TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI

KHOA CÔNG NGHỆ THÔNG TIN

======*======**

BÁO CÁO THỰC NGHIỆM

HỌC PHẦN AN NINH MẠNG

ĐỀ TÀI

XÂY DỰNG ỨNG DỤNG CHAT BẢO MẬT END-TO-END SỬ DỤNG HỆ MẬT RSA

GVHD:	ThS. Lê Thị Anh
Lớp:	20241IT6070002
Nhóm:	06
Sinh viên :	Nguyễn Thị Hoàng Mai - 2021607135
	Trần Văn Nhã - 2022603089
	Đỗ Hoài Phong - 2021603816
	Đào Thị Hương Quỳnh - 2022602154

Hà Nội – 2024

TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI

KHOA CÔNG NGHỆ THÔNG TIN

======***======

BÁO CÁO THỰC NGHIỆM

HỌC PHẦN AN NINH MẠNG

ĐỀ TÀI

XÂY DỰNG ỨNG DỤNG CHAT BẢO MẬT END-TO-END SỬ DỤNG HỆ MẬT RSA

GVHD: ThS. Lê Thị Anh

Lớp: 20241IT6070002

Nhóm: 06

Sinh viên : Nguyễn Thị Hoàng Mai - 2021607135

Trần Văn Nhã - 2022603089

Đỗ Hoài Phong - 2021603816

Đào Thị Hương Quỳnh - 2022602154

Hà Nội – 2024

MỤC LỤC

PHẦN 1: MỞ ĐẦU 4
PHẦN 2: NỘI DUNG 6
Chương 1: Tổng quan về mã hóa 6
1.1. Cơ sở lý thuyết chung về mật mã học và thám mã 6
1.1.1. Mã hóa 6
1.1.2. Thám mã (cryptanalysis) 7
1.2 Các loại mã hóa 7
1.2.1. Mã hóa đối xứng 7
1.2.2. Mã hóa bất đối xứng 9
1.3. Giới thiệu về mã hóa công khai 11
1.3.1. Mã hóa công khai 11
1.3.2. Hệ mật RSA 12
1.4. Quy trình hoạt động 12
1.5. Ứng dụng thực tế của hệ mật RSA 14
1.5.1. Chữ ký điện tử (Digital Signature) 14
1.5.2. SSL/TLS (Secure Socket Layer/Transport Layer Security) 15
Chương 2: Phân tích thiết kế 16
2.1. Công nghệ sử dụng 16
2.1.1. Ngôn ngữ lập trình C# 16
2.1.2. Framework WPF 17
2.1.3. Hàm băm sử dụng Argon2 18
2.1.4. SQL server 19
2.2. Giới thiệu sản phẩm 21
2.2.1. Giao diện ứng dụng 21
2.2.2 Xử lý lỗi và bảo mật 24
Chương 3: Kết quả thực hiện 29
3.1. Quy trình mã hóa RSA 29
3.1.1. Quy trình tạo khóa 29
3.1.2. Quy trình trao đổi khóa công khai 29
3.1.3. Quy trình mã hóa 30
3.1.4. Quy trình giải mã 30
3.1.5. Quy trình xác thực chữ ký số 30
3.2. Demo sản phẩm (Trực tiếp) 31
PHẦN 3: KẾT LUẬN 32
TÀI LIỆU THAM KHẢO 35

DANH MỤC HÌNH ẢNH

Hình 1.1: Mô tả thông plaintext và ciphertext 6
Hình 1.2. Mô tả quy trình hoạt động mã hóa đối xứng 8
Hình 1.3: Mô tả quy trình hoạt động của thuật toán 3DES 9
Hình 1.4: Mô tả quá trình truyền và sử dụng khóa công khai 10
Hình 1.5: Mô tả quá trình gửi tin nhắn từ Bob đến Alice 11
Hình 1.6: Mô tả quá trình hoạt động của thuật toán RSA 12
Hình 1.7: Quy trình thực hiện chữ ký điện tử : 14
Hình 2.1: Giao diện đăng nhập 21
Hình 2.2: Giao diện đăng ký tài khoản 21
Hình 2.3: Giao diện trang chủ 22
Hình 2.4: Giao diện chat của Alice 23
Hình 2.5: Giao diện chat của Bob 24
Hình 2.6: Báo lỗi email sai định dạng 24
Hình 2.7: Báo lỗi khi không nhập vào các trường 25
Hình 2.8: Đăng ký thành công tài khoản 25
Hình 2.9: Báo lỗi khi tài khoản không tồn tại 25
Hình 2.10: Thông báo lỗi khi tạo chữ ký số 26
Hình 2.11: Thông báo lỗi khi giải mã 26
Hình 2.12: Thông báo lỗi khi tạo chữ ký số 27
Hình 3.1: Quy trình tạo khóa 29
Hình 3.2: Quá trình xác thực chữ ký số 31

DANH MỤC BẢNG BIỂU

Bảng 1: Các loại thuật toán mã hóa 7
Bảng 1.2: Ưu và nhược điểm của C# 16
Bảng 1.3: Ưu điểm và hạn chế của WPF 17
Bảng 1.4: Ưu và nhược điểm của Argon2 19
Bảng 1.5: Ưu và nhược điểm của SQL Server 20

PHẦN 1: MỞ ĐẦU

1. Lý do chọn đề tài

Ngày nay, khi công nghệ số đã và đang trở thành một phần không thể thiếu trong cuộc sống của con người, việc bảo mật thông tin cũng là một trong những vấn đề cấp bách hàng đầu, đòi hỏi sự quan tâm đặc biệt của tất cả mọi người. Nhu cầu bảo vệ thông tin cá nhân, thông tin doanh nghiệp và những thông điệp truyền tải trong không gian số đã thúc đẩy sự phát triển mạnh mẽ của các phương pháp mã hóa, giúp người dùng có nhiều lựa chọn hơn để đảm bảo an toàn trên môi trường mạng. Trong số các giải pháp mã hóa, thuật toán RSA nổi lên như một trong những phương pháp mã hóa khóa công khai phổ biến và đáng tin cậy nhất. Với khả năng đảm bảo an toàn dữ liệu và đảm bảo tối ưu hóa hiệu suất, RSA được ứng dụng rộng rãi trong các hệ thống bảo mật, đặc biệt là trong việc bảo vệ thông tin liên lạc qua môi trường số.

Chính vì những ưu điểm nổi bật của hệ mật mã hóa khóa công khai RSA, kết hợp với sự gia tăng nhanh chóng của các mối đe dọa an ninh trên môi trường mạng, chúng em đã quyết định lựa chọn đề tài "Xây dựng ứng dụng chat bảo mật End-to-End sử dụng hệ mật RSA". Với mong muốn xây dựng một kênh giao tiếp an toàn, đảm bảo mọi tin nhắn đều được mã hóa từ đầu đến cuối, giúp chống lại các hành vi truy cập trái phép. Ứng dụng được kỳ vọng sẽ là giải pháp bảo mật an toàn và hiệu quả, đảm bảo mức bảo vệ tối đa cho thông tin của người dùng.

2. Mục tiêu nghiên cứu

Cơ sở lý thuyết về mã hóa nói chung, mã hóa công khai và thuật toán RSA nói riêng, quy trình xây dựng ứng dụng chat hỗ trợ mã hóa end-to-end bằng RSA. Thực hiện mô phỏng quá trình mã hóa, giải mã tin nhắn và truyền tin, sau đó thực hiện đánh giá hiệu quả của ứng dụng trong việc bảo mật thông tin.

3. Đối tượng nghiên cứu

Thuật toán mã hóa RSA: Cơ chế hoạt động, quy trình khởi tạo khóa, mã hóa, và giải mã.

Ứng dụng chat bảo mật: Phân tích thiết kế, các yếu tố công nghệ, yêu cầu bảo mật và tính năng cần thiết.

4. Phạm vi nghiên cứu

Tập trung vào việc ứng dụng hệ mật RSA để mã hóa và giải mã tin nhắn.

Chỉ nghiên cứu trong phạm vi xây dựng ứng dụng chat cơ bản, không mở rộng sang các hệ thống phức tạp hoặc tích hợp nhiều tính năng.

Đánh giá trên quy mô mô phỏng, chưa áp dụng trên môi trường thực tế rộng lớn.

5. Ý nghĩa khoa học và thực tiễn đề tài

Ý nghĩa khoa học: Đóng góp vào việc nghiên cứu và ứng dụng thuật toán RSA trong lĩnh vực trao đổi và bảo mật thông tin.

Ý nghĩa thực tiễn: Cung cấp giải pháp chat bảo mật, góp phần nâng cao ý thức về quyền riêng tư và bảo mật dữ liệu cá nhân cho người dùng.

6. Cấu trúc báo cáo:

Phần 1: Mở đầu: Trình bày lý do chọn đề tài, mục tiêu nghiên cứu, đối tượng nghiên cứu, phạm vi nghiên cứu, ý nghĩa khoa học và thực tiễn đề tài.

Phần 2: Nội dung:

- Chương 1: Tổng quan về mã hóa
- Chương 2: Phân tích và thiết kế
- Chương 3: Kết quả thực hiện

Phần 3: Kết Luận: Tóm tắt kết quả nghiên cứu và đóng góp của đề tài định hướng phát triển trong tương lai.

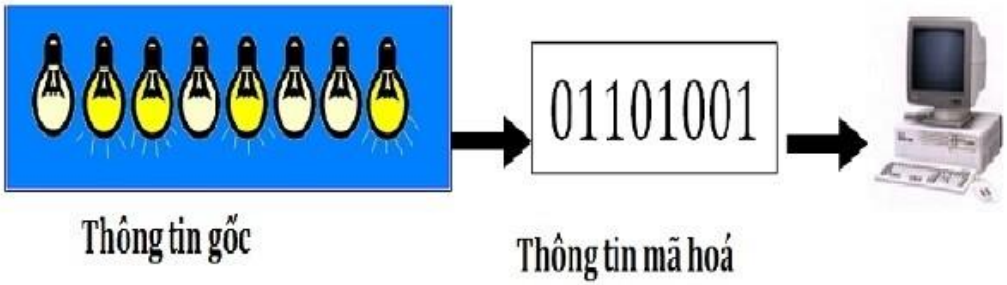
PHẦN 2: NỘI DUNG

Chương 1: Tổng quan về mã hóa

1.1. Cơ sở lý thuyết chung về mật mã học và thám mã

1.1.1. Mã hóa

- Mã hóa là quá trình chuyển đổi thông tin từ bản rõ (gọi là "plaintext" hoặc "cleartext") thành dạng không đọc được hoặc khó đọc được (gọi là "ciphertext") bằng cách sử dụng một thuật toán mã hóa và một khóa. Mục đích chính của mã hóa là bảo vệ tính bảo mật của thông tin khi nó được truyền qua mạng hoặc lưu trữ trong các hệ thống.
- Hiểu một cách đơn giản, mã hóa thông tin chính là chuyển dữ liệu ban đầu A thành dữ liệu B. Để đọc được dữ liệu A, người đọc phải giải mã được dữ liệu đã được biến đổi B về A.
- Mã hóa thông tin xuất hiện ở rất nhiều lĩnh vực của cuộc sống, trong đó có tin học. Trong lĩnh vực này, dữ liệu chính là thông tin đã được đưa vào máy tính. Vì vậy, mã hóa thông tin thành dữ liệu chính là quá trình biểu diễn thông tin dưới dạng bit để máy tính có thể hiểu, lưu trữ, xử lý thông tin như mong muốn của con người.



Hình 1.1: Mô tả thông plaintext và ciphertext

- Mã hóa thông tin là một công việc rất cần thiết và quan trọng bởi chúng đảm bảo sự riêng tư, xác thực, bảo mật và sự toàn vẹn của dữ liệu. Mật mã học giúp bảo đảm các yếu tố sau cho dữ liệu:

  + Tính bí mật: Thông tin chỉ được tiết lộ cho những ai được phép
  + Tính toàn vẹn: Thông tin không thể bị thay đổi mà không bị phát hiện.

