m) chia hết cho m và cho phần dư là 1 khi chia cho m.

Ví dụ: nếu n = 8, thì φ(n) = 4, vì có bốn số nguyên tố nhỏ hơn 8 và cùng nhau với 8 là 1, 3, 5 và 7.

Định lý Euler là một trong những định lý quan trọng trong lý thuyết số và được sử dụng rộng rãi trong các ứng dụng mã hóa thông tin và bảo mật. Nó được sử dụng để mã hóa thông tin và giải mã thông tin trong các hệ thống mật mã học, ví dụ như trong thuật toán khóa công khai RSA.

2.3.7 Thuật toán Bình phương và nhân

Thuật toán bình phương và nhân là thuật toán tính nhanh lũy thừa tự nhiên của một số (thực hoặc nguyên), trong trường hợp cơ số là số nguyên có thể được rút gọn theo một môđun nào đó.

Phép nâng lên lũy thừa tự nhiên bậc n của số x (x được gọi là cơ số) được định nghĩa từ hệ thức

$$x^n = \underbrace{x * x * ... * x}_{n}$$

Chẳng hạn với n=35 quá trình tính x^{35} qua 35 bước: $1 \Rightarrow x \Rightarrow x^2 = x * x \Rightarrow x^3 = x^2 * x \Rightarrow ... \Rightarrow x^{35} = x^{34} * x$

Ta nhận xét rằng có thể giảm bớt số phép nhân chẳng hạn với dãy phép tính

$1 \Rightarrow x \Rightarrow x^2 = x * x, \Rightarrow x^4 = (x^2)^2, \Rightarrow x^8 = (x^4)^2$

$\Rightarrow x^{16} = (x^8)^2 \Rightarrow x^{17} = x^{16} * x, \Rightarrow x^{34} = (x^{17})^2 \Rightarrow x^{35} = x^{34} * x.$

Quá trình tính toán trên chính là quá trình tính nhờ công thức đệ quy

1. Với n=0 thì $x^n = 1$
2. Với n>0 ta có công thức

$$f(n) = \begin{cases} (x^k)^2, & \text{khi } n = 2k \\ (x^k)^2 * x, & \text{khi } n = 2k+1 \end{cases}$$

Như vậy phép tính x^n được quy về một số phép bình phương và phép nhân do vậy mà có tên gọi **thuật toán bình phương và nhân**.

Ví dụ [sửa | sửa mã nguồn]

Trong ví dụ sau ta tính $37^{27} \pmod{101}$.

Đổi $n = 27$ ra số nhị phân ta được $27 = 11011_{(2)}$.

Bảng sau đây tính toán từng bước theo giá trị của các bít của 27.

Khởi tạo $p = 1$.

$b[i]$	$p = p^2$	$p = p \pmod{101}$	$p * x$	$p = \pmod{101}$
1	$1^2 = 1$	1	$1 * 37 = 37$	37
1	$37^2 = 1369$	56	$56 * 37 = 2072$	52
0	$52^2 = 2704$	78	-	78
1	$78^2 = 6084$	24	$24 * 37 = 888$	80
1	$80^2 = 6400$	37	$37 * 37 = 1369$	56

Như vậy ta có

$$37^{27} \pmod{101} = 56$$

2.3.8 Thuật toán RSA

2.3.8.1 Tạo khóa

Để cài đặt RSA, ban đầu mỗi người dùng sinh khóa công khai và khóa bí mật của mình bằng cách:

1) Chọn ngẫu nhiên 2 số nguyên tố lớn p và q, sao cho:

   $p \neq q$, lựa chọn ngẫu nhiên và độc lập

2) Tính số làm modulo của hệ thống: $n = p * q$

   Giá trị hàm số Euler: $\phi(n) = (p-1) * (q-1)$

3) Chọn ngẫu nhiên khoá mã hóa e, sao cho:

   $1 < e < \phi(n)$ và $GCD(e, \phi(n)) = 1$

4) Giải phương trình sau để tìm khoá giải mã d, sao cho:

   $d = e^{-1} \bmod \phi(n)$

5) Khoá mã công khai: $K_{pub} = \{e, n\}$

6) Giữ khoá riêng bí mật: $K_{pr} = \{d, p, q\}$

**Một số lưu ý:*

- Các số nguyên tố thường được chọn bằng phương pháp thử xác xuất
- Bước 3 và 4 có thể thực hiện bằng giải thuật Eculid mở rộng

Ở đây, p và q giữ vai trò rất quan trọng. Chúng là các phân tố của n và cho phép tính d khi biết e. Nếu không sử dụng dạng sau của khóa bí mật (dạng CRT) thì p và q sẽ được xóa ngay sau khi thực hiện xong quá trình tạo khóa.

Việc thiết lập khóa này được thực hiện 1 lần khi một người dùng thiết lập (thay thế) khóa công khai của họ. Mũ e thường là khá nhỏ (để mã hóa nhanh), và phải là nguyên tố cùng nhau với $\phi(n)$. Các giá trị thường được chọn cho e là 3 hoặc $2^{16} - 1 = 65535$. Tuy nhiên, khi e nhỏ thì d sẽ tương đối lớn. Khoá bí mật là (d, p, q). Các số p và q thường có giá trị xấp xỉ nhau nhưng không được bằng nhau. Chú ý là việc để lộ một trong các thành phần trên sẽ làm cho hệ mã hóa trở thành không an toàn.

2.3.8.2 Mã hóa (Sử dụng khóa công khai)

Để mã hóa thông điệp m:

$c = m^e \pmod n$ với $0 <= m < n$

2.3.8.3 Giải mã (Sử dụng khóa bí mật)

Để giải mã bảng mã c:

$m = c^d \pmod n$

2.3.8.4 Ví dụ minh họa các hoạt động

Sau đây là một ví dụ với những số cụ thể. Ở đây chúng ta sử dụng những số nhỏ để tiện tính toán còn trong thực tế phải dùng các số có giá trị đủ lớn.

Lấy:

p = 61 - số nguyên tố thứ nhất (giữ bí mật/hủy sau khi tạo khóa)

q = 53 - số nguyên tố thứ hai (giữ bí mật/hủy sau khi tạo khóa)

n = pq = 3233 - môđun (công bố công khai)

e = 17 - số mũ công khai

d = 2753 - số mũ bí mật

Khóa công khai là cặp (e, n). Khóa bí mật là d. Hàm mã hóa là:

$\text{encrypt}(m) = m^e \bmod n = m^{17} \bmod 3233$

với m là văn bản rõ. Hàm giải mã là:

$\text{decrypt}(c) = c^d \bmod n = c^{2753} \bmod 3233$

với c là văn bản mã.

Để mã hóa văn bản có giá trị 123, ta thực hiện phép tính:

$\text{encrypt}(123) = 123^{17} \bmod 3233 = 992$

Để giải mã văn bản có giá trị 992, ta thực hiện phép tính:

$\text{decrypt}(992) = 992^{2753} \bmod 3233 = 123$

Cả hai phép tính trên đều có thể được thực hiện hiệu quả nhờ thuật toán bình phương và nhân.

2.4 Thiết kế chương trình, cài đặt thuật toán C#

2.4.1 Giới thiệu ngôn ngữ C#

C# (hay còn gọi là C Sharp), là ngôn ngữ lập trình hướng đối tượng, hiện đại do Microsoft phát triển như một phần của nền tảng .NET. Nó được giới thiệu lần đầu tiên vào năm 2000 và kể từ đó đã trở thành một trong những ngôn ngữ lập trình được sử dụng rộng rãi nhất trong ngành.

C# có thiết kế đơn giản, mạnh mẽ và linh hoạt. Sở hữu một cú pháp tương tự như các ngôn ngữ lập trình phổ biến, bao gồm Java và C++, giúp các nhà phát triển đã quen thuộc với các ngôn ngữ này dễ làm quen. C# được sử dụng rộng rãi trong việc phát triển các ứng dụng Windows dành cho máy tính để bàn, game, ứng dụng web và các ứng dụng dành cho thiết bị di động. Ngoài ra, nó còn được sử dụng trong phát triển phần mềm doanh nghiệp, với nhiều công ty lớn sử dụng C# để xây dựng các ứng dụng kinh doanh của họ. Cho dù mới bắt đầu làm quen với lập trình hay là một nhà phát triển có kinh nghiệm, thì C# luôn là một ngôn ngữ tuyệt vời để học và sử dụng khi muốn xây dựng các ứng dụng mạnh mẽ, đáng tin cậy và có thể mở rộng.

Lý do lựa chọn ngôn ngữ C#:

C # là ngôn ngữ đơn giản: C# đã được loại bỏ sự phức tạp như các ngôn ngữ C++ và Java bao gồm loại bỏ macro, template, lớp cơ sở (virtual base class). Nếu bạn đã sử dụng qua các ngôn ngữ C, C++ hay cả Java thì C# cũng nó cũng tương tự về cú pháp, biểu thức, nhưng nó được cải tiến để làm ngôn ngữ đơn giản hơn.

Ngôn ngữ hướng đối tượng: Lập trình hướng đối tượng (OOP – Object Oriented Programming) là phương pháp lập trình với 4 tính chất:

- Tính trừu tượng (abstraction).
- Tính đóng gói (encapsulation).
- Tính đa hình (polymorphism).
- Tính kế thừa (inheritance).

Thời gian develop nhanh hơn: Ưu thế lớn nhất chính là khả năng tiết kiệm thời gian develop, vì:

- Được nhập tĩnh, dễ đọc, giảm thiểu thời gian debug.
- Sở hữu thư viện khổng lồ, cung cấp các chức năng cấp cao so với các ngôn ngữ khác như Java hay C++.
- Mang lại sự đơn giản và hiệu quả. Đồng thời còn hỗ trợ các lập trình viên viết các đoạn code phức tạp.
- Có ngân hàng bộ nhớ mở rộng, giảm thời gian develop.

Đường cong học tập thấp: Các developer, đặc biệt là full-stack developer, có xu hướng thích ngôn ngữ C# hơn. Ngoài việc tiết kiệm thời gian develop, nó còn có đường cong học tập (learning curve) thấp. Các developer có thể dành ít thời gian hơn cho việc học C# so với các ngôn ngữ khác. Tính đơn giản và dễ sử dụng rất thuận lợi cho cả những developer mới.

Khả năng mở rộng cao: Phát triển phần mềm yêu cầu ngôn ngữ lập trình dễ dàng bảo trì và có thể mở rộng và C# là một ngôn ngữ như vậy. Tính nghiêm ngặt của mã hóa tĩnh làm cho các chương trình nhất quán với nhau. Điều này cho phép các developer dễ dàng thực hiện các điều chỉnh cũng như bảo trì.