+ Tính xác thực: Người gửi/ nhận có thể chứng minh đúng họ.
+ Tính chống chối bỏ: Người gửi hoặc nhận sau này không thể chối bỏ việc đã gửi hoặc nhận thông tin.
+ Tính sẵn sàng: Đảm bảo truy cập, sử dụng thông tin kịp thời và đáng tin cậy.

– Có rất nhiều các thuật toán mã hóa khác nhau. Từ những thuật toán được công khai để mọi người sử dụng và áp dụng như là một chuẩn chung cho việc mã hoá dữ liệu đến những thuật toán mã hóa chưa được công bố. Có thể phân loại các thuật toán mã hóa như sau:

Bảng 1: Các loại thuật toán mã hóa

Theo phương pháp	Theo số lượng khóa
– Mã hóa đối xứng – Mã hóa bất đối xứng	– Mã hóa khóa bí mật – Mã hóa khóa công khai

1.1.2. Thám mã (cryptanalysis)

Thám mã hay còn gọi là phân tích mật mã – đây là ngành học nghiên cứu các phương thức để thu được ý nghĩa của thông tin đã được mã hóa.

Các phương pháp tấn công thám mã:

– Tìm khóa vét cạn.
– Phân tích thống kê.
– Phân tích toán học.

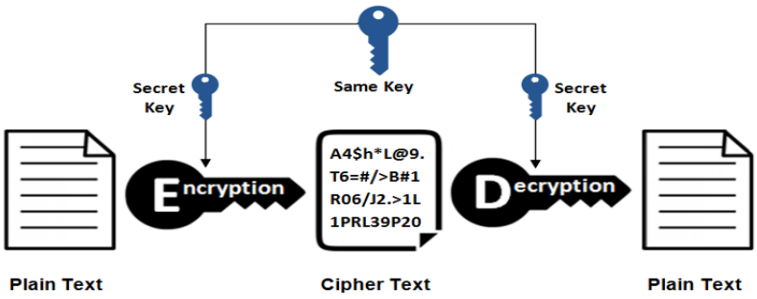
1.2 Các loại mã hóa

1.2.1. Mã hóa đối xứng

– Mã hoá đối xứng còn có một số tên gọi khác như Secret Key Cryptography (hay Private Key Cryptography), sử dụng cùng một khoá cho cả hai quá trình mã hoá và giải mã.
– Quy trình hoạt động:

Trước khi truyền dữ liệu, 2 bên gửi và nhận phải thoả thuận về khoá dùng chung cho quá trình mã hoá và giải mã. Sau đó, bên gửi sẽ mã hoá bản rõ bằng cách sử dụng khoá bí mật này và gửi thông điệp đã mã hoá cho bên nhận. Bên nhận sau

khi nhận được thông điệp đã mã hoá sẽ sử dụng chính khoá bí mật mà hai bên thỏa thuận để giải mã và lấy lại bản rõ.



Hình 1.2. Mô tả quy trình hoạt động mã hóa đối xứng

Hình vẽ trên chính là quá trình tiến hành trao đổi thông tin giữa bên gửi và bên nhận thông qua việc sử dụng phương pháp mã hoá đối xứng. Trong quá trình này, thì thành phần quan trọng nhất cần phải được giữ bí mật chính là khoá. Việc trao đổi, thoả thuận về thuật toán được sử dụng trong việc mã hoá có thể tiến hành một cách công khai, nhưng bước thỏa thuận về khóa trong việc mã hoá và giải mã phải tiến hành bí mật. Thuật toán mã hoá đối xứng sẽ rất có lợi khi được áp dụng trong các cơ quan hay tổ chức đơn lẻ. Nhưng nếu cần phải trao đổi thông tin với một bên thứ ba thì việc đảm bảo tính bí mật của khóa phải được đặt lên hàng đầu. Mã hoá đối xứng có thể được phân thành 02 loại:

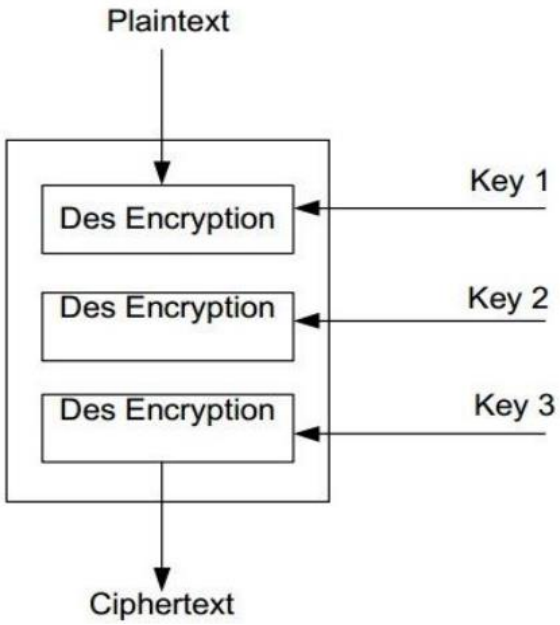
(1) Loại thứ nhất tác động trên bản rõ theo từng nhóm bits. Từng nhóm bits này được gọi với một cái tên khác là khối (Block) và thuật toán được áp dụng gọi là Block Cipher. Theo đó, từng khối dữ liệu trong văn bản ban đầu được thay thế bằng một 11 khối dữ liệu khác có cùng độ dài. Đối với các thuật toán ngày nay thì kích thước chung của một Block là 64 bits.

(2) Loại thứ hai tác động lên bản rõ theo từng bit một. Các thuật toán áp dụng được gọi là Stream Cipher. Theo đó, dữ liệu của văn bản được mã hoá từng bit một. Các thuật toán mã hoá dòng này có tốc độ nhanh hơn các thuật toán mã hoá khối và nó thường được áp dụng khi lượng dữ liệu cần mã hoá chưa biết trước. Một số thuật toán nổi tiếng trong mã hoá đối xứng là: DES, Triple DES(3DES), RC4, AES…

+ DES (Data Encryption Standard). Với DES, bản rõ (Plaintext) được mã hoá theo từng khối 64 bits và sử dụng một khoá là 64 bits, nhọng thực tế thì chỉ có 56 bits là thực sự được dùng để tạo khoá, 8 bits còn lại dùng để

kiểm tra tính chẵn, lẻ. DES là một thuật toán được sử dụng rộng rãi nhất trên thế giới. Khoảng 30 năm đầu tiên kể từ ngày công bố, DES tỏ ra rất an toàn. Tuy nhiên, cho tới thời điểm hiện tại DES không còn được đánh giá cao do kích thước của khoá quá nhỏ (56 bits), và dễ dàng bị phá vỡ bởi tốc độ xử lý của siêu máy tính.

+ Triple DES (3DES): 3DES cải thiện độ mạnh của DES bằng việc sử dụng một quá trình mã hoá và giải mã sử dụng 3 khoá. Toàn bộ chiều dài khóa là 192 bit gồm 3 khóa DES là K1, K1, K3. Khối 64-bits của bản rõ đầu tiên sẽ được mã hoá sử dụng khoá thứ nhất. Sau đó, dữ liệu bị mã hóa được giải mã bằng việc sử dụng một khoá thứ hai. Cuối cùng, sử dụng khoá thứ ba và kết quả của quá trình mã hoá trên để mã hoá.



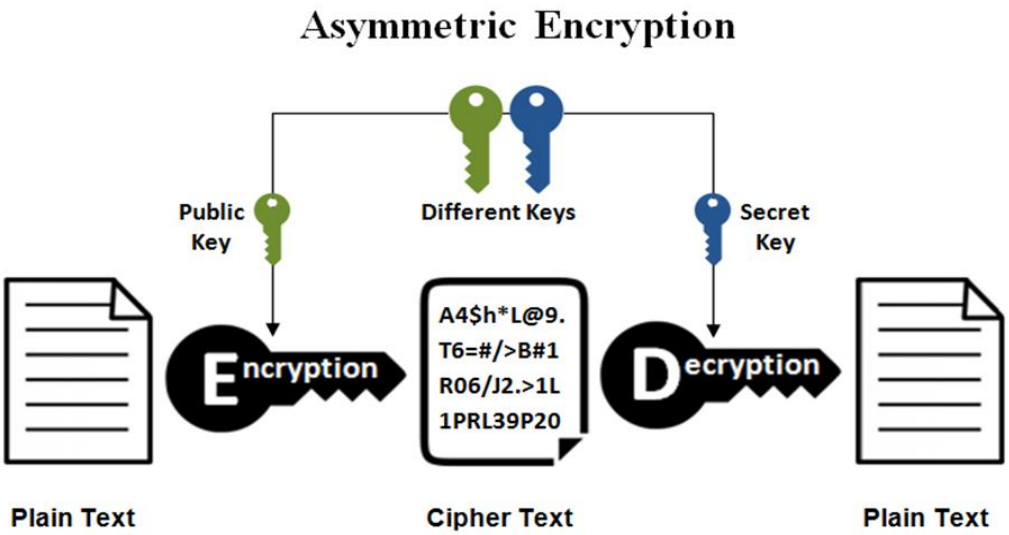
Hình 1.3: Mô tả quy trình hoạt động của thuật toán 3DES

1.2.2. Mã hóa bất đối xứng

- Mã hoá bất đối xứng còn được gọi với một cái tên khác là mã hoá khoá công khai (Public Key Cryptography), được thiết kế sao cho khoá sử dụng trong quá trình mã hoá khác biệt với khoá được sử dụng trong quá trình giải mã. Khoá sử dụng trong quá trình giải mã không thể được tính toán hay luận ra được từ khoá được dùng để mã hoá và ngược lại, tức là hai khoá này có quan hệ với nhau về mặt toán học nhưng không thể suy diễn được ra nhau.
- Một người bất kỳ có thể dùng khoá này để mã hoá dữ liệu nhưng chỉ duy nhất người mà có khoá giải mã tương ứng mới có thể đọc được dữ liệu mà thôi. Do đó trong thuật toán này có 2 loại khoá:

+ Khoá để mã hoá được gọi là Public Key
+ Khoá để giải mã được gọi là Private Key.

– Quá trình truyền và sử dụng mã hoá khoá công khai:



Hình 1.4: Mô tả quá trình truyền và sử dụng khóa công khai

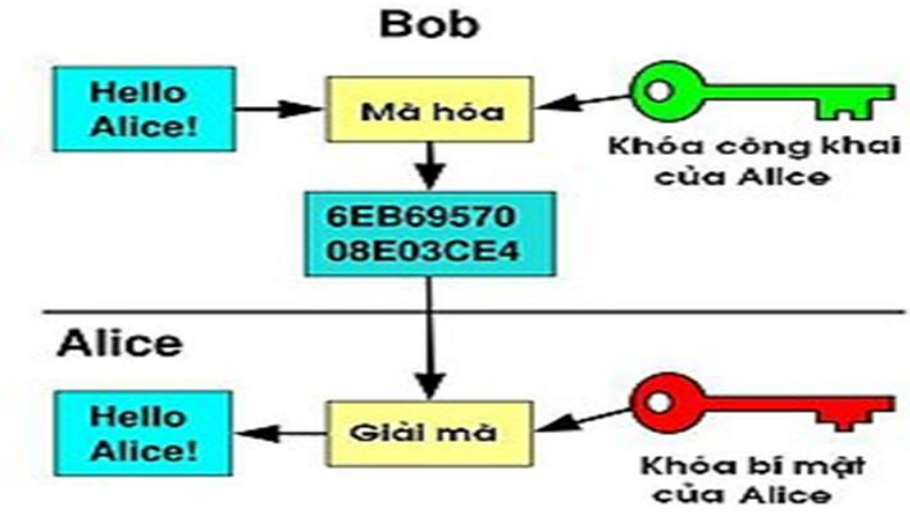
+ Bên gửi yêu cầu cung cấp hoặc tự tìm khoá công khai của bên nhận trên một server chịu trách nhiệm quản lý khoá.
+ Sau đó hai bên thống nhất thuật toán dùng để mã hoá dữ liệu, bên gửi sử dụng khoá công khai của bên nhận cùng với thuật toán đã thống nhất để mã hoá thông tin được gửi đi.
+ Khi nhận được thông tin đã mã hoá, bên nhận sử dụng khoá bí mật của mình để giải mã và lấy ra thông tin ban đầu.

Với sự ra đời của mã hóa công khai thì khoá được quản lý một cách linh hoạt và hiệu quả hơn. Người sử dụng chỉ cần bảo vệ Private key. Tuy nhiên tốc độ mã hóa của nó chậm hơn rất nhiều so với mã hoá đối xứng. Do đó, người ta thường kết hợp hai hệ thống mã hoá khoá đối xứng và công khai lại với nhau và được gọi là Hybrid Cryptosystems. Một số thuật toán mã hóa công khai nổi tiếng: Diffie-Hellman, RSA,...

1.3. Giới thiệu về mã hóa công khai

1.3.1. Mã hóa công khai

- Việc giữ bí mật khóa mật đồng nghĩa với việc giữ mật thông tin. Nên việc trao đổi khóa chỉ diễn ra trên kênh mật thì mới đảm bảo được, thế nhưng việc trao đổi này cũng không dễ để đảm bảo độ an toàn cao. Từ đây hình thành nên ý tưởng của mật mã công khai. Tức là không cần phải trao đổi khóa mật qua kênh nữa.
- Ý tưởng của hệ mật công khai được Diffie và Hellman đưa ra năm 1976. Còn việc thực hiện hệ mật công khai thì do Rivest, Shamir và Adleman đưa ra đầu tiên năm 1977, họ đề xuất một hệ mật RSA nổi tiếng. Và kể từ đó có một số hệ mật khác được công bố, độ mật của chúng dựa trên các bài toán khác nhau, như dựa trên độ khó của bài toán phân tích thành nhân tử như hệ mật RSA, dựa vào độ khó logarithm rời rạc như hệ mật ElGamal, hay dựa trên đường cong Elliptic. Chúng ta sẽ tìm hiểu cụ thể các hệ mật này trong các phần sau. Nhưng trước tiên chúng ta đi tìm hiểu sơ đồ và nguyên tắc mã và giải mã của hệ mật công khai.



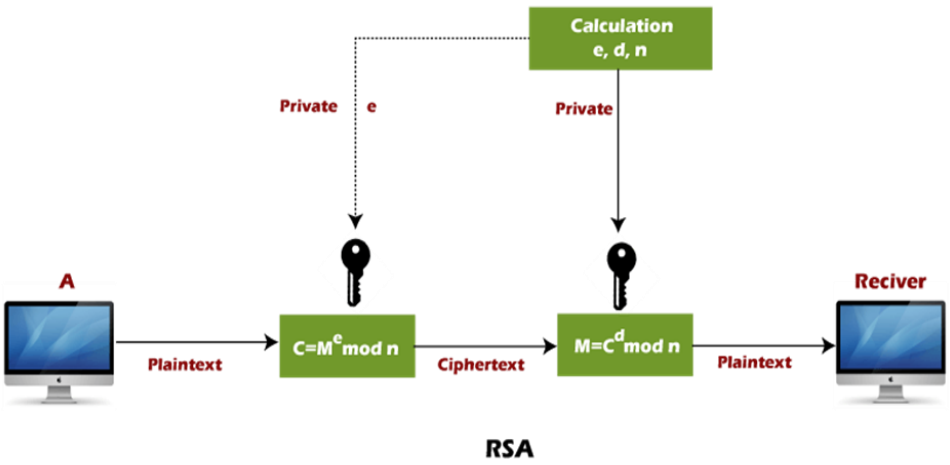
Hình 1.5: Mô tả quá trình gửi tin nhắn từ Bob đến Alice

- Hệ mã công khai sử dụng hai khóa có quan hệ toán học với nhau, tức là một khóa này được hình thành từ khóa kia: Người muốn nhận bản mã (Alice) tạo ra một khóa mật (private key) và từ khóa mật tính ra khóa công khai (public key) với một thủ tục không phức tạp, còn việc tìm khóa mật khi biết khóa công khai là bài toán khó giải được. Khóa công khai sẽ đưa đến cho người gởi bản tin (Bob) qua kênh công cộng. Và bản tin được Bob mã hóa bằng khóa công cộng. Bản mã truyền đến Alice, và nó được giải mã bằng khóa mật.

1.3.2. Hệ mật RSA

- RSA (Rivest-Shamir-Adleman) là một thuật toán mã hóa công khai (public-key cryptography) được phát triển bởi Ron Rivest, Adi Shamir và Leonard Adleman vào năm 1977. Thuật toán RSA nổi tiếng và phổ biến trong lĩnh vực bảo mật thông tin.
- Thuật toán RSA dựa trên việc sử dụng một cặp khóa, bao gồm khóa công khai (public key) và khóa bí mật (private key). Khóa công khai được chia sẻ với mọi người trong khi khóa bí mật chỉ được giữ bí mật bởi người sở hữu. Thuật toán RSA dựa trên tính khó giải ra hai số nguyên tố lớn từ tích của chúng.
- Quá trình mã hóa thông điệp sử dụng khóa công khai và quá trình giải mã sử dụng khóa bí mật. Để mã hóa, thông điệp ban đầu sẽ được chuyển đổi thành một số nguyên và sau đó được mã hóa bằng cách sử dụng khóa công khai. Sau đó, thông điệp mã hóa có thể được gửi đi một cách an toàn. Để giải mã, người nhận sẽ sử dụng khóa bí mật để giải mã thông điệp mã hóa và khôi phục lại thông điệp ban đầu.
- RSA được sử dụng rộng rãi trong các ứng dụng bảo mật, như mã hóa giao tiếp qua mạng, chữ ký số, chứng thực và xác nhận danh tính. Thuật toán RSA cũng được sử dụng trong các hệ thống mật mã hàng đầu như TLS/SSL để bảo vệ việc trao đổi dữ liệu giữa máy chủ và trình duyệt web.
- Tuy thuật toán RSA mạnh về mặt bảo mật, nhưng nó cũng có khái niệm về kích thước khóa để đảm bảo tính bảo mật. Hiện nay, kích thước khóa RSA thường được giới thiệu từ 2048 bit trở lên.

1.4. Quy trình hoạt động



Hình 1.6: Mô tả quá trình hoạt động của thuật toán RSA

Với thuật toán RSA, đầu tiên cần tạo ra cho mình cặp khóa gồm khóa công khai và khóa bí mật theo các bước sau:

- Chọn 2 số nguyên tố lớn khác nhau p, q thỏa mãn điều kiện : $|p| \approx |q|$
- Tính tích của nó: $n = p.q$
- Tính giá trị hàm Phi Euler của n: $\varphi(n) = (p-1)(q-1)$
- Chọn số nguyên d, sao cho $d < \varphi(n)$ và $\gcd(d, \varphi(n)) = 1$
- Tính giá trị e thỏa mãn điều kiện: $ed = 1 \ mod(\varphi(n))$
- Khóa công khai (e, n) bao gồm:
  + n, môđun.
  + e, số mũ công khai.
- Khóa bí mật (d, n) bao gồm:
  + n, môđun, xuất hiện cả trong khóa công khai và khóa bí mật.
  + d, số mũ bí mật.
- Mã hóa:
  + Chúng ta có bản rõ *M* chúng ta cần chuyển nó thành một số tự nhiên m < n theo một hàm có thể đảo ngược (từ có thể xác định lại *M*) được thỏa thuận trước.
  + c là bản mã hóa của theo công thức: $c = m^e \ mod \ n$.
- Giải mã: Ở phía người nhận, họ sẽ giải mã từ c để lấy được m như sau:

$$m = c^d \ mod \ n$$

Ví dụ: Mã hóa bất đối xứng. Cho văn bản gốc: P = <TEN>. Thực hiện mã hoá từng ký tự trong P theo: hệ mật RSA với p= 3, q=7. Tìm khóa bí mật, khóa công khai và bản mã hóa cho văn bản P.

Giải

- **Tính n**: n = p.q = 3.7 = 21
- **Tính Φ:** Φ(n) = (p-1)*(q-1) = (3-1)*(7-1) = 12
- **Chọn e** sao cho 1 < e < Φ(n) và và UCLN(e, φ(n)) = 1.
  Ta chọn e = 5 => Khóa công khai: (e, n) = (5, 21)
- **Tìm d** sao cho (d * e) mod φ(n) = 1.
  Tương đương với d * e = k * φ(n) + 1.
  - Với e = 5 và φ(n) = 12, ta có 5d = 12k + 1
  - Chọn k = 2, ta được 5d = 25 => d = 5

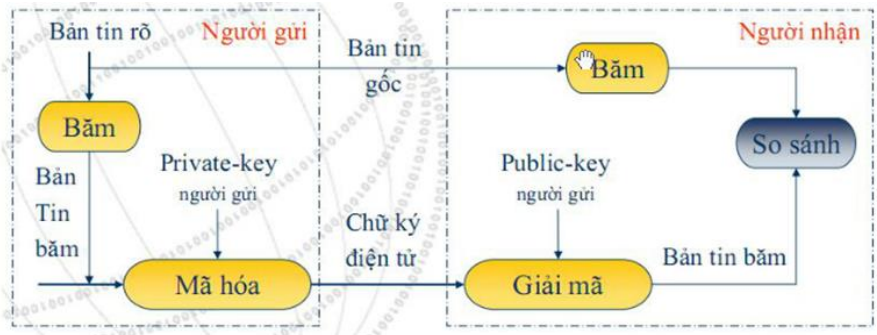
- **Khóa công khai:** (e, n) = (5, 21)
- **Khóa bí mật:** (d, n) = (5, 21)
- **Mã hóa:**
  + Mã hóa ký tự 'T' = 20: C = $M^e \ mod \ n = 20^5 \ mod \ 21 = 16$
  + Mã hóa ký tự 'E' = 5: C = $M^e \ mod \ n = \ 5^5 \ mod \ 21 = 17$
  + Mã hóa ký tự 'N' =14:C =$M^e \ mod \ n = \ 14^5 \ mod \ 21 = \ 14$
  + Vậy bản mã là (16, 17, 14).

1.5. Ứng dụng thực tế của hệ mật RSA

1.5.1. Chữ ký điện tử (Digital Signature)

- Chữ ký điện tử là một loại chữ ký được tạo bằng phương tiện điện tử, gắn liền với thông điệp dữ liệu, có giá trị pháp lý tương đương chữ ký tay, giúp xác nhận danh tính người ký và sự đồng ý của họ đối với nội dung.
- Quá trình tạo chữ ký bắt đầu bằng việc băm văn bản gốc bằng thuật toán băm một chiều để tạo giá trị băm (hash value). Giá trị này được mã hóa bằng khóa riêng (private key) của người ký để tạo ra chữ ký số và đính kèm vào văn bản. Khi nhận, dùng public key của người ký để giải mã chữ ký số, lấy giá trị băm và so sánh với giá trị băm tự tính từ văn bản nhận được. Nếu hai giá trị băm trùng khớp, văn bản được đảm bảo tính toàn vẹn và xác thực người ký.

  Mô hình chung của chữ ký điện tử:



Hình 1.7: Quy trình thực hiện chữ ký điện tử :

Các bước mã hóa :

1. Dùng thuật toán băm (như MD5, SHA-1, SHA-256, SHA-512) để tạo ra giá trị băm của message. Ví dụ, SHA-256 tạo ra giá trị băm 256 bit.

2. Mã hóa giá trị băm: Sử dụng khóa riêng (private key) của người gửi để mã hóa giá trị băm. Thường dùng thuật toán RSA để thực hiện mã hóa. Kết quả của bước này là chữ ký số, đại diện duy nhất cho thông điệp

3. Đính kèm: Gắn chữ ký số vào thông điệp gốc. Quá trình này gọi là "ký nhận" thông điệp. Chữ ký đảm bảo rằng bất kỳ thay đổi nào với thông điệp đều có thể bị phát hiện và xác nhận rằng thông điệp thực sự được gửi bởi người ký.

Các bước kiểm tra :

1. Giải mã chữ ký số: Dùng public key của người gửi (khóa được thông báo đến mọi người) để giải mã chữ ký số của message, thu được giá trị băm ban đầu.
2. Băm thông điệp nhận được: Sử dụng cùng thuật toán băm đã dùng ở bước 1 của quá trình mã hóa để tính giá trị băm của message nhận được.
3. So sánh: Nếu trùng nhau, ta kết luận message này không bị thay đổi trong quá trình truyền và message này là của người gửi.