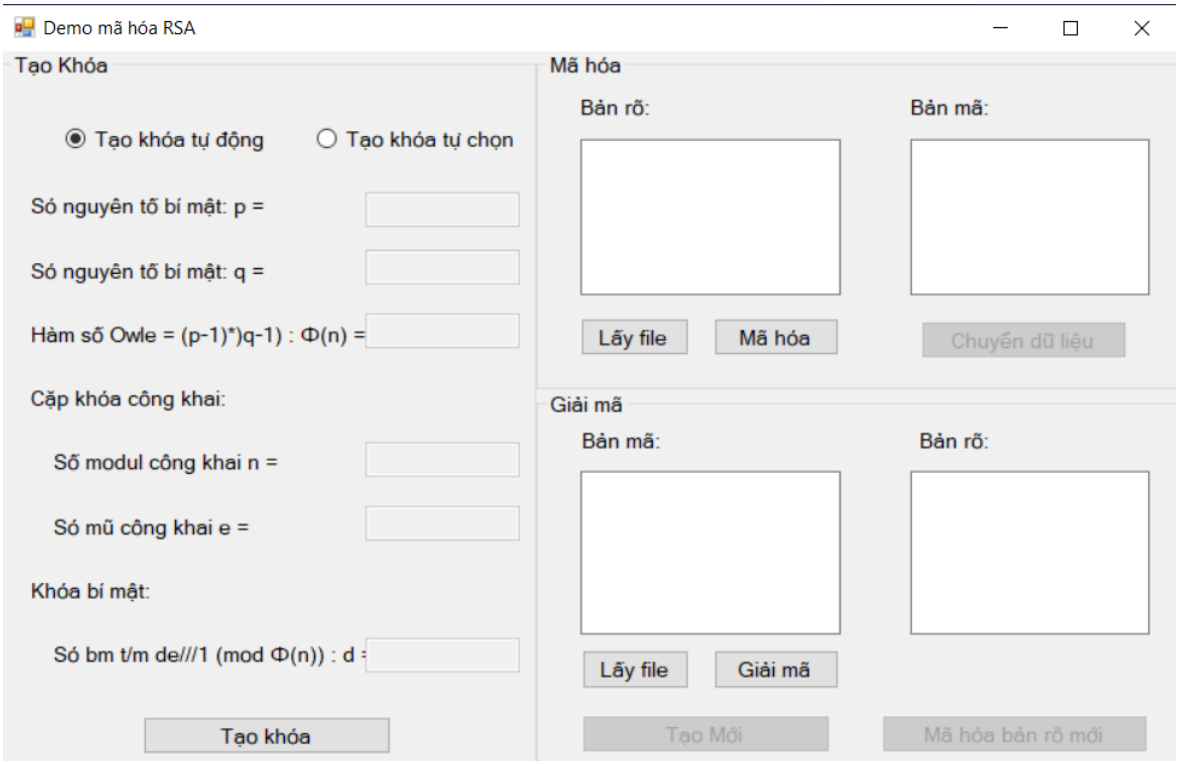
C# có thể được nâng cấp và bảo trì liên tục để mở rộng

2.4.2 Thiết kế kịch bản chương trình

1. Yêu cầu người dùng nhập vào thông tin cần mã hóa, ví dụ như một chuỗi ký tự hoặc một số nguyên. Sử dụng thuộc tính Text để nhận đầu vào từ người dùng.
2. Tạo khóa tự động sẽ tự sinh các số nguyên tố p, q sử dụng hàm RSA_ChonSoNgauNhien(), tạo khóa tự chọn sẽ nhập vào hai giá trị p, q. Sử dụng các hàm RSA_kiemTraNguyenTo(), RSA_nguyenToCungNhau() để kiểm tra điều kiện p, q và hàm RSA_taoKhoa() để tính các giá trị còn lại.
3. Sau khi hoàn thành bước sinh khóa và nhập bản rõ, người dùng nhấn nút "Mã hóa" sử dụng hàm bt_mahoa_click() để nhận lệnh, RSA_MaHoa() để mã hóa.
4. Sử dụng nút "Chuyển dữ liệu" để chuyển bản mã, sử dụng hàm bt_Chuyen_Click().
5. Sau khi nhập bản mã, người dùng nhấn nút "Giải mã" sử dụng hàm bt_giaima_click() để nhận lệnh, RSA_GiaiMa() để mã hóa.
6. Sử dụng try-catch để bắt lỗi và các MessageBox để hiển thị các thông báo khi cần thiết.

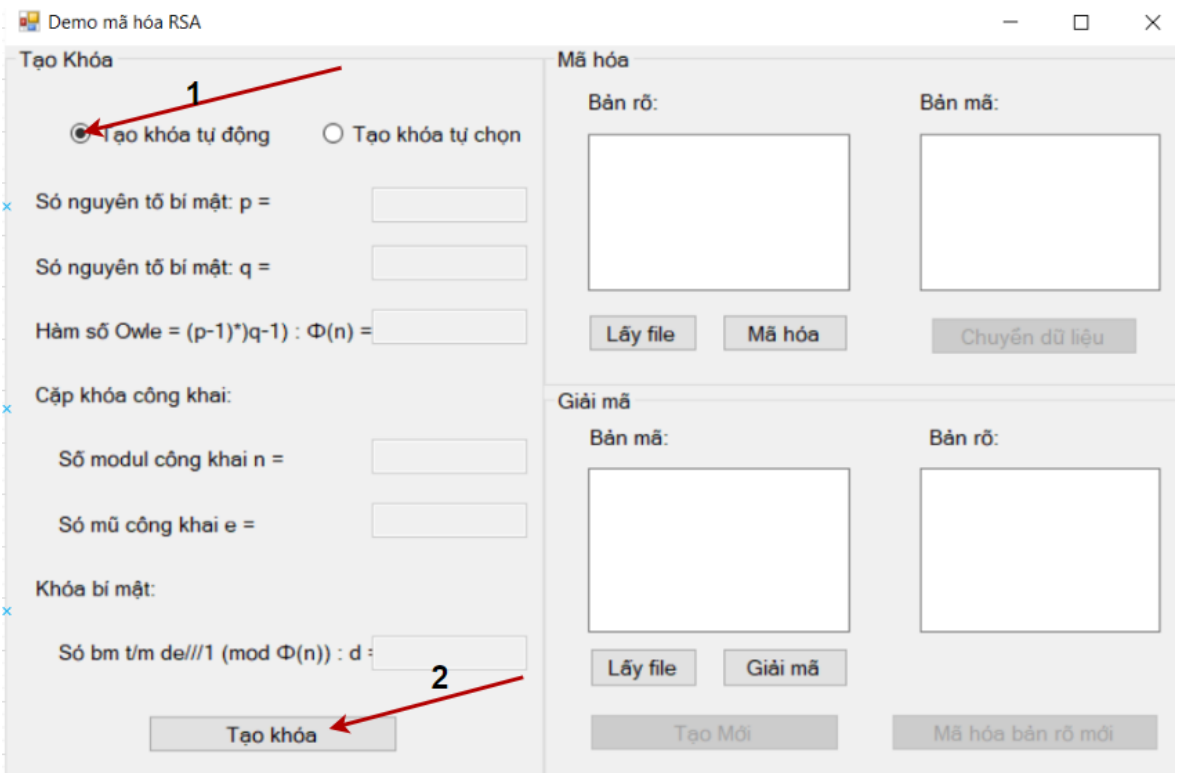
2.4.3 Cài đặt thuật toán, hình ảnh minh hoạ

Giao diện chương trình

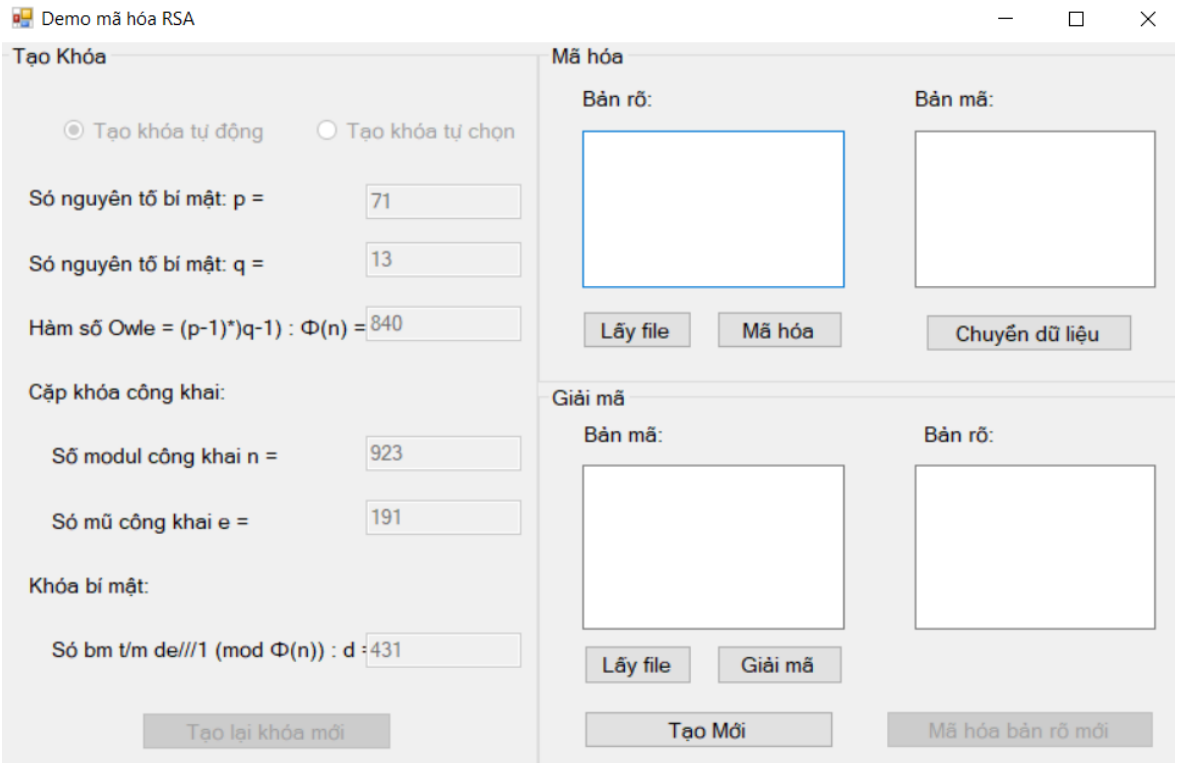


Trường hợp sử dụng tạo khóa tự động:

Bước 1: Người dùng nhấn vào nút "Tạo khóa tự động", sau đó nhấn vào nút "tạo khóa", các số p, q, hàm Ơle, n, e,d sẽ tự động sinh.

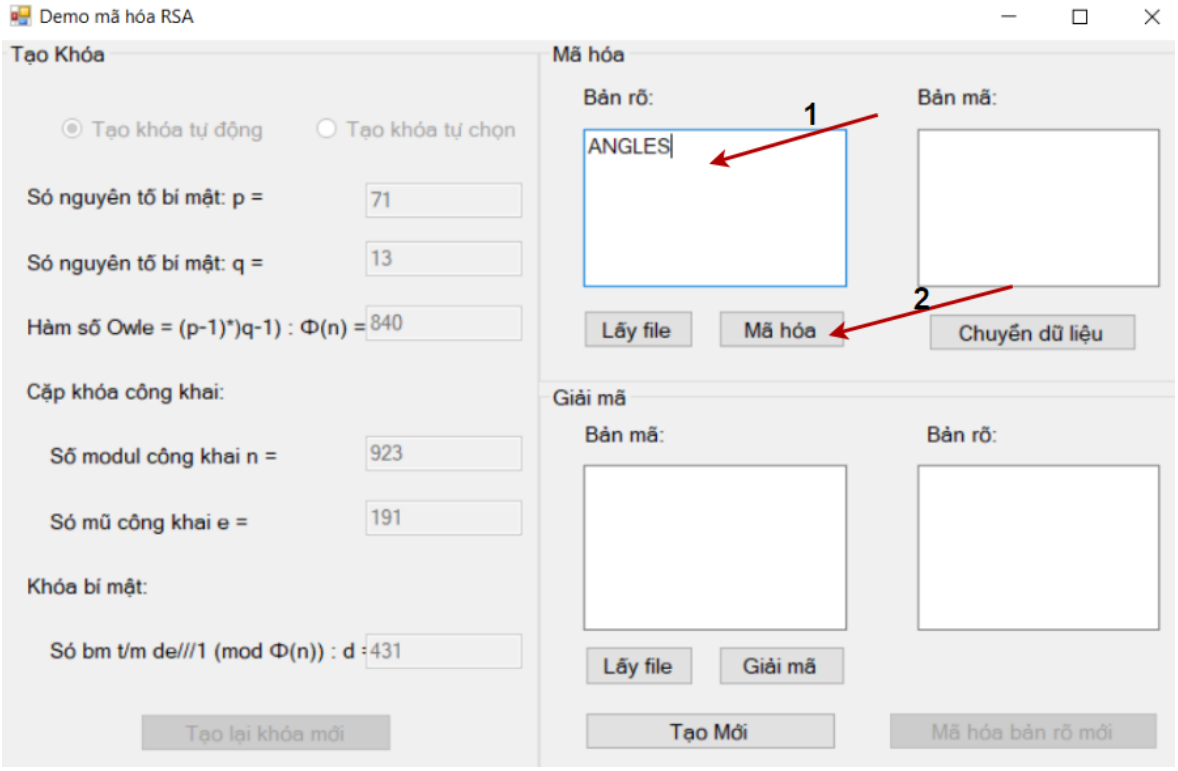


Tạo khóa tự động thành công:

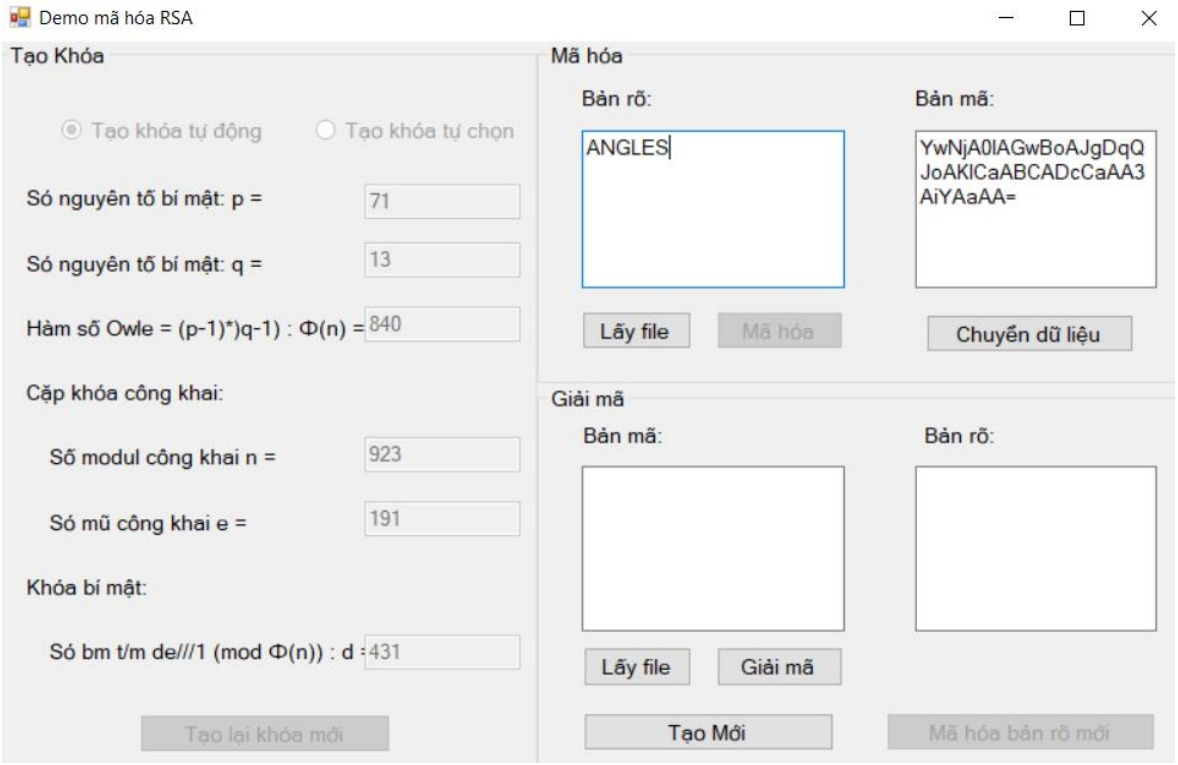


Bước 2: Nhập bản rõ hoặc nhấn nút “Lấy file” để lấy file cần mã hóa sau đó nhấn nút “Mã hóa”, hệ thống sẽ mã hóa bản rõ thành bản mã và hiện lên ở

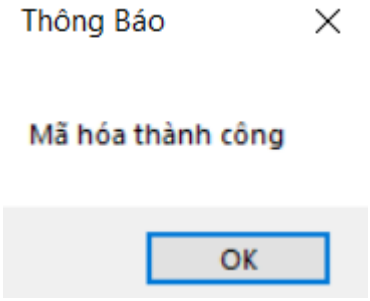
ô bản mã, đồng thời ghi vào file txt. Lúc này, nút “Tạo khóa” sẽ được đổi tên thành “Tạo lại khóa mới”.



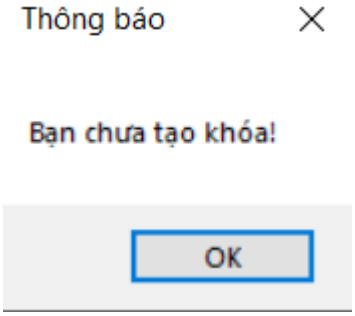
Mã hóa thành công:



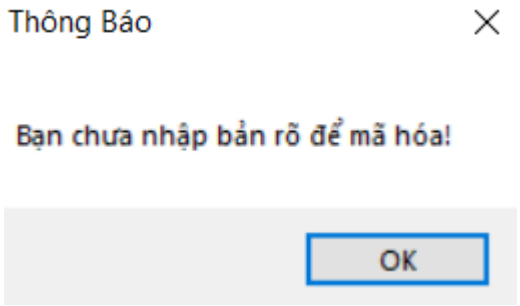
Thông báo hiện ra:



Nếu chưa tạo khóa và nhấn nút "Mã hóa"/"Giải mã", hệ thống sẽ hiển thị ra thông báo "Bạn chưa tạo khóa".

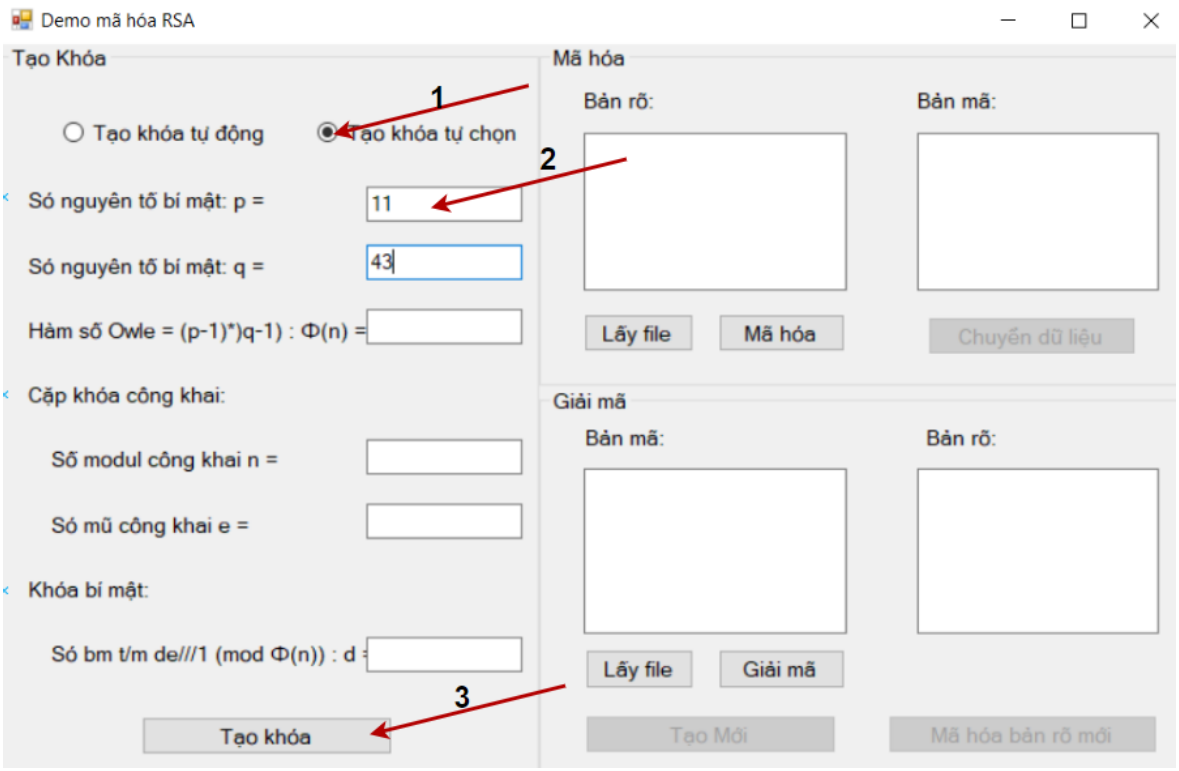


Nếu bản rõ để trống và nhấn nút "Mã hóa, hệ thống sẽ hiển thị ra thông báo "Bạn chưa nhập bản rõ để mã hóa".

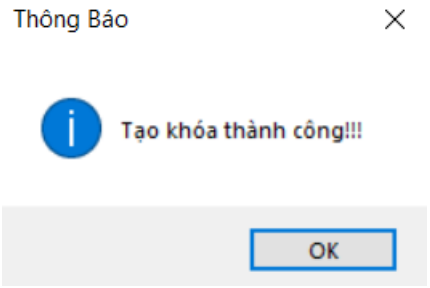


Trường hợp sử dụng tạo khóa tự chọn:

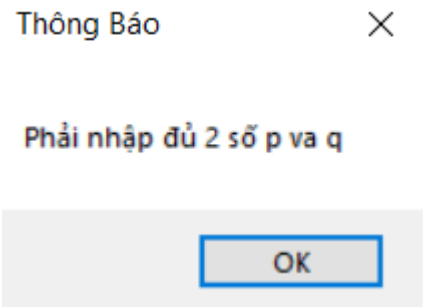
Bước 1: Người dùng nhấn vào nút "Tạo khóa tự chọn", sau đó nhập hai giá trị của p và q >=11. Nhấn chuột vào nút "Tạo khóa", các giá trị còn lại hệ thống sẽ tự tính theo công thức lập trình sẵn.



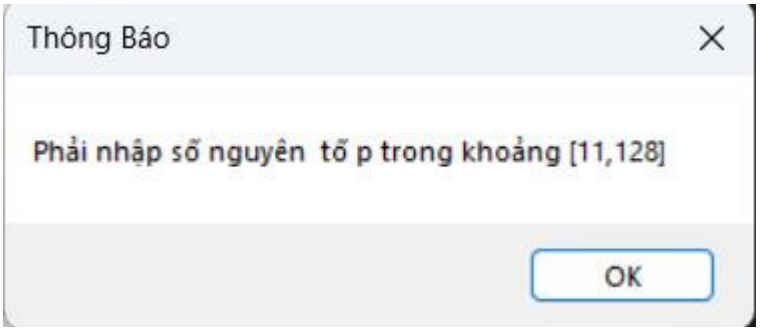
Sau khi tạo khóa thành công, hệ thống sẽ hiển thị ra thông báo"Tạo khóa thành công".



Nếu chưa nhập đủ hai số p, q, hệ thống sẽ hiển thị thông báo "Phải nhập đủ 2 số p và q".



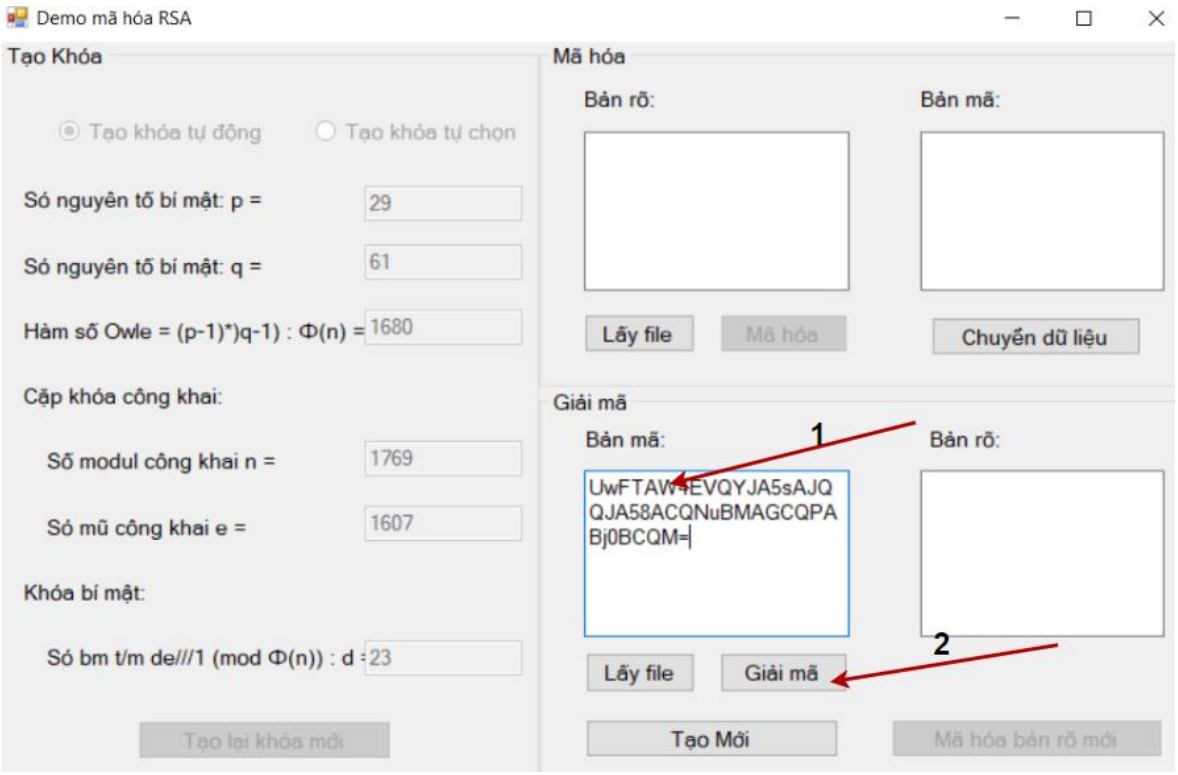
Nếu hai số p, q nhập vào không thoa nãm điều kiến >=11 và <= 128 thì sẽ hiện ra thông báo "Hai số p và q phải nằm trong khoảng [11, 128]:



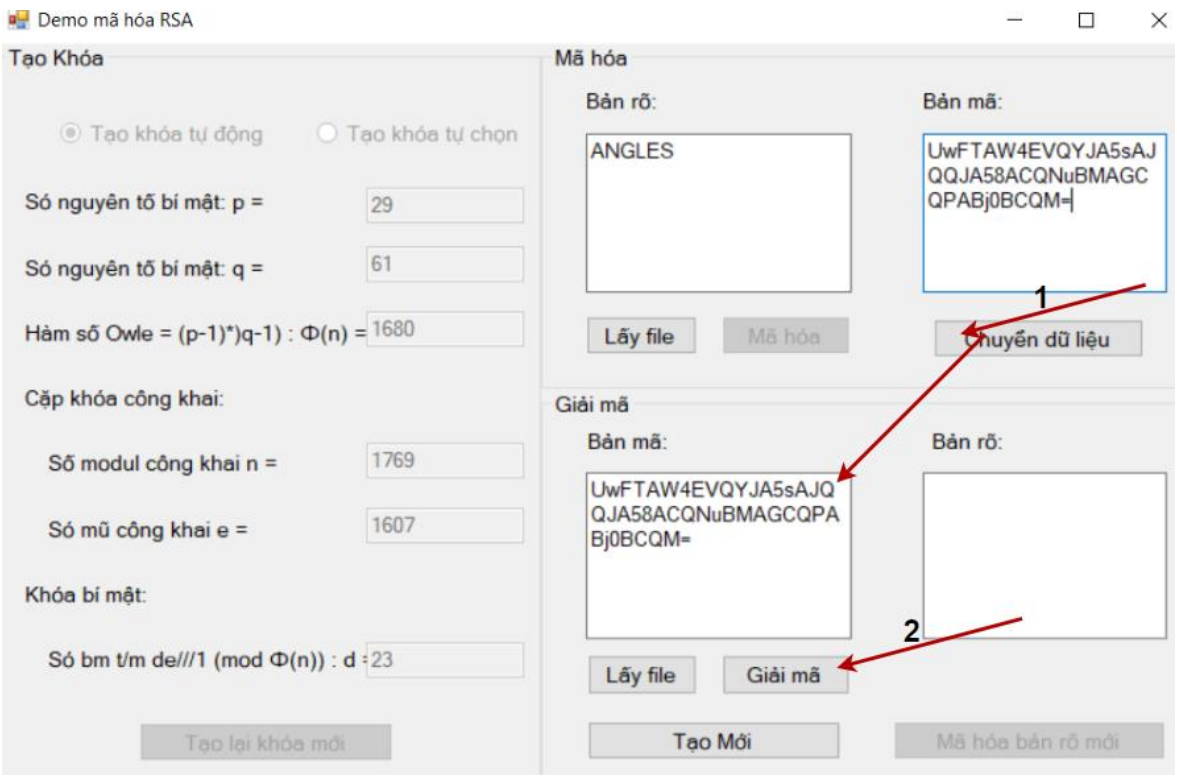
Bước 2 + Bước 3: Tương tự như trường hợp tạo khóa tự động.

Giải mã:

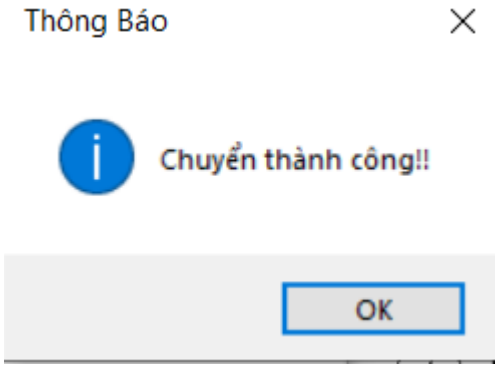
Bước 1: Nhập vào ô "Bản mã" bản mã muốn giải mã hoặc nhấn nút "Lấy file" để chọn file txt cần giải mã, sau đó nhấn vào nút "Giải mã".



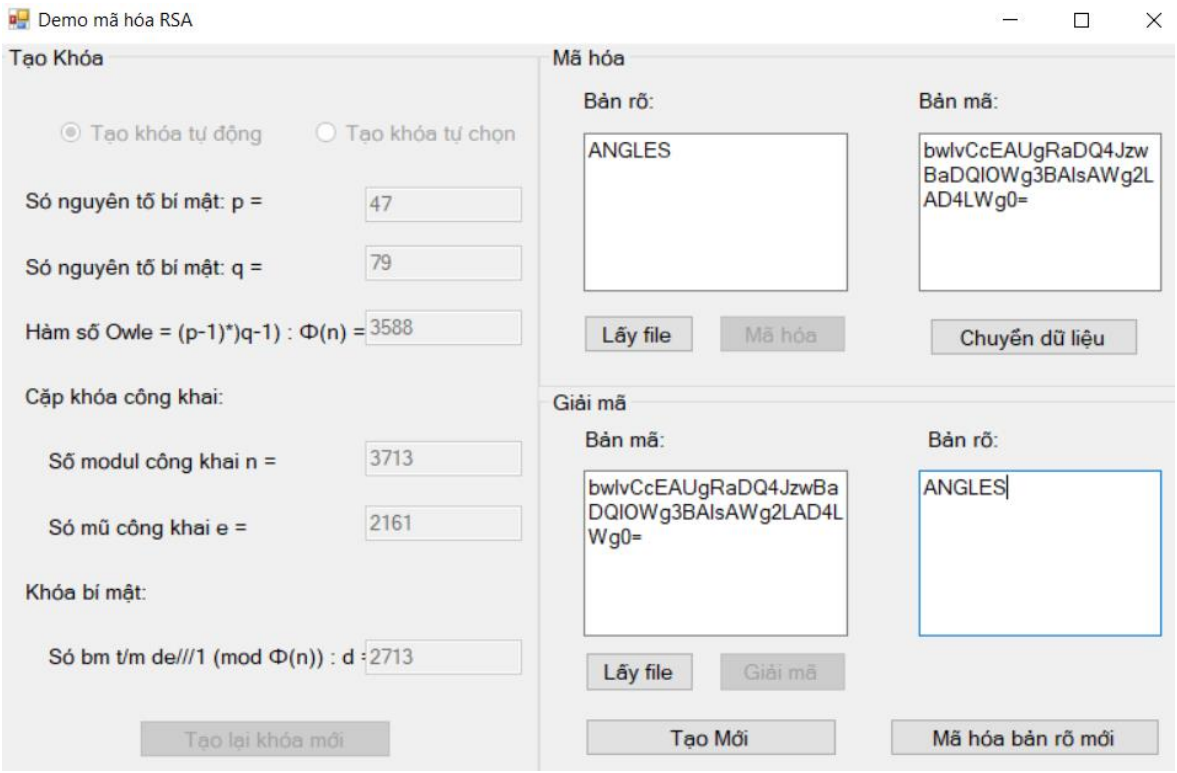
Hoặc sau khi mã hóa, người dùng có thể ấn nút "Chuyển dữ liệu" để chuyển bản mã vừa mã hóa sang ô bản mã phần giải mã.



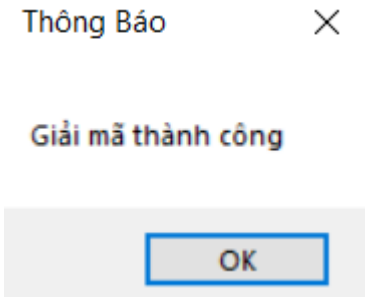
Chuyển dữ liệu thành công sẽ có thông báo hiện ra "Chuyển dữ liệu thành công":



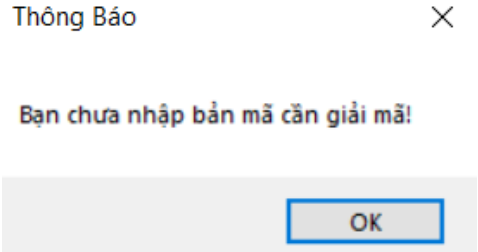
Giải mã thành công:



Thông báo khi giải mã thành công:



Nếu trong quá trình nhập bản mã, người dùng để trống và nhấn nút giải mã, hệ thống sẽ hiển thị ra thông báo “Bạn chưa nhập bản mã cần giải mã”



Nếu bản mã phần giải mã bị thay đổi và nhấn nút “Giải mã", hệ thống sẽ hiển thị ra thông báo “Không thể mã hóa”:

Reset lại chương trình:

Để reset lại chương trình trở về như ban dầu nhằm xóa hết những giá trị khóa và bản mã vừa sử dụng, nhấn nút "Tạo mới". Chương trình sẽ trở về giao diện ban đầu. Hoặc nhấn nút "Mã hóa bản rõ mới" để sử dụng lại khóa đã tạo và mã hóa bản rõ khác.

❖ **Tạo khóa**

```
2 references
private void RSA_taoKhoa()
{
    //Tinh n=p*q
    RSA_soN = RSA_soP * RSA_soQ;
    rsa_soN.Text = RSA_soN.ToString();
    //Tính Phi(n)=(p-1)*(q-1)
    RSA_soPhi_n = (RSA_soP - 1) * (RSA_soQ - 1);
    rsa_soPhiN.Text = RSA_soPhi_n.ToString();
    //Tính e là một số ngẫu nhiên có giá trị 0< e <phi(n) và là số nguyên tố cùng nhau với Phi(n)
    do
    {
        Random RSA_rd = new Random();
        RSA_soE = RSA_rd.Next(2, RSA_soPhi_n);
    }
    while (!RSA_nguyenToCungNhau(RSA_soE, RSA_soPhi_n));
    rsa_soE.Text = RSA_soE.ToString();

    //Tính d là nghịch đảo modular của e
    RSA_soD = 0;
    int i = 2;
    while (((1 + i * RSA_soPhi_n) % RSA_soE) != 0 || RSA_soD <= 0)
    {
        i++;
        RSA_soD = (1 + i * RSA_soPhi_n) / RSA_soE;
    }
    rsa_soD.Text = RSA_soD.ToString();
}
```

❖ **Mã hóa**

```
1 reference
public void RSA_MaHoa(string ChuoiVao)
{
    // taoKhoa();
    // Chuyen xau thanh ma Unicode
    byte[] mh_temp1 = Encoding.Unicode.GetBytes(ChuoiVao);
    string base64 = Convert.ToBase64String(mh_temp1);

    // Chuyen xau thanh ma Unicode
    int[] mh_temp2 = new int[base64.Length];
    for (int i = 0; i < base64.Length; i++)
    {
        mh_temp2[i] = (int)base64[i];
    }

    //Mảng a chứa các kí tự đã mã hóa
    int[] mh_temp3 = new int[mh_temp2.Length];
    for (int i = 0; i < mh_temp2.Length; i++)
    {
        mh_temp3[i] = RSA_mod(mh_temp2[i], RSA_soE, RSA_soN); // mã hóa
    }

    //Chuyển sang kiểu kí tự trong bảng mã Unicode
    string str = "";
    for (int i = 0; i < mh_temp3.Length; i++)
    {
        str = str + (char)mh_temp3[i];
    }
    byte[] data = Encoding.Unicode.GetBytes(str);
    tx_banma1.Text = Convert.ToBase64String(data);
    dataa = data;

}
```

❖ Giải mã

```
// hàm giải mã
1 reference
public void RSA_GiaiMa(string ChuoiVao)
{
    byte[] temp2 = Convert.FromBase64String(ChuoiVao);
    string giaima = Encoding.Unicode.GetString(temp2);

    int[] b = new int[giaima.Length];
    for (int i = 0; i < giaima.Length; i++)
    {
        b[i] = (int)giaima[i];
    }
    //Giải mã
    int[] c = new int[b.Length];
    for (int i = 0; i < c.Length; i++)
    {
        c[i] = RSA_mod(b[i], RSA_soD, RSA_soN);// giải mã
    }

    string str = "";
    for (int i = 0; i < c.Length; i++)
    {
        str = str + (char)c[i];
    }
    byte[] data2 = Convert.FromBase64String(str);
    tx_banro2.Text = Encoding.Unicode.GetString(data2);

}
#endregion
```

❖ Một số thuật toán cần dùng

```
4 references
private bool RSA_kiemTraNguyenTo(int xi)
{
    bool kiemtra = true;
    if (xi == 2 || xi == 3)
    {
        // kiemtra = true;
        return kiemtra;
    }
    else
    {
        if (xi == 1 || xi % 2 == 0 || xi % 3 == 0)
        {
            kiemtra = false;
        }
        else
        {
            for (int i = 5; i <= Math.Sqrt(xi); i = i + 6)
                if (xi % i == 0 || xi % (i + 2) == 0)
                {
                    kiemtra = false;
                    break;
                }
        }
    }
    return kiemtra;
}
// "Hàm kiểm tra hai số nguyên tố cùng nhau"
1 reference
private bool RSA_nguyenToCungNhau(int ai, int bi)
{
    bool ktx_;
    // giải thuật Euclid;
    int temp;
    while (bi != 0)
    {
        temp = ai % bi;
        ai = bi;
        bi = temp;
    }
    if (ai == 1) { ktx_ = true; }
    else ktx_ = false;
    return ktx_;
}
```

2.5 Thiết kế chương trình, cài đặt thuật toán Java

2.5.1 Giới thiệu ngôn ngữ Java

Java là ngôn ngữ lập trình hướng đối tượng mạnh mẽ và được sử dụng rộng rãi, được biết đến với tính đơn giản, nền tảng độc lập và thư viện phong phú. Được tạo bởi James Gosling và nhóm của ông tại Sun Microsystems vào giữa những năm 1990, Java đã trở thành một trong những ngôn ngữ lập trình phổ biến nhất trên toàn cầu. Tính linh hoạt, hỗ trợ cộng đồng mạnh mẽ và hệ sinh thái phong phú khiến nó trở thành lựa chọn tuyệt vời để phát triển các ứng dụng phần mềm khác nhau, bao gồm các chương trình mã hóa như RSA.

Lý do chọn ngôn ngữ Java: Khi thiết kế chương trình mã hóa RSA, việc chọn ngôn ngữ lập trình phù hợp là rất quan trọng. Dưới đây là một số lý do thuyết phục tại sao Java là một lựa chọn lý tưởng:

1. **Độc lập nền tảng**: Nguyên tắc "viết một lần, chạy mọi nơi" của Java cho phép các chương trình được viết bằng Java chạy trên bất kỳ nền tảng nào với Máy ảo Java (JVM). Điều này có nghĩa là chương trình mã hóa RSA được phát triển bằng Java sẽ có thể mang theo được và có thể được thực thi trên nhiều hệ điều hành khác nhau, bao gồm Windows, macOS và Linux. Hơn nữa, đặc điểm này cho phép dễ dàng phân phối và triển khai chương trình trên các thiết bị khác nhau.
2. **Mạnh mẽ và Bảo mật**: Thiết kế của Java nhấn mạnh đến tính bảo mật, làm cho nó trở thành một ngôn ngữ tuyệt vời cho các tác vụ liên quan đến mã hóa. Nó cung cấp các cơ chế tích hợp để quản lý bộ nhớ, xử lý ngoại lệ và phát hiện lỗi thời gian chạy, giảm khả năng xảy ra lỗ hổng và cải thiện độ bền của chương trình. Ngoài ra, thư viện tiêu chuẩn mở rộng của Java bao gồm các API mật mã tạo thuận lợi cho việc triển khai các thuật toán mã hóa an toàn như RSA.

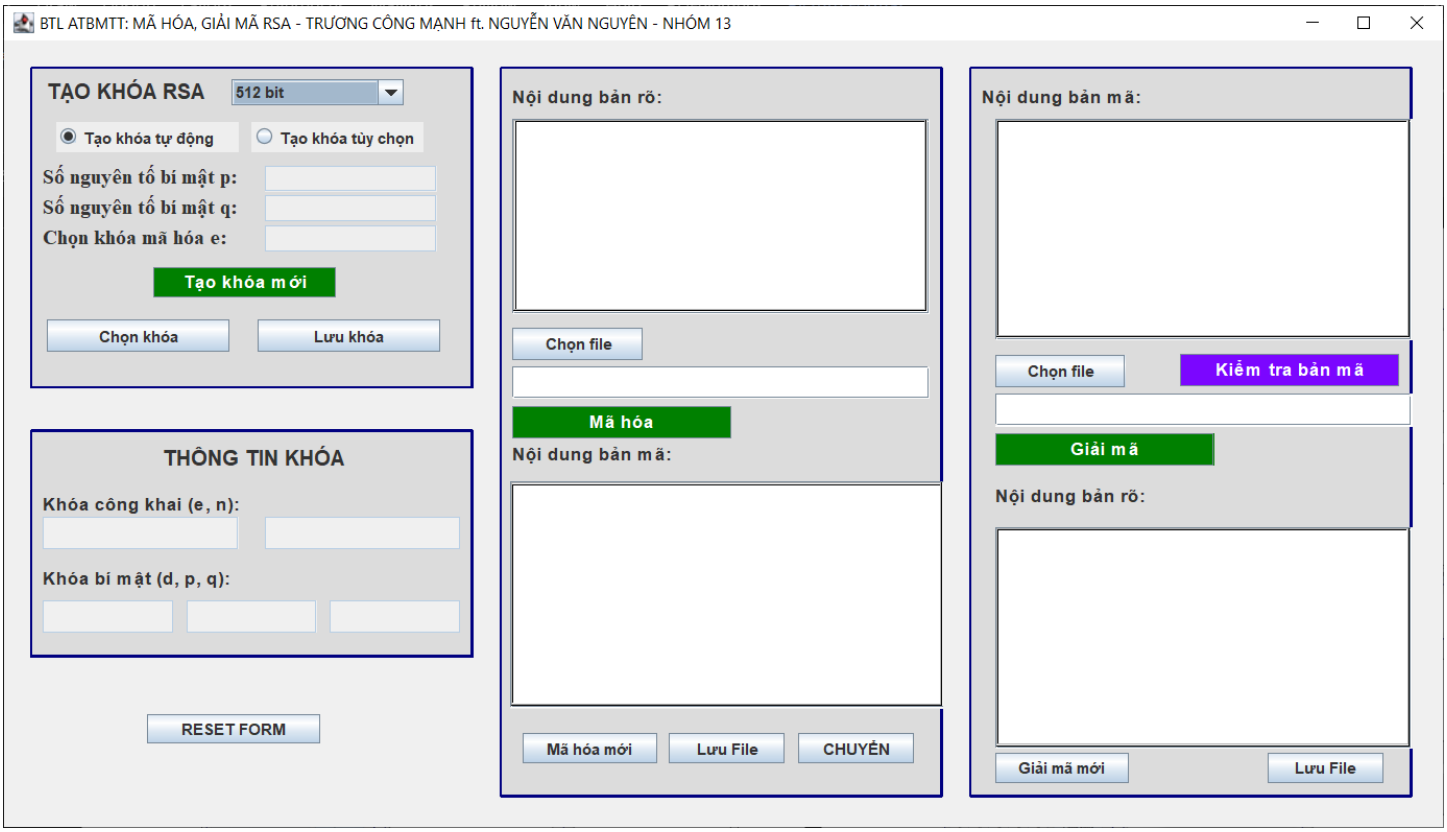
3. **Hỗ trợ thư viện rộng lớn**: Java tự hào có một bộ sưu tập lớn các thư viện và khung giúp đơn giản hóa các tác vụ lập trình phức tạp. Đối với mã hóa RSA, các thư viện như Java Cryptography Architecture (JCA) và Bouncy Castle cung cấp các chức năng mã hóa toàn diện, bao gồm tạo khóa, mã hóa và giải mã. Các thư viện này tuân thủ các giao thức mã hóa tiêu chuẩn, đảm bảo tính tương thích và khả năng tương tác với các hệ thống mật mã khác.

2.5.2 Thiết kế kịch bản chương trình

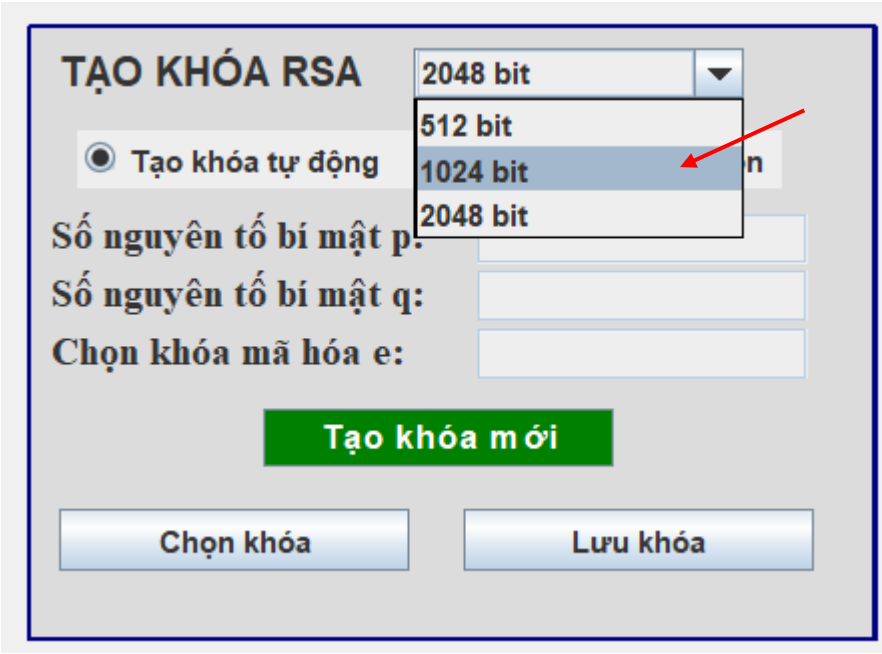
1. Yêu cầu người dùng nhập vào thông tin cần mã hóa, ví dụ như một chuỗi ký tự hoặc một số nguyên. Sử dụng lớp Scanner để nhận đầu vào từ người dùng.
2. Sinh hai số nguyên tố lớn p và q, có thể sử dụng hàm BigInteger.probablePrime() để sinh số nguyên tố ngẫu nhiên với độ dài bit chỉ định.
3. Tính giá trị của n = p x q và hàm Euler phi = (p-1) x (q-1).
4. Sinh khóa công khai e bằng cách chọn một số nguyên e ngẫu nhiên trong khoảng từ 1 đến phi, sao cho e và phi là hai số nguyên tố cùng nhau. Có thể sử dụng hàm gcd() để tính ước chung lớn nhất của hai số nguyên.
5. Sinh khóa bí mật d bằng cách tính nghịch đảo modulo của e theo phi, sử dụng phép toán modInverse().
6. Mã hóa thông tin bằng cách sử dụng khóa công khai e và số n để tính message^e mod n. Có thể sử dụng hàm modPow() để tính lũy thừa modulo.
7. Giải mã thông tin bằng cách sử dụng khóa bí mật d và số n để tính ciphertext^d mod n.
8. Hiển thị thông tin đã được mã hóa và giải mã lên màn hình.