**Chú ý:* Mỗi cá nhân khi tham gia vào hệ thống chữ ký điện tử cần phải được cung cấp một bộ khóa (Public key, Private key) dùng để định danh cá nhân đó bởi một tổ chức cơ quan có thẩm quyền và được công nhận trong phạm vi sử dụng.

1.5.2. SSL/TLS (Secure Socket Layer/Transport Layer Security)

- SSL (Secure Sockets Layer), nay được thay thế bởi phiên bản mới hơn và an toàn hơn là TLS (Transport Layer Security), là một giao thức mật mã được sử dụng rộng rãi để bảo mật giao tiếp giữa máy chủ web và trình duyệt web (hoặc giữa hai ứng dụng bất kỳ). SSL/TLS đảm bảo tính bảo mật, toàn vẹn và xác thực dữ liệu được truyền tải.
- **RSA đóng vai trò quan trọng trong SSL/TLS, cụ thể:**
  - + Trao đổi khóa phiên:
    - Trong giai đoạn bắt tay (handshake), máy khách tạo khóa phiên ngẫu nhiên và mã hóa bằng khóa công khai của máy chủ.
    - Máy chủ dùng khóa riêng để giải mã, lấy khóa phiên.
    - Khóa phiên (đối xứng) được sử dụng để mã hóa và giải mã dữ liệu trong phiên, giúp tăng hiệu suất.
  - + Xác thực máy chủ:
    - Máy chủ sử dụng chứng chỉ số (chứa khóa công khai) do cơ quan chứng thực (CA) cấp để chứng minh danh tính.
    - RSA được dùng để ký và xác minh chứng chỉ số, đảm bảo máy khách kết nối với đúng máy chủ.
- RSA được sử dụng trong việc trao đổi khóa phiên và xác thực máy chủ, đảm bảo tính bảo mật và tin cậy của kết nối là một phần quan trọng trong việc thiết lập kết nối an toàn giữa máy khách và máy chủ.

Chương 2: Phân tích thiết kế

2.1. Công nghệ sử dụng

2.1.1. Ngôn ngữ lập trình C#

- C# là ngôn ngữ lập trình hướng đối tượng do Microsoft phát triển (ra mắt năm 2000). Ban đầu chạy trên .NET Framework, nay hỗ trợ nhiều nền tảng như .NET Core và .NET 5/6/7, phù hợp phát triển ứng dụng desktop, web, di động, và game.
- *Tính năng nổi bật của C#:*

  + Hướng đối tượng (OOP): Giúp tổ chức mã nguồn dễ dàng và hiệu quả.
  + Type-safe: Ngăn chặn lỗi kiểu dữ liệu, giảm các lỗi tại thời điểm chạy.
  + Garbage Collection (GC): Quản lý bộ nhớ tự động giúp giảm thiểu các lỗi bộ nhớ và nguy cơ tràn bộ nhớ.
  + LINQ: Cho phép truy vấn dữ liệu từ các nguồn như cơ sở dữ liệu hay XML trực tiếp trong mã nguồn.
  + Cross-platform: C# hỗ trợ nhiều nền tảng, giúp các ứng dụng có thể chạy trên nhiều hệ điều hành khác nhau.

Bảng 1.2: Ưu và nhược điểm của C#

Ưu điểm	Hạn chế
Có cú pháp rõ ràng và dễ học.	Phụ thuộc vào hệ sinh thái Microsoft
Hiệu suất cao: Kết hợp giữa tính linh hoạt của lập trình hướng đối tượng và hiệu suất của ngôn ngữ biên dịch.	Hiệu suất thấp hơn C++: Việc sử dụng Garbage Collection có thể ảnh hưởng đến hiệu suất trong một số trường hợp.
Hỗ trợ mạnh mẽ từ Microsoft, có cộng đồng rộng lớn	Yêu cầu tài nguyên lớn
Bảo mật cao: C# tích hợp các cơ chế bảo mật giúp phát triển ứng dụng an toàn hơn.	Cần nhiều thời gian để thành thạo các tính năng nâng cao với C#

- *Lý do chọn WPF cho phát triển ứng dụng:*
  + Tương thích C# và .NET Framework: Hỗ trợ triển khai mã hóa bảo mật.
  + Giao diện phong phú: Tạo giao diện động, mượt mà, dễ tương tác.
  + Hỗ trợ bất đồng bộ: Giảm độ trễ khi xử lý dữ liệu.
  + Quản lý bộ nhớ tự động: Hạn chế lỗi bộ nhớ.
  + Bảo mật cao: Triển khai dễ dàng mã hóa End-to-End.
  + Đa nền tảng: Chạy được trên nhiều hệ điều hành nhờ .NET Core.

2.1.2. Framework WPF

- WPF (Windows Presentation Foundation) là framework của Microsoft dùng để phát triển ứng dụng desktop cho nền tảng Windows, nổi bật với khả năng xây dựng giao diện người dùng phong phú và hỗ trợ các tính năng đồ họa mạnh mẽ. Là công cụ lý tưởng cho các ứng dụng yêu cầu giao diện đẹp mắt và bảo mật cao, đặc biệt trong việc xây dựng các ứng dụng desktop, web, di động và game.
- *Tính năng nổi bật của WPF:*
  + Giao diện phong phú: Hỗ trợ các hiệu ứng hình ảnh, chuyển động, và đồ họa vector, mang đến trải nghiệm người dùng trực quan và hấp dẫn.
  + XAML: Ngôn ngữ khai báo giao diện giúp tách biệt rõ ràng giữa giao diện và logic, đơn giản hóa việc bảo trì và tái sử dụng mã nguồn.
  + Hỗ trợ mạnh mẽ việc liên kết dữ liệu, tự động đồng bộ hóa giữa giao diện và dữ liệu mà không cần viết nhiều mã lệnh.
  + Tích hợp C# và .NET Framework: WPF làm việc hiệu quả với C#, hỗ trợ triển khai các tính năng bảo mật mã hóa RSA một cách dễ dàng.
  + Quản lý bộ nhớ tự động
  + Hỗ trợ bất đồng bộ: Cho phép thực thi các tác vụ nặng như mã hóa, giải mã mà không làm ảnh hưởng đến giao diện người dùng.

Bảng 1.3: Ưu điểm và hạn chế của WPF

Ưu điểm	Hạn chế
Hiệu suất cao nhờ GPU rendering.	Độ phức tạp cao, khó học với người mới.
Linh hoạt, hỗ trợ thiết kế giao diện hiện đại.	Yêu cầu phần cứng cao, khó chạy trên máy cũ.
Tích hợp tốt với hệ sinh thái .NET.	Chỉ hỗ trợ Windows, không đa nền tảng.

Hỗ trợ MVVM, dễ tổ chức và bảo trì mã.	Ít phổ biến, Microsoft ưu tiên công nghệ mới.
Data Binding mạnh mẽ, tùy chỉnh cao.	Công cụ phát triển và sử dụng phức tạp.

- *Lý do chọn WPF cho ứng dụng:*
  + Tính bảo mật cao: Dễ dàng tích hợp mã hóa RSA để bảo vệ dữ liệu tin nhắn, chỉ người gửi và người nhận mới có thể đọc được.
  + Giao diện người dùng linh hoạt: Cho phép xây dựng giao diện phức tạp và dễ sử dụng, phù hợp với ứng dụng chat bảo mật.
  + Khả năng xử lý bất đồng bộ: Giảm độ trễ khi mã hóa và giải mã, giúp trải nghiệm người dùng mượt mà.
  + Tính mở rộng và linh hoạt: Hỗ trợ các nền tảng khác ngoài Windows khi kết hợp với .NET Core.
  + Môi trường phát triển mạnh mẽ: Công cụ Visual Studio giúp tăng hiệu quả phát triển và giảm thời gian triển khai.

2.1.3. Hàm băm sử dụng Argon2

- Argon2 là thuật toán băm mật khẩu nổi bật, chiến thắng cuộc thi "Password Hashing Competition" năm 2015. Được đánh giá là một trong những thuật toán tốt nhất hiện nay, Argon2 được thiết kế để chống lại các cuộc tấn công brute-force và đảm bảo bảo mật cao.
- *Các biến thể của Argon2:*
  + **Argon2i**: Tối ưu hóa chống tấn công kênh bên (side-channel attacks), truy cập dữ liệu tuần tự.
  + **Argon2d**: Chống lại brute-force sử dụng GPU, tập trung vào xử lý phụ thuộc bộ nhớ.
  + **Argon2id**: Kết hợp ưu điểm của Argon2i và Argon2d, đảm bảo bảo mật toàn diện.
- Tính năng nổi bật của Argon2:
  + Khả năng điều chỉnh tài nguyên: Tùy chỉnh tham số (bộ nhớ, thời gian xử lý, luồng) để cân bằng hiệu suất và bảo mật.
  + Chống brute-force: Sử dụng bộ nhớ lớn và tính toán phức tạp, làm tăng chi phí tấn công.
  + Hiệu suất tốt: Hoạt động hiệu quả trên CPU và GPU, tối ưu hóa cho nhiều nền tảng.

+ Bảo mật cao: Phù hợp lưu trữ mật khẩu và mã hóa dữ liệu.

Bảng 1.4: Ưu và nhược điểm của Argon2

Ưu điểm	Hạn chế
Bảo mật cao, chống brute force và GPU.	Cần cấu hình tham số phức tạp.
Linh hoạt, tùy chỉnh bộ nhớ và thời gian.	Hiệu suất phụ thuộc vào phần cứng.
Được NIST khuyến nghị sử dụng.	Không tương thích rộng trên hệ thống cũ.
Chống tấn công side-channel.	Yêu cầu hiểu biết sâu khi triển khai
Hỗ trợ nhiều phiên bản: Argon2i, Argon2d.	Hiệu quả thấp trên thiết bị yếu.

– *Lý do chọn Argon2 cho ứng dụng bảo mật:*
  + Tính an toàn cao: Bảo vệ mạnh mẽ trước brute-force, phù hợp ứng dụng nhạy cảm như chat bảo mật.
  + Khả năng tùy chỉnh linh hoạt: Đáp ứng nhu cầu sử dụng khác nhau.
  + Hiệu quả thực tế: Hoạt động tốt trên cả hệ thống mạnh và trung bình, phù hợp với các cấp độ bảo mật.

2.1.4. SQL server

– SQL Server là hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) được phát triển bởi Microsoft. Đây là công cụ mạnh mẽ dành cho việc lưu trữ, quản lý và xử lý dữ liệu, phù hợp với nhiều loại ứng dụng từ quy mô nhỏ đến doanh nghiệp lớn. *Tính năng nổi bật của SQL Server:*

  + Quản lý dữ liệu hiệu quả: Hỗ trợ xử lý khối lượng lớn dữ liệu với bảng, chỉ mục và phân vùng, đảm bảo hiệu suất cao.

  + T-SQL (Transact-SQL): Ngôn ngữ truy vấn mạnh mẽ để thao tác, quản lý và xử lý dữ liệu linh hoạt.

  + Bảo mật nâng cao: Tích hợp mã hóa dữ liệu, chứng thực người dùng, kiểm soát truy cập để bảo vệ dữ liệu nhạy cảm.

+ Hiệu suất tối ưu: Sử dụng chỉ mục, bộ nhớ đệm (caching) và tối ưu hóa truy vấn để tăng tốc độ xử lý.

+ Tích hợp và phân tích: Hỗ trợ các công cụ phân tích chuyên sâu như SSAS và SSRS để tạo báo cáo chi tiết.

+ Khả năng mở rộng và sẵn sàng cao: Always On Availability Groups đảm bảo mở rộng linh hoạt và tính sẵn sàng của hệ thống.

Bảng 1.5: Ưu và nhược điểm của SQL Server

Ưu điểm	Hạn chế
Hiệu năng cao, xử lý dữ liệu lớn hiệu quả.	Chi phí bản quyền cao cho các phiên bản đầy đủ.
Tích hợp tốt với hệ sinh thái Microsoft.	Hỗ trợ chính thức chủ yếu trên Windows.
Bảo mật mạnh mẽ, hỗ trợ mã hóa và xác thực tốt.	Yêu cầu phần cứng mạnh để tối ưu hiệu suất.
Hỗ trợ nhiều công cụ quản trị và phát triển.	Cấu hình phức tạp, đòi hỏi kỹ năng quản trị cao.
Hỗ trợ tính năng sao lưu, phục hồi, HA/DR tốt.	Kém tối ưu trên các hệ thống mã nguồn mở.

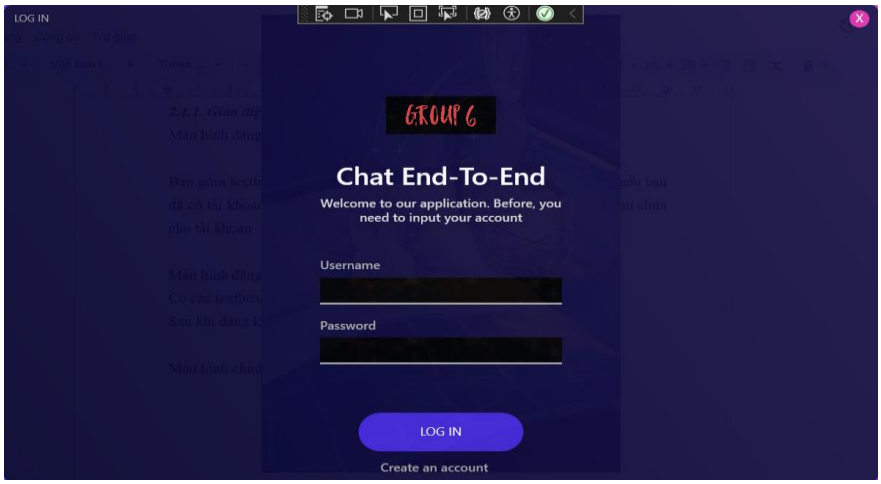
Lý do chọn SQL Server cho ứng dụng chat bảo mật:

+ Bảo mật dữ liệu cao
+ Hiệu suất và độ tin cậy cao: Truy vấn nhanh, hỗ trợ phục hồi dữ liệu, đảm bảo hệ thống hoạt động ổn định.
+ Tích hợp tốt với C# và WPF
+ Quản lý dữ liệu lớn: Xử lý hàng triệu tin nhắn với độ trễ thấp, phù hợp cho ứng dụng có nhiều người dùng.
+ Khả năng mở rộng: Hỗ trợ tăng quy mô hệ thống linh hoạt, đáp ứng nhu cầu phát triển ứng dụng trong tương lai.

2.2. Giới thiệu sản phẩm

2.2.1. Giao diện ứng dụng

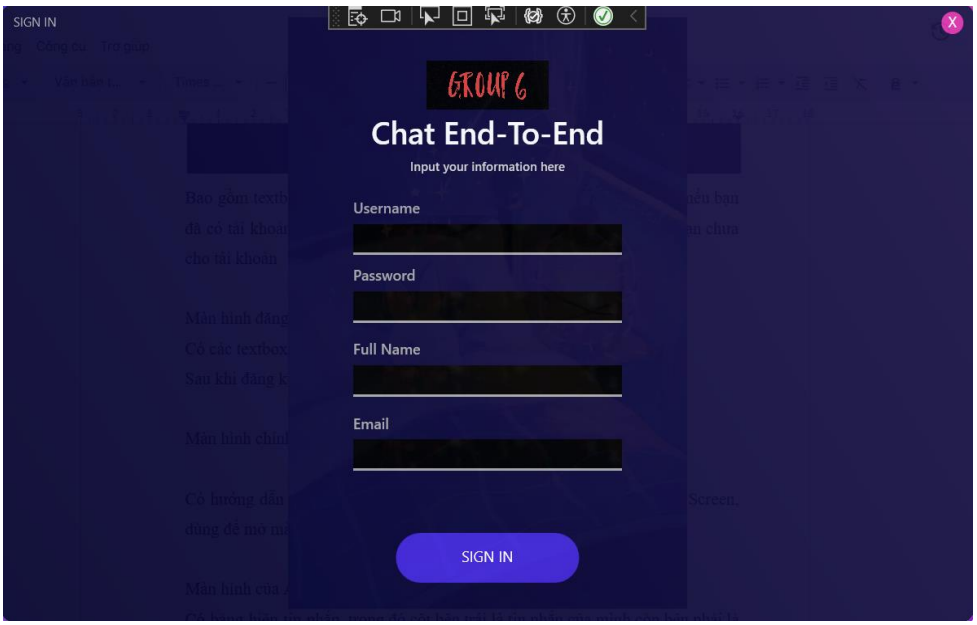
2.2.1.1. Màn hình đăng nhập:



Hình 2.1: Giao diện đăng nhập

Bao gồm textbox nhập gmail (kiểu string) và mật khẩu (kiểu string), cùng với đó là nút đăng nhập nếu bạn đã có tài khoản, ngược lại có nút "Create an account" để đăng ký nếu bạn chưa có tài khoản

2.2.1.2. Màn hình đăng ký:



Hình 2.2: Giao diện đăng ký tài khoản

Có các textbox nhập tên tài khoản, tên người dùng, gmail, mật khẩu (tất cả đều thuộc kiểu string). Tại đây người dùng phải nhập đúng kiểu dữ liệu và không được để trống nếu không sẽ có thông báo lỗi.

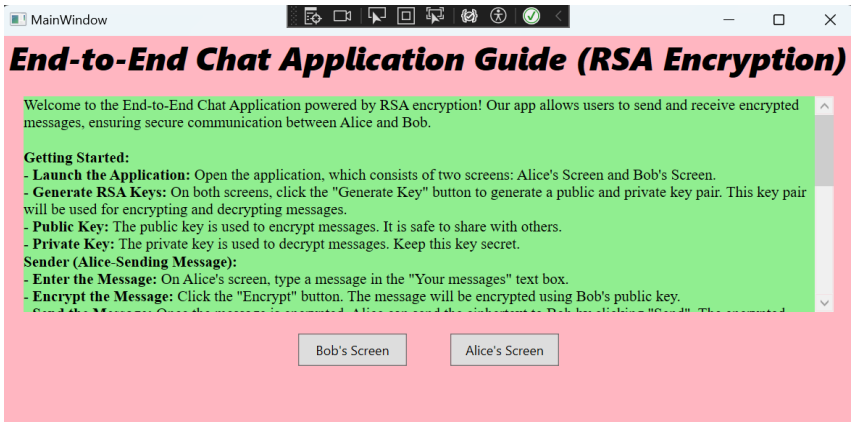
Người dùng cần thực hiện điền đầy đủ thông tin form yêu cầu và cần lưu ý:

+ Username, Fullname, Password, Email không được để trống.
+ Email phải đầy đủ yêu cầu, phần trước dấu @ phải có ít nhất 1 ký tự, cho phép chữ cái (cả hoa và thường) và chữ số. Cho phép các ký đặc biệt: . , _ , % , +, -.
+ Phải có một dấu @.
+ Phần giữa @ và dấu chấm cuối cùng phải chứa ít nhất một ký tự là chữ cái, số, dấu chấm, hoặc dấu gạch ngang.
+ Phải có một dấu chấm cuối cùng.
+ Phần đuôi phải chứa ít nhất hai ký tự chữ cái.

Sau khi đăng ký thành công thì sẽ có account mới được ghi trong CSDL.

– Mật khẩu được băm: Mật khẩu được lưu trữ dưới dạng Argon2id, đảm bảo tính bảo mật. Người dùng không cần lo lắng về việc mật khẩu bị lộ dưới dạng plain text.
– Salt riêng cho mỗi người dùng: Mỗi người dùng có một salt riêng, được sử dụng cùng với Argon2id để băm mật khẩu.
– Xác thực phía server: Việc xác thực đăng nhập được thực hiện bằng cách so sánh mật khẩu băm nhập vào với mật khẩu băm lưu trữ trong cơ sở dữ liệu.

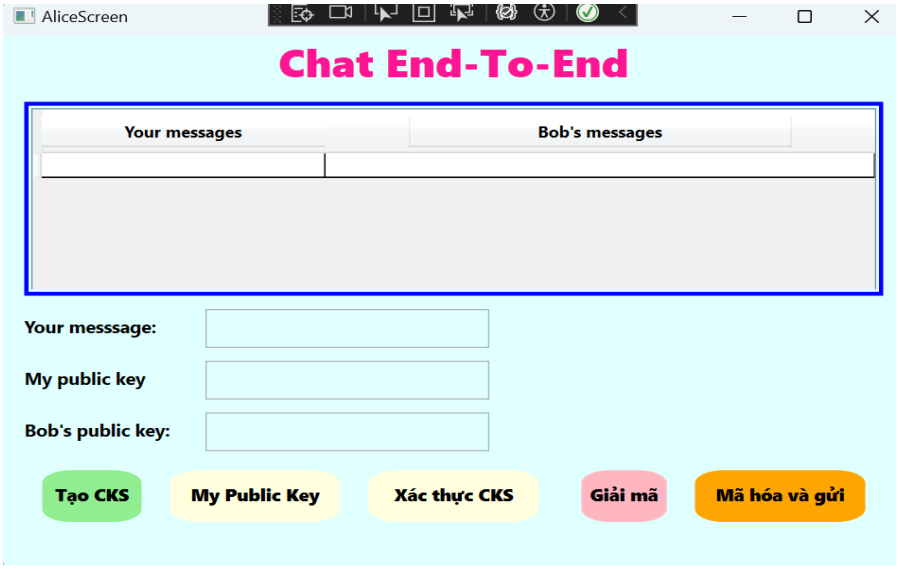
2.2.1.3. Màn hình chính:



Hình 2.3: Giao diện trang chủ

Có hướng dẫn sử dụng phần mềm được viết chi tiết từng bước theo tiếng Anh và hai button Bob's Screen và Alice's Screen, dùng để mở màn hình nhắn tin của Bob và Alice.

2.2.1.4. Màn hình của Alice:



Hình 2.4: Giao diện chat của Alice

Có bảng hiện tin nhắn, trong đó cột bên trái là tin nhắn của mình còn bên phải là tin nhắn chưa mã hóa của Bob.

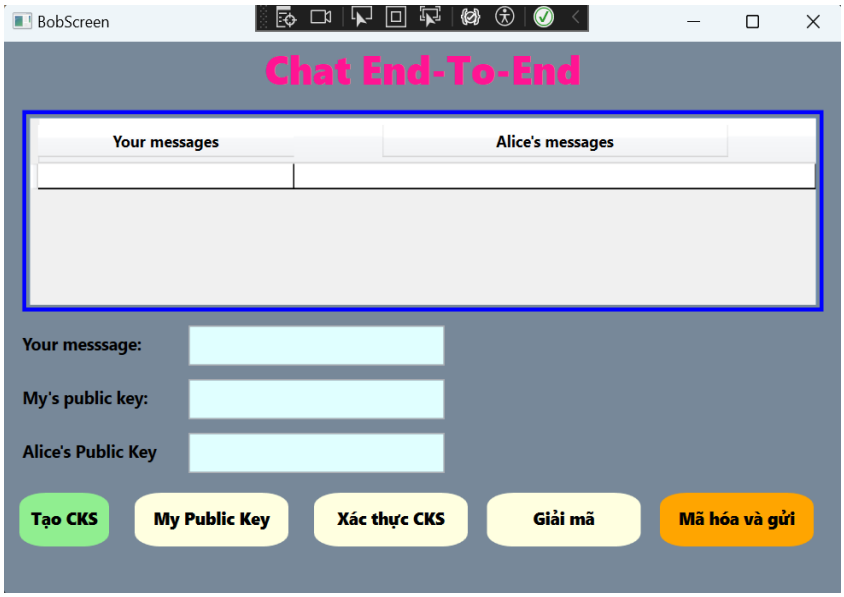
Bao gồm 3 textbox:

- Your message: Nơi bạn nhập tin nhắn để gửi cho Bob.
- My public key: Hiện public key của Alice.
- Bob's public key: Hiển thị public key của Bob.