2.5.3 Cài đặt thuật toán, hình ảnh minh hoạ

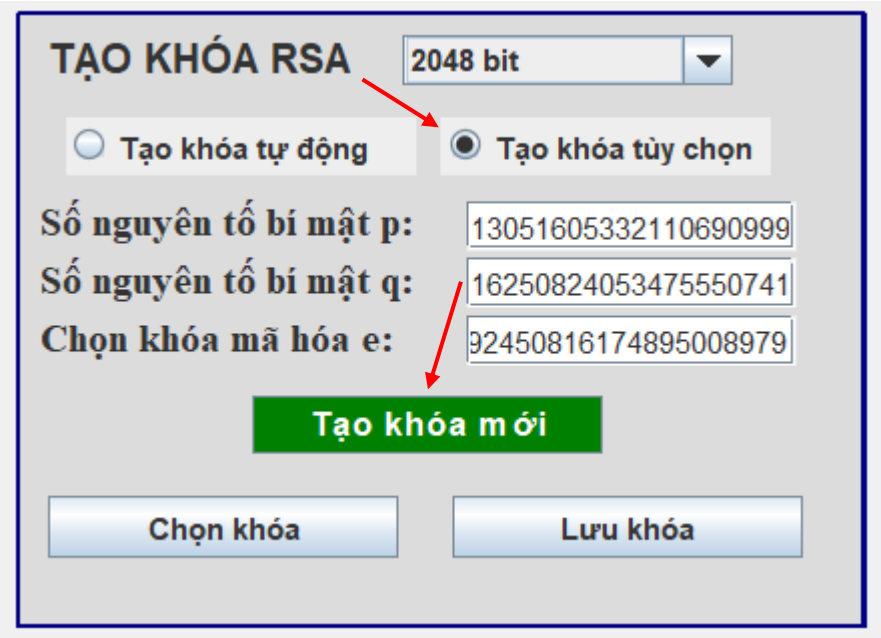
2.5.3.1 Hình ảnh minh họa



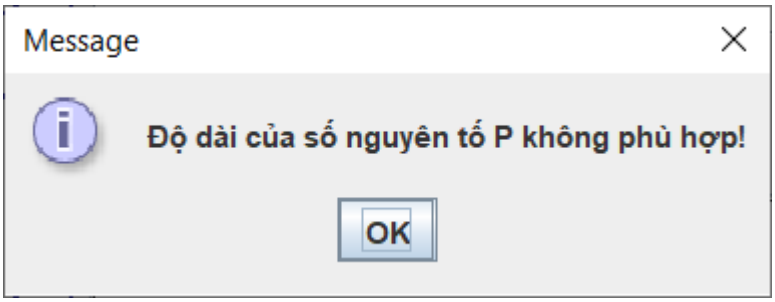
Hình 2.7 Giao diện chương trình

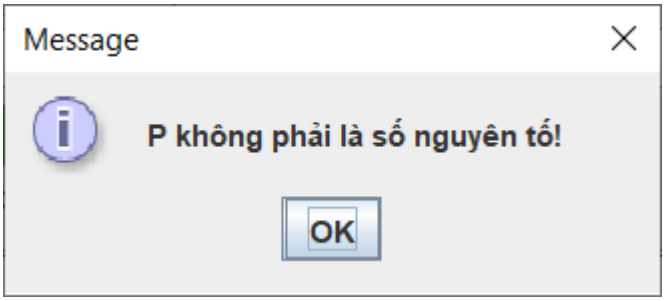


Hình 2.8 Chọn độ dài khóa

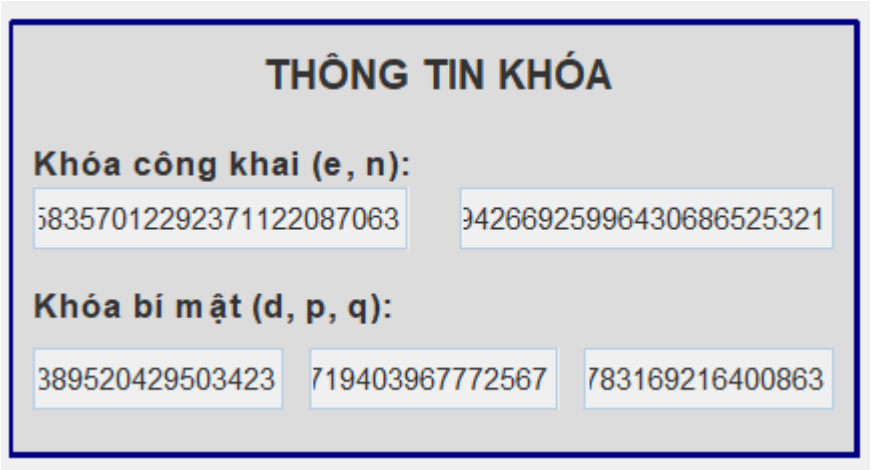


Hình 2.9 Sinh khóa tùy chọn cho phép nhập khóa

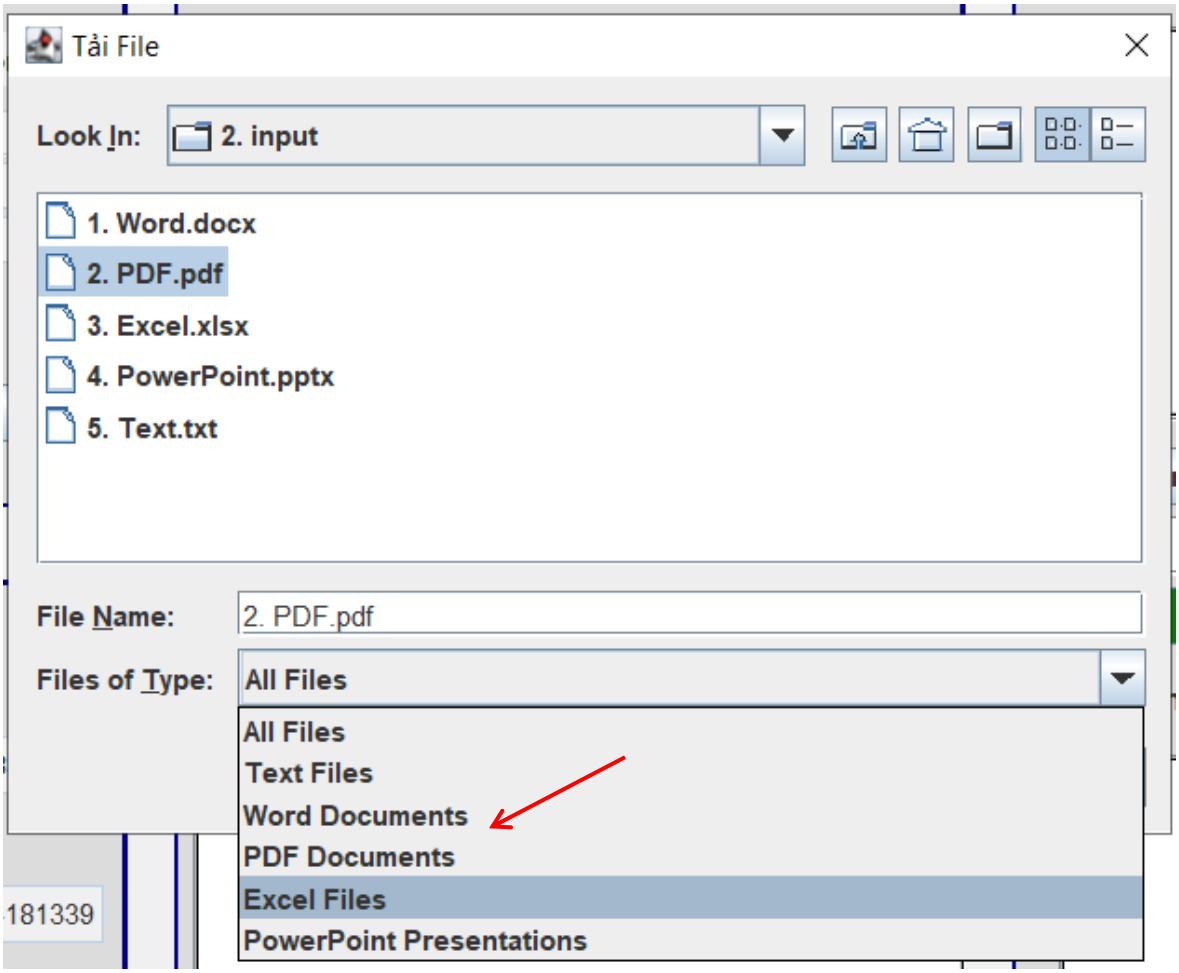




Hình 2.10 Thông báo lỗi khi khóa không đáp ứng

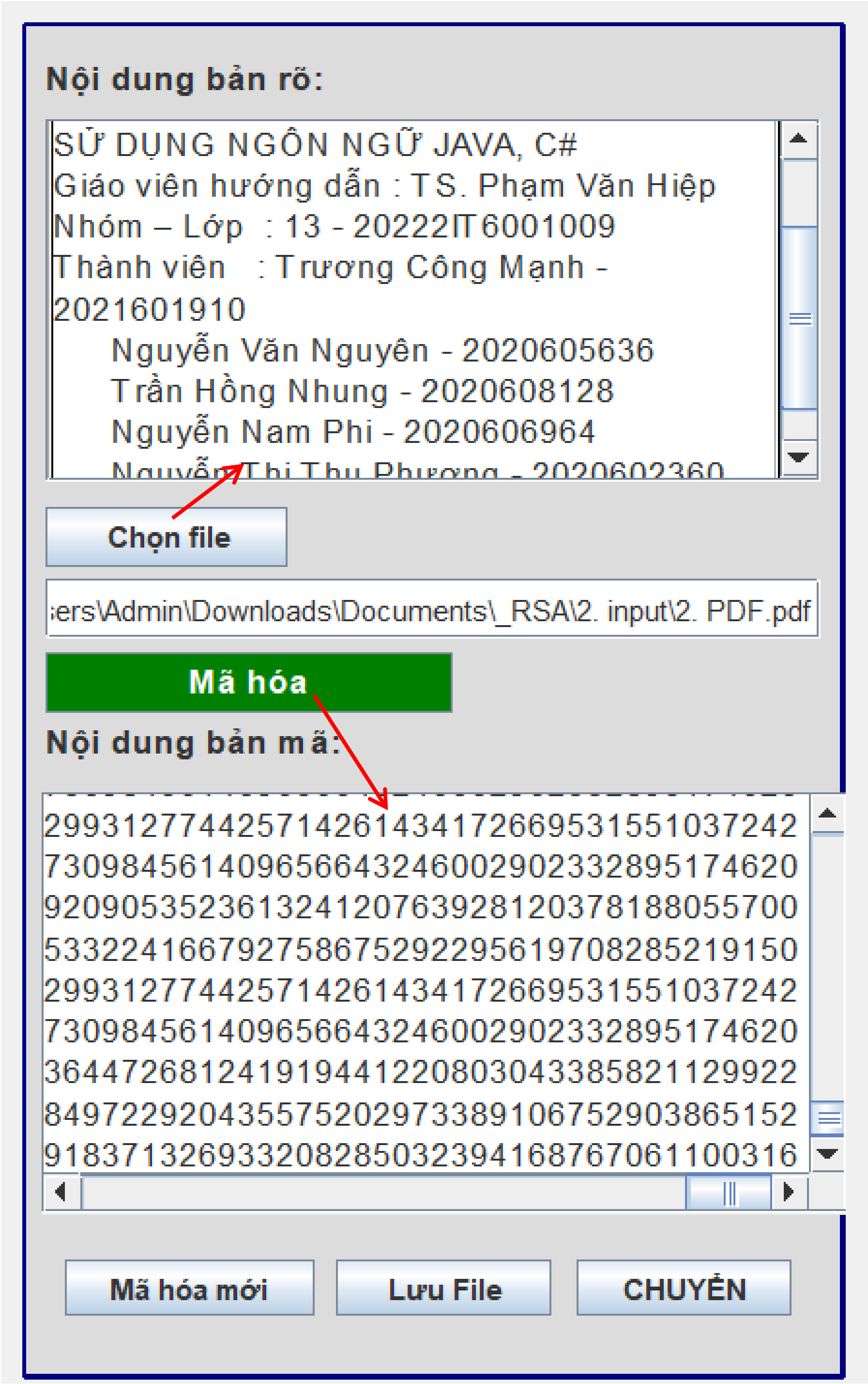


Hình 2.11 Giao diện hiển thị thông tin khóa

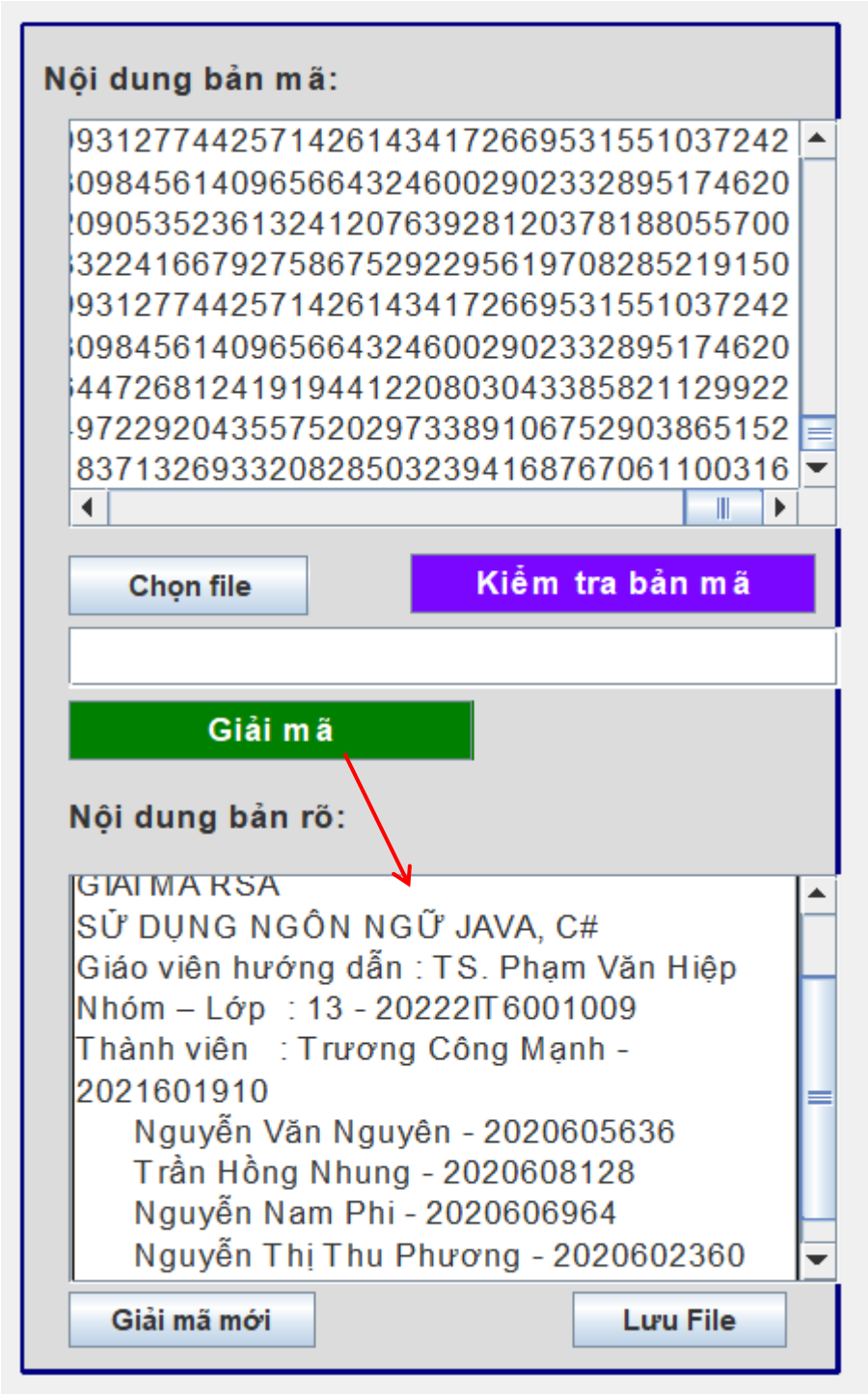


Hình 2.12 Hỗ trợ 5 định dạng tệp (.txt, Word, PDF, Excel, PowerPoint)

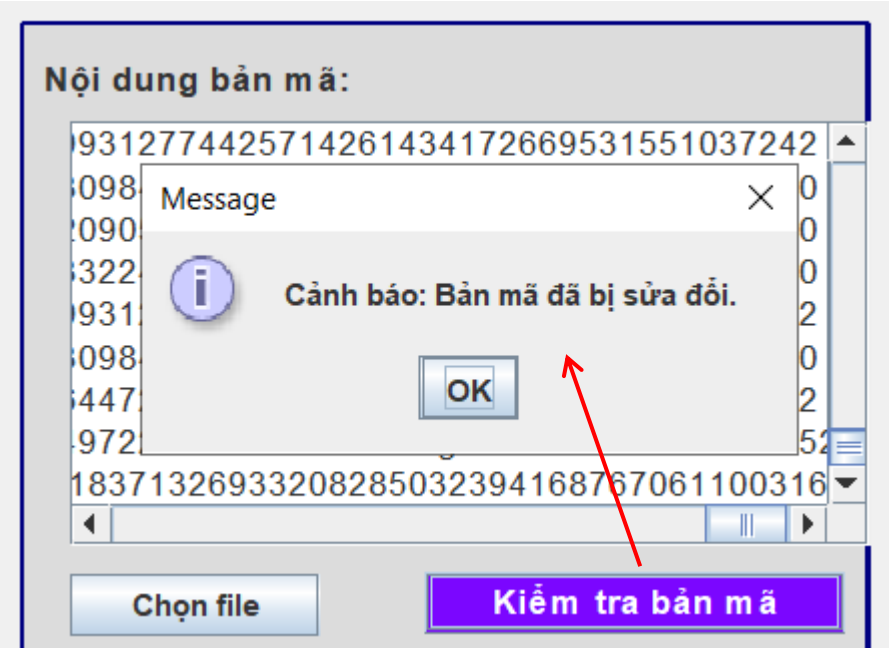
Hình 2.13 Tạo khóa tự động



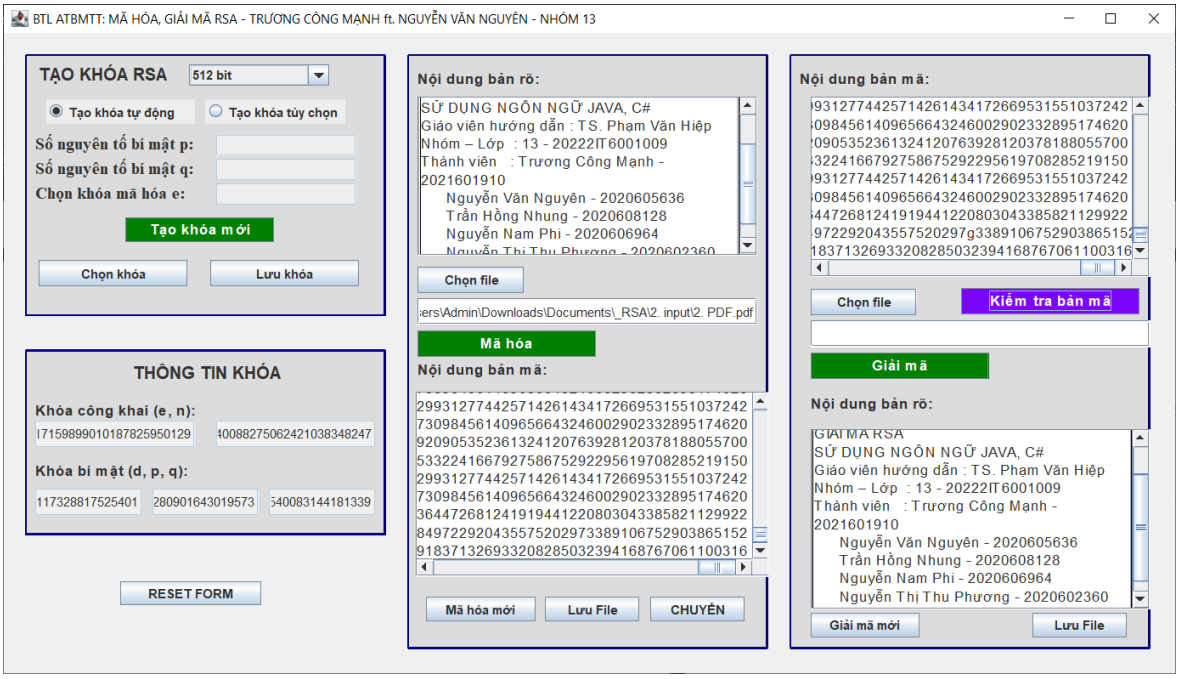
Hình 2.14 Mã hóa



Hình 2.15 Giải mã



Hình 2.16 Kiểm tra bản mã



Hình 2.17 Giao diện cuối cùng

Bước 1: Nhập vào dữ liệu cần thiết và nhấn nút **Tạo khóa mới** để tạo khóa (Tạo khóa tùy chọn) hoặc ấn **Tạo khóa mới** để tạo khóa ngẫu nhiên (Tạo khóa tự động).

Bước 2: Nhập vào dữ liệu đầu vào ở ô **Nội dung bản rõ** để nhập vào bản rõ cần mã hóa hoặc ấn **Chọn file** để chọn file văn bản chứa bản rõ.

Bước 3: Ấn nút **Mã hóa** để thực hiện mã hóa văn bản trong ô **Nội dung bản rõ** và hiện bản mã hiện vào ô **Nội dung bản mã.**

Bước 4: Ấn nút **Chuyển** để thực hiện đưa vản bản **ô Nội dung bản mã** phần **Mã hóa** sang ô **Nội dung bản mã** phần **Giải mã.** Hoặc ấn nút **Lưu file** để lưu bản mã vào 1 file văn bản.

Bước 5: Nhập vào dữ liệu đầu vào ở ô **Nội dung bản mã** để nhập vào bản mã cần giải hóa hoặc ấn **Chọn file** để chọn file văn bản chứa bản mã.