Và các button:

- "Tạo CKS": Dùng để tạo chữ ký số cho một thông điệp.
- "My public key": Sinh ra hai số nguyên tố ngẫu nhiên, tính khóa RSA mới cho Alice (cả public key lẫn private key), hiển thị khóa công khai của Alice lên giao diện sau đó cập nhật khóa công khai của Alice trên màn hình của Bob.
- "Xác thực CKS": Xác thực tính toàn vẹn và nguồn gốc của thông điệp bằng cách:
    + Giải mã thông điệp đã mã hóa và tính lại giá trị băm (hash1).
    + Giải mã chữ ký số của Bob để lấy giá trị băm ban đầu (hash2).
    + So sánh hai giá trị băm:
        - Nếu hash1 == hash2, xác thực thành công (tính toàn vẹn và nguồn gốc thông điệp được đảm bảo).
        - Nếu khác nhau, xác thực thất bại.
- Giải mã: Giải mã thông điệp được chọn từ danh sách dtglst bằng khóa bí mật của Alice, sau đó hiển thị kết quả giải mã (plaintext) trong một MessageBox.
- Mã hóa và gửi: Mã hóa thông điệp từ textbox bằng khóa công khai của Bob sau đó lưu trữ thông điệp đã mã hóa và hiển thị danh sách thông điệp trên giao diện của Bob.

2.2.1.5. Màn hình của Bob:



Hình 2.5: Giao diện chat của Bob

Tương tự như Alice cũng có bảng hiện tin nhắn, trong đó cột bên trái là tin nhắn của mình còn bên phải là tin nhắn chưa mã hóa của Alice:

Tiếp theo đến 3 textbox:

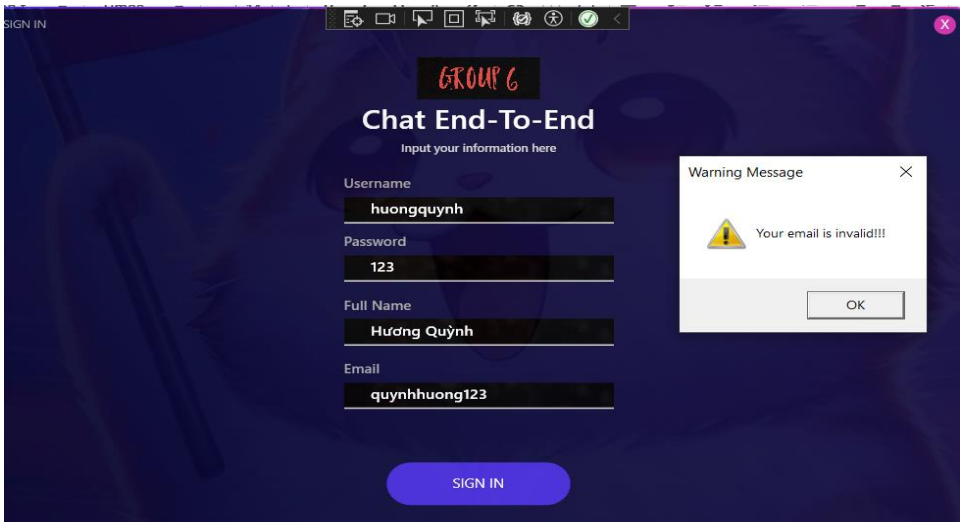
- Your message: Nơi bạn nhập tin nhắn để gửi cho Alice.
- My public key: Hiện public key của Bob.
- Alice's public key: Hiển thị public key của Alice.

Cùng với 5 button với các chức năng tương tự như màn hình của Alice.

2.2.2 Xử lý lỗi và bảo mật

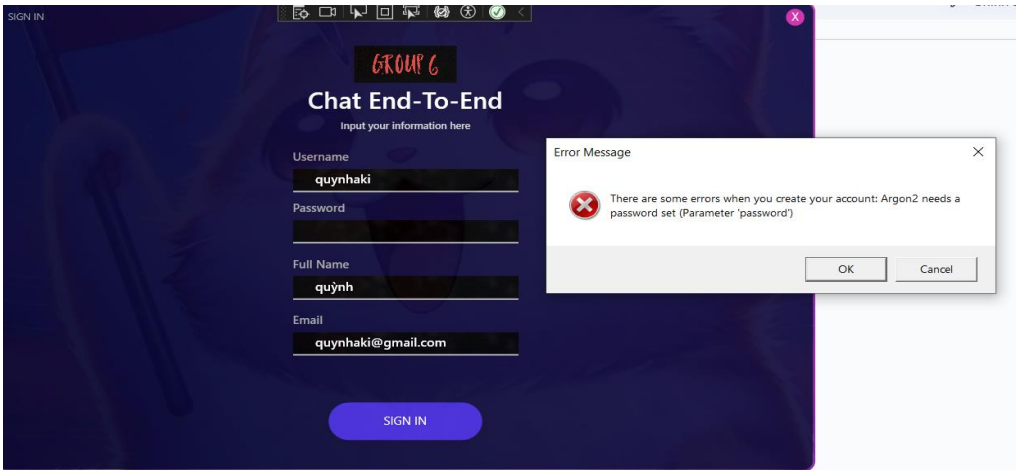
2.2.2.1 Xử lý lỗi:

- Tại màn hình đăng ký:



Hình 2.6: Báo lỗi email sai định dạng

Khi điền sai định dạng email, sẽ hiện ra cảnh báo email của bạn không hợp lệ, yêu cầu người đăng ký phải nhập đúng email



Hình 2.7: Báo lỗi khi không nhập vào các trường

Sau khi đăng ký thành công, thông tin sẽ được lưu trong SQL:

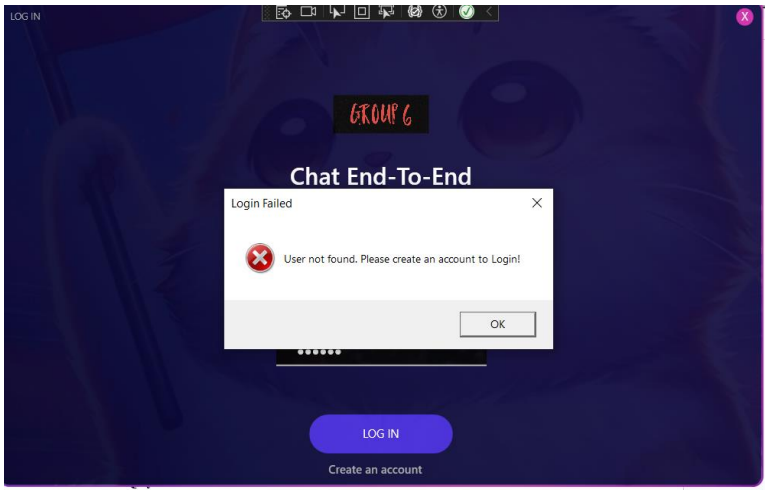
32 %

Results | Messages

	Username	Password	Name	Email	Salt
1		aOd829TesvcGdmxvq57G+af8tXqc06lhM7ChGC6hYjk=	quỳnh	quynhaki@gmail.com	b91e4bc6-a812-4d3d-be57-116f74333cfc
2	huongquynh	ruHmaTr9PAtVLNZPeiCA2jUU+6S0WC2B3cRX9LyGrUk=	Hương Quỳnh	huongquynh@gmail.com	66fbd0bb-89cc-444c-b243-50d99c520cef
3	nguyencuong	IgjYHsQWa15sfRvwgCnlWHYX8C8dITFVo/akKChhwFU=	Nguyễn Huy Cường	cuong04@gmail.com	5294e222-0d07-495d-830b-6d238d055b5e
4	nguyenha256	CfHygj4d0MGxh7U5QFNahmRAtDq7lbuzHRDNg/ArPAE=	Nguyễn Thị Hà	nguyen256@gmail.com	040087ac-6146-4f0b-aeac-ff01c4fd7c99
5	NguyenThuHuong	/E8iMT8X85KXQgju4MtLkcdk4AbZp+UzDwOKUcjAAA4=	Nguyễn Thu Hương	huongthu@gmail.com	952a9137-30dc-408b-af14-443f4e762e92
6	nguyentuan	Dg1Gs02dcFdfS7eJchKXGo4EP6tO2N3FfnOCjq5pou8=	Nguyễn Quốc Tuấn	128932@gmail.com	54569efd-6c75-41e8-ab3c-1f378721609a
7	tientoan123	SVxdk4n7sdXjeWx28TBXGAJ1VOrPQVWslkUZV6fvEK0=	Khuất Tiến Toàn	tientoan12@gmail.com	a88aa589-01e8-4529-a027-14620d9b4148
8	vannha	K6oSaxSVz70NU+1czw2rYXVpKlRIWC9tB2XqK2+xnR0=	Trần Văn Nhã	nha01246892@gmail.com	371e8ba1-6661-44ed-b685-c9a9cc7bee2a

Hình 2.8: Đăng ký thành công tài khoản

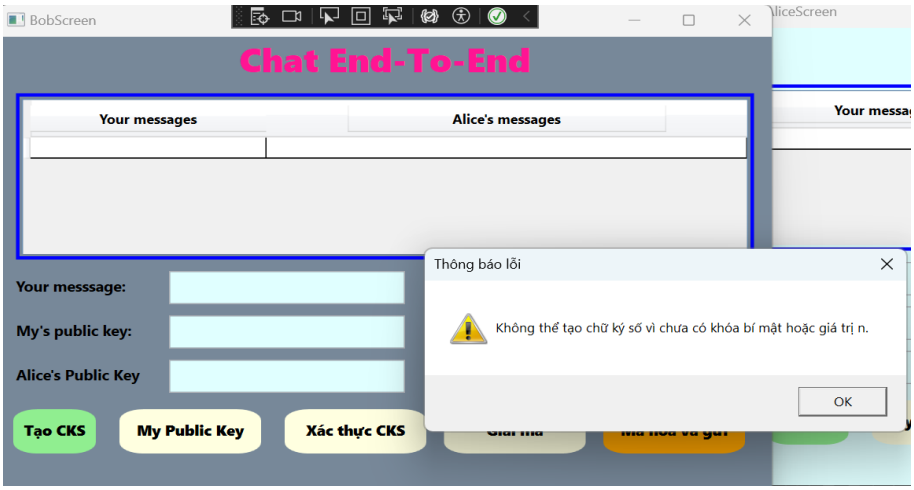
- Tại màn hình đăng nhập:



Hình 2.9: Báo lỗi khi tài khoản không tồn tại

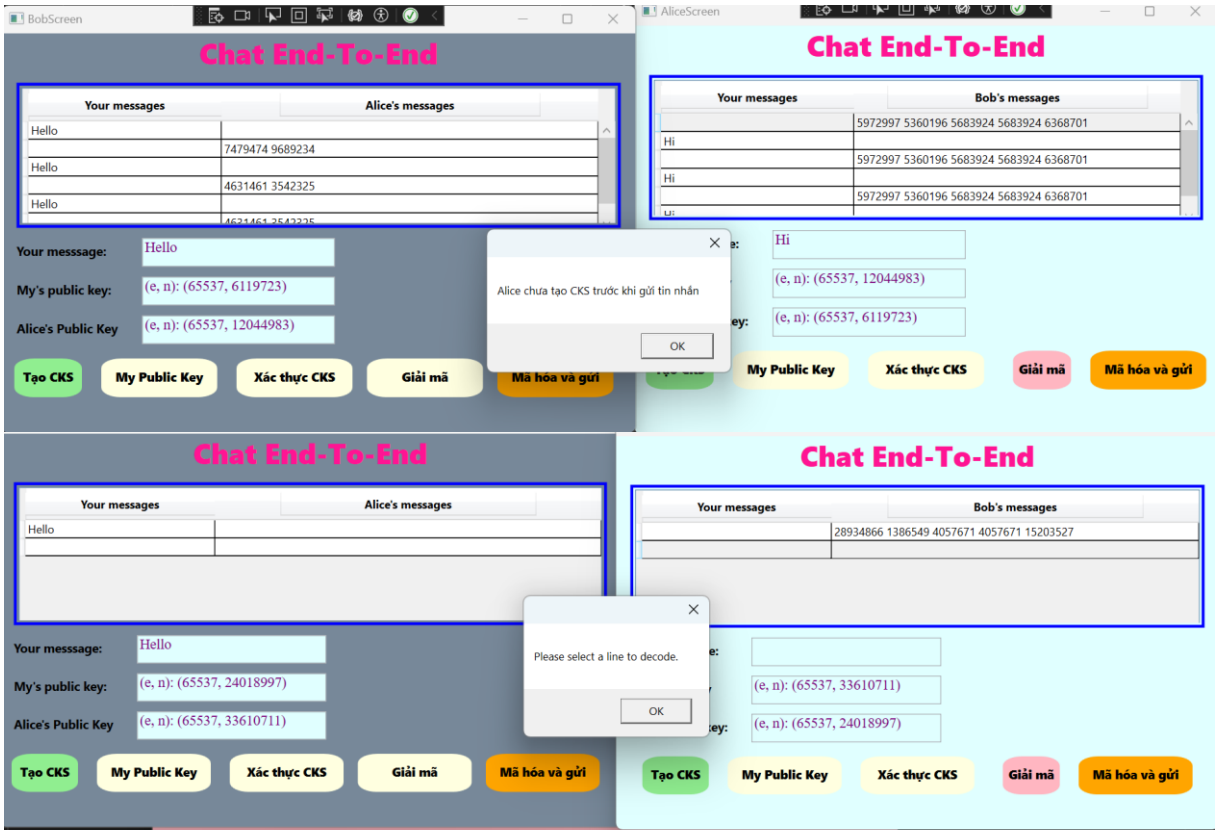
Khi đăng nhập với tài khoản chưa tồn tại, sẽ hiện lỗi không thể tìm thấy người dùng, yêu cầu phải tạo ra tài khoản trước khi đăng nhập.

- Tại màn hình tin nhắn:



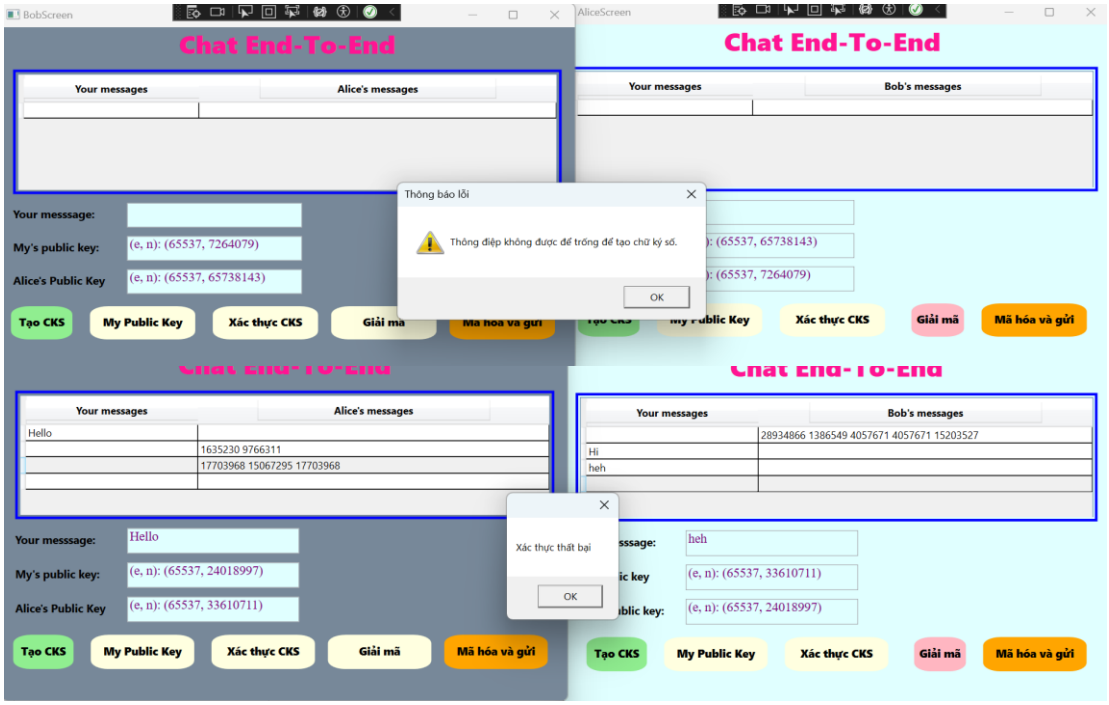
Hình 2.10: Thông báo lỗi khi tạo chữ ký số

Nếu ấn "Tạo CKS" khi chưa có public key thì sẽ hiện thông báo lỗi vì không có giá trị n để tạo CKS.



Hình 2.11: Thông báo lỗi khi giải mã

Thông báo lỗi sẽ hiện ra khi người dùng không chọn dòng muốn giải mã.
Không thể "Xác thực CKS" trước khi đối phương chưa khởi tạo "CKS".



Hình 2.12: Thông báo lỗi khi tạo chữ ký số

Nếu không có message thì khi ấn "Tạo CKS" sẽ hiện ra thông báo lỗi không có thông điệp để đi qua hàm băm và tạo CKS.

Nếu chữ ký số chưa được khởi tạo trước khi gửi tin nhắn, tin nhắn sẽ không thể được xác thực.

2.2.2.2 Các biện pháp bảo mật đã áp dụng trong ứng dụng:

Sử dụng thuật toán RSA để mã hóa và giải mã

- Cặp khóa công khai và bí mật: RSA sử dụng cặp khóa công khai (public key) và bí mật (private key). Public key được dùng để mã hóa tin nhắn, trong khi private key được dùng để giải mã.
- Tính toán phi(n): Sử dụng hai số nguyên tố lớn p và q để tạo giá trị n và phi(n), đảm bảo độ khó của việc phân tích thừa số nguyên tố.
- Lũy thừa nhanh (ModExp): Tính toán hiệu quả để mã/ giải mã thông điệp.

Kiểm tra và loại bỏ các số không phải nguyên tố

Sử dụng thuật toán sàng Eratosthenes để chọn các số nguyên tố trong phạm vi lớn (1000-9999), giảm khả năng chọn số yếu.

Sử dụng chữ ký số để bảo đảm tính toàn vẹn và xác thực

- Tạo chữ ký số (Digital Signature): Băm thông điệp bằng thuật toán SHA-256, sau đó mã hóa kết quả băm bằng khóa bí mật của người gửi. Điều này tạo ra chữ ký số.
- Xác thực chữ ký số: Người nhận giải mã chữ ký số bằng khóa công khai của người gửi để kiểm tra giá trị băm của thông điệp, đảm bảo:
    + Tính toàn vẹn: Tin nhắn không bị chỉnh sửa.
    + Tính xác thực: Tin nhắn được gửi từ đúng người gửi.

Sử dụng thuật toán băm SHA-256

Kiểm tra tính hợp lệ của khóa và thông điệp

Xác thực khóa công khai

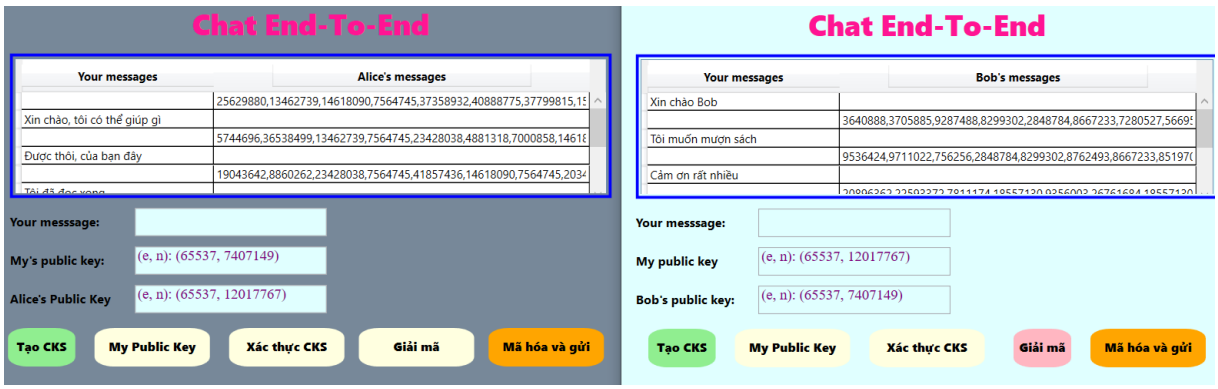
Quản lý lỗi và ngoại lệ

Chương 3: Kết quả thực hiện

3.1. Quy trình mã hóa RSA

3.1.1. Quy trình tạo khóa

Khi Bob/ Alice ấn nút "My PublicKey" trên giao diện, khóa công khai sẽ được hiển thị chéo nhau giữa 2 người dùng. Đảm bảo họ có thể sử dụng những PublicKey này để mã hóa dữ liệu.



Hình 3.1: Quy trình tạo khóa

3.1.2. Quy trình trao đổi khóa công khai

Mô tả cách để Bob & Alice trao đổi public key trước khi giao tiếp.

Alice tạo khóa:

- Nhấn nút "My Public Key" trên giao diện AliceScreen.
- Cách ứng dụng được thực thi:
  + PrimeFiller(): Tạo danh sách các số nguyên tố.
  + SetKeys(): Chọn ngẫu nhiên hai số nguyên tố p và q từ danh sách, tính toán n, e, d.
  + Khóa công khai của Alice (e, n) được lưu vào Server.AlicePublicKey . Server đóng vai trò trung gian để trao đổi khóa.
  + Khóa bí mật của Alice (d) chỉ được lưu trữ cục bộ trong biến private_key và không bao giờ được chia sẻ.
  + Khóa công khai (e, n) được hiển thị trên giao diện của Alice.
  + Nếu cửa sổ BobScreen đang mở, hàm UpdateAlicePublicKey() trong BobScreen được gọi để cập nhật khóa công khai của Alice trên giao diện của Bob.

Bob tạo khóa: (Tương tự)

3.1.3. Quy trình mã hóa

Alice gửi tin nhắn cho Bob:

- Alice nhập tin nhắn vào ô “Your message” trên giao diện AliceScreen.
- Alice nhấn nút "Mã hóa và gửi".
- Kiểm tra khóa Bob: Code kiểm tra xem Bob đã tạo khóa công khai chưa (Server.BobPublicKey và Server.BobN). Nếu chưa, hiển thị thông báo lỗi.
- Lấy khóa công khai của Bob: Alice lấy khóa công khai của Bob (e, n) từ Server.BobPublicKey và Server.BobN.
- Mã hóa từng ký tự: Vòng lặp duyệt qua từng ký tự trong tin nhắn của Alice. Mỗi ký tự được chuyển đổi thành mã ASCII (long) và được mã hóa bằng hàm EncryptWithPublicKey sử dụng khóa công khai của Bob.
- Lưu tin nhắn đã mã hóa: Các ký tự đã mã hóa được lưu vào một danh sách encoded. Danh sách này, cùng với tin nhắn gốc của Alice, được lưu vào một đối tượng PlainCipher và thêm vào Server.Messages.

3.1.4. Quy trình giải mã

Bob giải mã tin nhắn từ Alice:

- Bob chọn tin nhắn cần giải mã trong DataGrid trên BobScreen. Dữ liệu của DataGrid được liên kết với Server.Messages.
- Bob nhấn "Giải mã": Sự kiện btngiaima_Click_1 được kích hoạt.
- Kiểm tra tin nhắn được chọn: Code kiểm tra xem Bob đã chọn một tin nhắn trong DataGrid chưa. Nếu chưa, hiển thị thông báo lỗi.
- Lấy tin nhắn mã hóa: Code lấy tin nhắn đã mã hóa (ciphertext1) từ đối tượng PlainCipher được chọn.
- Giải mã từng ký tự: Vòng lặp trong hàm BobDecryptsMessage giải mã từng ký tự bằng khóa bí mật của Bob, sử dụng phép toán (ký tự mã hóa ^ BobPrivateKey) mod BobN.
- Hiển thị tin nhắn: Tin nhắn đã giải mã được hiển thị trong một MessageBox.