Bước 6: Ấn nút **Giải hóa** để thực hiện mã hóa văn bản trong ô **Nội dung bản mã** và hiện bản mã hiện vào ô **Nội dung bản rõ.**

2.5.3.2 Cài đặt thuật toán

❖ **Tạo Khóa**

```
/**
 * Hàm sinh tự động khóa RSA
 * @param bitLength - Chiều dài của khóa
 * @return
 */
public boolean automaticKeyGeneration(int bitLength) {
    BigInteger p, q, n, phiN, e, d;

    p = generateRandomPrime(bitLength);

    do {
        q = generateRandomPrime(bitLength);
    } while (p.compareTo(q) == 0);

    n = p.multiply(q);

    phiN = p.subtract(BigInteger.ONE);
    phiN = phiN.multiply(q.subtract(BigInteger.ONE));

    do {
        e = new BigInteger(2 * bitLength, new Random());
    } while(euclideanGCD(e, phiN).compareTo(BigInteger.ONE) != 0 ||
            (e.compareTo(BigInteger.ONE) == -1 && e.compareTo(phiN) == 1));

    d = extendedEuclideanInverse(e, phiN);

    this.rsa = new RSA(p, q, n, phiN, e, d);
    return true;
}
```

Hình 2.18 Cài đặt chức năng Sinh khóa tự động

```
/**
 * Kiểm tra tạo khóa thủ công
 * @param p
 * @param q
 * @param e
 * @return
 */
public boolean checkManualKeyGeneration(BigInteger p, BigInteger q, BigInteger e) {
    if (!millerRabinTest(p) || !millerRabinTest(q) || p.compareTo(q) == 0)
        return false;

    BigInteger n = p.multiply(q);
    BigInteger phiN = p.subtract(BigInteger.ONE);
    phiN = phiN.multiply(q.subtract(BigInteger.ONE));

    if (e.compareTo(BigInteger.ONE) == -1 && e.compareTo(phiN) == 1)
        return false;

    if (euclideanGCD(e, phiN).compareTo(BigInteger.ONE) != 0)
        return false;

    BigInteger d = extendedEuclideanInverse(e, phiN);

    this.rsa = new RSA(p, q, n, phiN, e, d);
    return true;
}
```