3.1.5. Quy trình xác thực chữ ký số

Bước 1: Tạo chữ ký số (Alice):

- Đầu tiên sẽ băm thông điệp, Alice sử dụng một hàm băm để tạo ra giá trị băm duy nhất từ thông điệp cần gửi và đầu ra của hàm băm sẽ là một giá trị băm chuỗi dữ liệu có độ dài cố định và là unique cho thông điệp cần gửi.
- Sau đó là mã hoá giá trị băm là sử dụng private key của mình (AlicePrivateKey) để mã hoá giá trị băm rồi chúng ta sẽ có chữ ký số.
- Đồng thời dùng PublicKey của Bob để mã hóa thông điệp, tạo ciphertext.

Bước 2: Gửi thông điệp & chữ ký số:

Alice gửi đi 2 phần dữ liệu là thông điệp được mã hóa bằng public key của Bob (Ciphertext) và chữ ký số đã được tạo ra trước đó.

Bước 3: Xác thực chữ ký số (Bob)

Bob sẽ giải mã chữ ký số mà Alice gửi bằng public key:

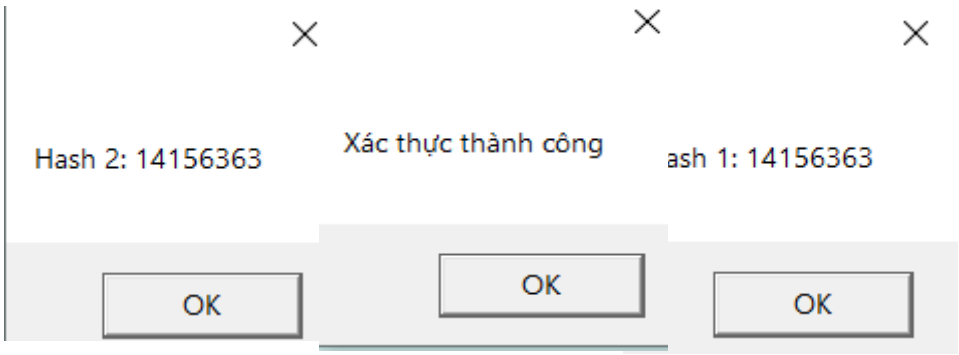
Hash1 = decrypt(signature, publicKeyA)

tại đây decryptedHash sẽ là giá trị băm gốc nếu chữ ký số thực sự được tạo ra bằng khóa riêng của A.

Giải mã ciphertext bằng chính Private key của Bob để cho ra plaintext. Sau đó tính thêm giá trị băm từ thông điệp này: Tại đây Bob sử dụng hàm băm mà Alice dùng ở bước 1 để tính giá trị băm từ thông điệp nhận được.

Hash2 = hash(message)

lúc này ta so sánh Hash1 với Hash2, nếu bằng nhau thì lúc này xác thực thành công là do người A gửi. và nếu khác nhau thì có thể là do bị giả mạo hoặc thay đổi lúc này xác thực không thành công.



Hình 3.2: Quá trình xác thực chữ ký số

3.2. Demo sản phẩm (Trực tiếp)

PHẦN 3: KẾT LUẬN

1. Kết quả đạt được

Trong quá trình nghiên cứu và phát triển ứng dụng chat bảo mật end-to-end sử dụng hệ mật RSA, chúng em đã đạt được những kết quả như sau:

- **Bảo mật tin nhắn:** Tin nhắn được mã hóa và chỉ có thể giải mã tại thiết bị nhận nhờ vào hệ thống khóa công khai (**public key**) và khóa riêng (**private key**) của RSA. Điều này đảm bảo rằng ngay cả khi dữ liệu bị chặn giữa đường, kẻ tấn công cũng không thể đọc được nội dung tin nhắn. Hệ thống bảo mật được kiểm tra kỹ lưỡng qua các bài kiểm thử và đánh giá bảo mật chuyên sâu, khẳng định tính an toàn và tin cậy.
- **Quản lý khóa an toàn:** Các khóa RSA được tạo ra chúng em lưu trữ một cách an toàn nhờ cơ chế quản lý khóa tiên tiến. Người dùng không cần lo lắng về việc mất khóa hoặc khóa bị đánh cắp vì hệ thống đã triển khai các lớp bảo vệ đa tầng để ngăn chặn các hành vi truy cập trái phép.
- **Giao diện người dùng thân thiện:** Giao diện được thiết kế tối giản, dễ sử dụng nhưng vẫn đảm bảo tính thẩm mỹ. Người dùng có thể dễ dàng thao tác, từ việc gửi tin nhắn.
- **Bảo vệ thông tin mật khẩu**: Nếu hacker truy cập được vào cơ sở dữ liệu, thay vì nhìn thấy mật khẩu dạng văn bản thuần (plain text), họ sẽ chỉ thấy các giá trị hash. Nhờ đó, ngay cả khi dữ liệu bị rò rỉ, mật khẩu của người dùng vẫn an toàn hơn.

2. Đánh giá sản phẩm

- **Điểm mạnh:**
    + Sử dụng hệ mật RSA, một trong những thuật toán mã hóa mạnh mẽ nhất hiện nay, được triển khai một cách hiệu quả và tối ưu.
    + Giao diện người dùng khá đẹp mắt, bố trí giao diện hợp lý kể cả những người không rành về công nghệ cũng có thể dễ dàng sử dụng.
    + hệ thống được chúng em xây dựng linh hoạt, có khả năng mở rộng để tích hợp thêm các tính năng khác trong tương lai mà không ảnh hưởng đến hiệu năng hiện tại.
    + Ứng dụng đáp ứng các tiêu chí về bảo mật end-to-end, bảo vệ quyền riêng tư của người dùng. Điều này phù hợp với nhu cầu của các ứng dụng ngày càng tăng về bảo mật thông tin trong thời đại số.

- **Hạn chế:**
    + Performance: Việc sử dụng hệ mật RSA với độ dài khóa lớn có thể làm chậm quá trình mã hóa và giải mã trên các thiết bị cấu hình thấp. Điều này gây ảnh hưởng đến trải nghiệm của một số người dùng và khi lượng người dùng tăng lên đột biến, hệ thống có thể đối mặt với các vấn đề như giảm tốc độ xử lý hoặc tăng tải trên máy chủ.
    + Thách thức về trải nghiệm người dùng: Mặc dù giao diện đã được thiết kế tối giản, nhưng một số người dùng mới vẫn cần thêm hướng dẫn để hiểu rõ cách sử dụng đầy đủ các chức năng.
    + Tính năng giới hạn: Hiện tại, ứng dụng chỉ tập trung vào nhắn tin và bảo mật, chưa hỗ trợ các tính năng khác như gọi video, chia sẻ file lớn, hoặc quản lý nhóm chat phức tạp.

3. Đề xuất cải thiện

Sau khi phân tích những hạn chế còn tồn tại và tham khảo một số ứng dụng chat như zalo, messenger chúng em nhận thấy cần có các đề xuất cải thiện để tăng cường hiệu quả và trải nghiệm của ứng dụng. Những đề xuất này được đưa ra dựa trên mục tiêu tối ưu hóa hiệu suất, đáp ứng tốt hơn nhu cầu thực tế và chuẩn bị cho sự phát triển dài hạn của hệ thống:

- **Nâng cao hiệu suất mã hóa:** Kết hợp RSA với các thuật toán mã hóa đối xứng như AES để tận dụng sức mạnh của cả hai phương pháp. Điều này sẽ cải thiện tốc độ mã hóa và giải mã, đặc biệt trên các thiết bị cấu hình thấp.
- **Tăng cường bảo mật:** Sử dụng xác thực hai lớp (2FA) để nâng cao tính bảo mật, cung cấp cảnh báo khi phát hiện các hành vi bất thường hoặc nghi ngờ tài khoản bị tấn công và thực hiện kiểm tra định kỳ và cập nhật bảo mật để phòng tránh các mối đe dọa mới.
- **Cải thiện hạ tầng:** Tăng cường máy chủ lưu trữ và sử dụng các giải pháp cân bằng tải để đảm bảo hiệu suất ổn định khi lượng người dùng tăng cao và tích hợp các công nghệ mới như điện toán đám mây để tăng cường tính linh hoạt và khả năng mở rộng.
- **Tối ưu hóa trải nghiệm người dùng:** Cung cấp thêm các hướng dẫn sử dụng chi tiết hoặc tích hợp chatbot hỗ trợ trực tiếp trong ứng dụng cũng như tùy chỉnh giao diện theo sở thích cá nhân của người dùng để tăng tính cá nhân hóa.
- **Bổ sung tính năng mới:** có thể hỗ trợ gọi video và âm thanh với mức độ bảo mật tương tự như tin nhắn văn bản, cũng như cung cấp khả năng chia sẻ file lớn, đảm

bảo rằng các file này cũng được mã hóa trước khi gửi đi và phát triển tính năng nhóm chat với quản lý quyền truy cập linh hoạt và bảo mật.

- **Cải tiến quá trình trao đổi khóa:** Bằng các thuật toán như Diffie Hellman để quá trình gửi/ nhận tin nhắn được bảo mật hơn.
- **Xây dựng một ứng dụng chat thực sự:** Bằng cách 2 người ở 2 thiết bị có thể nhắn tin trực tiếp được cho nhau mà vẫn ứng dụng được các giải pháp bảo mật mà nhóm đã đưa ra.

Từ những kết quả và đề xuất đã trình bày, nhóm 6 chúng em hy vọng rằng báo cáo không chỉ mang lại cái nhìn toàn diện về tầm quan trọng của an ninh mạng mà còn cung cấp các phương pháp và giải pháp thực tiễn để cải thiện chất lượng ứng dụng chat end-to-end. Chúng em kỳ vọng rằng các đề xuất này sẽ giúp ứng dụng chat phát triển thành một ứng dụng tiên tiến, đáng tin cậy, phục vụ tốt hơn nhu cầu bí mật của cộng đồng.

Cuối cùng, nhóm 6 xin gửi lời cảm ơn chân thành đến TS. Lê Thị Anh vì sự hỗ trợ và chỉ dẫn tận tình trong suốt quá trình học tập và thực hiện báo cáo. Những đóng góp quý báu từ cô đã giúp nhóm chúng em hoàn thiện hơn không chỉ trong nội dung nghiên cứu mà còn trong tư duy và cách tiếp cận vấn đề. Chúng em rất mong nhận được thêm ý kiến đóng góp từ cô giáo để hoàn thiện báo cáo cũng như các sản phẩm nghiên cứu trong tương lai.

TÀI LIỆU THAM KHẢO

Tài liệu Tiếng Việt

[1] Bài giảng An ninh mạng (Network security) . Học viện kỹ thuật mật mã. Phan Đình Diệu, Giáo trình lý thuyết mật mã và an toàn thông tin, Nhà xuất bản Đại học Quốc gia Hà Nội, 2002.

[2] ViettelIDC, End to end Encryption: Giải thích chi tiết và ứng dụng, 20/05/2024, https://viettelidc.com.vn/tin-tuc/end-to-end-encryption-ma-hoa-dau-cuoi [Truy cập 10/12/2024]

Tài liệu Tiếng Anh

[1] Xin Zhou, Xiaofei Tang, Research and implementation of RSA algorithm for encryption and decryption, 12/2011, https://ieeexplore.ieee.org/abstract/document/6021216 [Truy cập 11/11/2024]

[2] Geekforgeeks, RSA Algorithm in Cryptography, 11/8/2024, **https://www.geeksforgeeks.org/rsa-algorithm-cryptography/** [Truy cập 11/2024]

[3] Shanika, Splunk is an Industry Leader in Security, Splunk Blog, 26/11/2024, https://www.splunk.com/en_us/blog/learn/rsa-algorithm-cryptography.html#:~:text=RSA%20is%20a%20popular%20and,the%20source%20of%20a%20message [Truy cập ngày 1/12/2024]

[4] RSA Cryptography Standard, October 27, 2012, Network security essentials applications and standards - Pearson - Stallings, William (2016-2017)