Hình 2.19 Cài đặt chức năng Tạo khóa thủ công

❖ Mã hóa

```java
/**
 * Mã hóa RSA
 * @param plainText - bản rõ
 * @return bản mã của RSA
 */
public String encryptRSA(String message) {
    BigInteger e = rsa.getE();
    BigInteger n = rsa.getN();

    StringBuilder encrypted = new StringBuilder();
    for (int i = 0; i < message.length(); i++) {
        char character = message.charAt(i);
        long characterValue = (long) character;
        BigInteger characterBigInt = new BigInteger(Long.toString(characterValue));
        BigInteger encryptedChar = squareAndMultiply(characterBigInt, e, n);
        encrypted.append(encryptedChar).append(" ");
    }

    return encrypted.toString();
}
```

Hình 2.20 Cài đặt chức năng Mã hóa

❖ Giải mã

```java
/**
 * Giải mã RSA
 * @param cipherText - bản mã
 * @return bản rõ
 */
public String decryptRSA(String message) {
    BigInteger d = rsa.getD();
    BigInteger n = rsa.getN();

    String[] messageArray = message.trim().split("\\s+");
    BigInteger[] encrypted = new BigInteger[messageArray.length];

    for (int i = 0; i < messageArray.length; i++) {
        encrypted[i] = new BigInteger(messageArray[i]);
    }

    StringBuilder decrypted = new StringBuilder();
    for (BigInteger cipher : encrypted) {
        BigInteger decryptedBigInt = squareAndMultiply(cipher, d, n);
        long decryptedValue = decryptedBigInt.longValue();
        char decryptedChar = (char) decryptedValue;
        decrypted.append(decryptedChar);
    }

    return decrypted.toString();
}
```

Hình 2.21 Cài đặt chức năng Giải mã

❖ Một số thuật toán cần dùng

```
/**
 * Kiểm tra số nguyên tố bằng thuật toán Miller - Ranbin
 * @param n - số cần kiểm tra có phải là số nguyên tố
 * @return
 */
public boolean millerRabinTest(BigInteger n) {
    // Kiểm tra các số nguyên tố nhỏ
    if (n.equals(BigInteger.TWO) || n.equals(BigInteger.valueOf(3))) {
        return true;
    }

    // Kiểm tra số chẵn hoặc số nhỏ hơn 2
    if (n.compareTo(BigInteger.TWO) < 0 || n.mod(BigInteger.TWO).equals(BigInteger.ZERO)) {
        return false;
    }

    // Write n-1 as 2^s * d
    int s = 0;
    BigInteger d = n.subtract(BigInteger.ONE);
    while (d.mod(BigInteger.TWO).equals(BigInteger.ZERO)) {
        d = d.divide(BigInteger.TWO);
        s++;
    }

    // Thực hiện kiểm tra tính nguyên tố Miller-Rabin k lần
    for (int i = 0; i < ITERATIONS_MILLER_RABIN; i++) {
        // Sinh một số ngẫu nghiên từ 2 đến n - 1
        BigInteger a = randomBigInt(BigInteger.TWO, n.subtract(BigInteger.ONE));

        // Tính x = a^d mod n
        BigInteger x = squareAndMultiply(a, d, n);

        // Kiểm tra nếu x = 1 hoặc x = n - 1
        if (x.equals(BigInteger.ONE) || x.equals(n.subtract(BigInteger.ONE))) {
            continue;
        }

        boolean isPrime = false;
```

```
        // Lặp lại x^2 và kiểm tra 1 hoặc n-1
        for (int r = 1; r < s; r++) {
            x = squareAndMultiply(x, BigInteger.TWO, n);

            // Nếu x trở thành 1, số đó là hợp số
            if (x.equals(BigInteger.ONE)) {
                return false;
            }

            // Nếu x trở thành n-1, thì đó là số nguyên tố khả dĩ
            if (x.equals(n.subtract(BigInteger.ONE))) {
                isPrime = true;
                break;
            }
        }

        // Nếu không tìm thấy số nguyên tố nào thì số đó là hợp số
        if (!isPrime) {
            return false;
        }
    }

    // Sau khi vượt qua tất cả vòng lặp, n có thể là số nguyên tố
    return true;
}
```

Hình 2.22 Cài đặt thuật toán Miller-Rabin

```java
/**
 * Euclid - tìm ước chung lớn nhất GCD(a, b) <br/>
 * ỨNG DỤNG kiểm tra điều kiện của e (khóa công khai): GCD(e, phiN) = 1
 * @param a
 * @param b
 * @return ước chung lớn nhất của a và b
 */
public BigInteger euclideanGCD(BigInteger a, BigInteger b) {
    if (b.compareTo(BigInteger.ZERO) == 0)
        return a;
    return euclideanGCD(b, a.mod(b));
}
```

Hình 2.23 Cài đặt thuật toán Eculid

```java
/**
 * Euclid mở rộng - tìm nghịch đảo modulo: a^-1 mod n <br/>
 * ỨNG DỤNG tìm d (khóa bí mật): d = e^-1 mod phiN
 * @param a
 * @param n
 * @return số nghịch đảo của a trong modulo n
 */
public BigInteger extendedEuclideanInverse(BigInteger a, BigInteger n) {
    BigInteger r0 = n, r1 = a;
    BigInteger r2 = r0.mod(r1);
    BigInteger q = r0.divide(r1);
    BigInteger t0 = BigInteger.ZERO, t1 = BigInteger.ONE;

    while (r2.compareTo(BigInteger.ZERO) != 0) {
        BigInteger tmp = t0.subtract(q.multiply(t1));
        tmp = tmp.mod(n);
        r0 = r1;
        r1 = r2;
        r2 = r0.mod(r1);
        q = r0.divide(r1);
        t0 = t1;
        t1 = tmp;
    }

    if (r1.compareTo(BigInteger.ONE) == 0)
        return t1;

    System.out.println("Khong tim duoc " + a + "^-1 mod " + n);
    return BigInteger.valueOf(-1);
}
```

Hình 2.24 Cài đặt thuật toán Eculid mở rộng

```java
/**
 * Bình phương và nhân - tìm: a^k mod n <br/>
 * ỨNG DỤNG để mã hóa, giải mã RSA <br/>
 * Encrypt: cipherText = plainText^e mod n <br/>
 * Decrypt: plainText = cipherText^d mod n
 * @param a - giá trị unicode của một ký tự
 * @param k - số mũ
 * @param n - giá trị (không gian) modulo
 * @return kết quả
 */
public BigInteger squareAndMultiply(BigInteger a, BigInteger k, BigInteger n) {
    BigInteger result = BigInteger.ONE;
    BigInteger base = a;
    BigInteger exponent = k;

    BigInteger[] convertToBinary = new BigInteger[100_000];
    int count = 0;
    while (exponent.compareTo(BigInteger.ZERO) > 0) {
        convertToBinary[count] = exponent.mod(BigInteger.valueOf(2));
        exponent = exponent.divide(BigInteger.valueOf(2));
        count++;
    }

    for (int i = count - 1; i >= 0; i--) {
        result = result.multiply(result);
        result = result.mod(n);
        if (convertToBinary[i].compareTo(BigInteger.ONE) == 0) {
            result = result.multiply(base);
            result = result.mod(n);
        }
    }
    return result;
}
```

Hình 2.25 Cài đặt thuật toán bình phương và nhân

CHƯƠNG 3.
KẾT LUẬN VÀ BÀI HỌC KINH NGHIỆM

3.1 Kiến thức kỹ năng đã học được trong quá trình thực hiện đề tài.

3.1.1 Các kiến thức đã lĩnh hội

- Hiểu được về hệ mã hóa công khai và hệ mật mã RSA
- Tạo được khóa tự động và khóa tùy chọn
- Quá trình mã hóa RSA
- Quá trình giải mã RSA
- Thực hiện được chương trình demo bằng 2 ngôn ngữ Java và C#

3.1.2 Các kỹ năng đã tiếp thu

- Đánh giá được vai trò của bảo mật thông tin, các cơ chế, yêu cần an toàn bảo mật thông tin, các loại hình tấn công và các mức độ bảo vệ.
- Phân tích được các kỹ thuật sử dụng để mã hóa và giải mã thông tin.
- Hiểu và áp dụng các thuật toán liên quan đến hệ mã hóa RSA như (thuật toán sinh khóa, thuật toán mã hóa, thuật toán giải mã) vào việc mã hóa và giải mã để giải quyết bài toán có tính ứng dụng vào thực tiễn.
- Tổ chức được hoạt động nhóm.
- Bên cạnh đó còn có kỹ năng quản lý thời gian hiệu quả, kỹ năng viết báo cáo, thành thạo các công cụ trong bộ office…
- Biết thêm về hai ngôn ngữ lập trình khác nhau là Java và C#

3.2 Bài học kinh nghiệm

- Nắm rõ kỹ năng xác định vấn đề, kĩ năng phân tích vấn đề và sàng lọc ý kiến.
- Nhóm trưởng cần xác định vai trò của từng thành viên, kiểm soát công việc tối ưu nhất và đưa ra những quyết định đúng đắn.
- Biết lắng nghe, tôn trọng ý kiến của từng thành viên.

- Đặt tinh thần trách nhiệm trong công việc thành ưu tiên hàng đầu.
- Có thêm kỹ năng sử dụng tra tài liệu bằng tiếng anh

3.3 Đề xuất về tính khả thi của chủ đề nghiên cứu, những thuận lợi, khó khăn

3.3.1 Tính khả thi của đề tài

- Chủ đề nghiên cứu của nhóm chúng em khá phù hợp với thời gian được cho phép để hoàn thiện bài tập lớn, bên cạnh đó các thuật toán và mã hóa đều đã có sẵn được thử nghiệm bởi các nhà nghiên cứu bảo mật nên trong thời gian nghiên cứu, nhóm nhận thấy cần phải thực sự hiểu rõ về hệ mật mã RSA
- Các thư viện có sẵn trên mạng hỗ trợ tối đa và có kỹ thuật lập trình đã được học ở những học phần trước nên chúng em có thể hoàn thành được đề tài này

3.3.2 Những thuận lợi và khó khăn nhóm gặp phải

❖ ***Thuận lợi***

- Có thể đọc hiểu được tài liệu tiếng anh nên có thể dễ dàng tiếp cận các nguồn tài liệu chính thống.
- Các thành viên trong nhóm hòa đồng, cởi mở, tương tác với các thành viên khác trong nhóm khá sôi nổi nên các công việc liên quan đến cả nhóm thường diễn r suôn sẻ.

❖ ***Khó khăn***

❖ Công đoạn thiết giao diện phần mềm mã hóa chưa được bắt mắt do chưa có nhiều kinh nghiệm trong kỹ thuật xử lý giao diện.

❖ Ở phần xử lý về File Docx sử dụng để lấy bản rõ khá mới mẻ với các thành viên trong nhóm nên giai đoạn hoàn thiện chức năng tốn khá nhiều thời gian.

KẾT LUẬN

Trên đây nhóm 13 đã trình bày toàn bộ nội dung về xây dựng chương trình mã hóa và giải mã mật mã khóa công khai RSA. Để tìm hiểu về cách hoạt động của mật mã khóa công khai, nhóm 13 đã tìm hiểu về cơ sở toán học, các thuật toán để xây dựng về mã hóa RSA. Mã hóa RSA hoạt động dựa trên cơ sở toán học như các định lý Fermat, định lý Euler, kiểm tra số nguyên tố và các thuật toán Euler, thuật toán Miller-Rabin, thuật toán Bình phương và nhân, thuật toán RSA. Chương trình mã hóa và giải mã được xây dựng trên ngôn ngữ Java, C#.

Nhóm đã cố gắng hoàn thiện bài thực nghiệm một cách tốt nhất, rấtmong được ý kiến của thầy cô và các bạn để bài cáo cáo được hoàn thiện hơn. Đồng thời, nhóm xin chân thành cảm ơn TS. Phạm Văn Hiệp- giảng viên hướng dẫn trực tiếp, và các thầy cô khác trong bộ môn tận tình giúp đỡ và tạo điều kiện để nhóm em hoàn thành bài thực nghiệm này

TÀI LIỆU THAM KHẢO

[1] TS. Phạm Văn Hiệp – Bài giảng, tài liệu môn *An toàn bảo mật thông tin* – Đại học Công nghiệp Hà Nội.

[2] Tài liệu điện tử trên Internet.

[3] Bùi Doãn Khanh, Nguyễn Đình Thúc, *Mã hóa thông tin – Lý thuyết và ứng dụng*, NXB Lao động xã hội, 2011.