

**BỘ CÔNG THƯƠNG
TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP
KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO BÀI TẬP LỚN THUỘC HỌC PHẦN
AN TOÀN VÀ BẢO MẬT THÔNG TIN**

Đề tài:

**ỨNG DỤNG CỦA RSA TRONG CHỮ KÝ SỐ ĐIỆN TỬ.
DEMO SẢN PHẨM MINH HỌA.**

GVHD: TS. Lê Thị Anh

Nhóm: 1

Mã lớp: 2023IT6001001

Hà Nội, 2023

DANH SÁCH THÀNH VIÊN VÀ BẢNG PHÂN CÔNG NHIỆM VỤ NHÓM 1

STT	HỌ VÀ TÊN	NHIỆM VỤ TUẦN 1	MỨC ĐỘ HOÀN THÀNH
1	Nguyễn Phương Thảo (Nhóm trưởng)	Tổng hợp, viết báo cáo	100%
2	Tô Thị Thu Trang	Tìm hiểu về chữ ký số điện tử	100%
3	Nguyễn Đình Công	Giới thiệu về RSA	100%
4	Lê Minh Anh	Tìm hiểu Hàm băm	100%
5	Nguyễn Đức Minh	Ứng dụng của RSA	100%

STT	HỌ VÀ TÊN	NHIỆM VỤ TUẦN 2	MỨC ĐỘ HOÀN THÀNH
1	Nguyễn Phương Thảo (Nhóm trưởng)	Chốt ý tưởng thiết kế	100%
2	Tô Thị Thu Trang	Lên ý tưởng thiết kế	100%
3	Nguyễn Đình Công	Lên ý tưởng thiết kế	100%
4	Lê Minh Anh	Lên ý tưởng thiết kế	100%
5	Nguyễn Đức Minh	Lên ý tưởng thiết kế	100%

STT	HỌ VÀ TÊN	NHIỆM VỤ TUẦN 3	MỨC ĐỘ HOÀN THÀNH
1	Nguyễn Phương Thảo (Nhóm trưởng)	Cài đặt chương trình trên môi trường Jupyter notebook	100%
2	Tô Thị Thu Trang	Cài đặt GUI và kết nối thuật toán	100%
3	Nguyễn Đình Công	Cài đặt thuật toán RSA	100%
4	Lê Minh Anh	Cài đặt thuật toán RSA	100%
5	Nguyễn Đức Minh	Cài đặt GUI và kết nối thuật toán	100%

STT	HỌ VÀ TÊN	NHIỆM VỤ TUẦN 4	MỨC ĐỘ HOÀN THÀNH
1	Nguyễn Phương Thảo (Nhóm trưởng)	Tổng hợp, viết báo cáo	100%
2	Tô Thị Thu Trang	Góp ý về trình bày báo cáo	100%
3	Nguyễn Đình Công	Góp ý về trình bày báo cáo	100%
4	Lê Minh Anh	Góp ý về trình bày báo cáo	100%
5	Nguyễn Đức Minh	Góp ý về trình bày báo cáo	100%

MỤC LỤC

Chương 1: TỔNG QUAN NỘI DUNG LÝ THUYẾT	5
1. Lí do chọn lựa đề tài.....	5
2. Mật mã khóa công khai	5
2.1. Khái niệm.....	5
2.2. Lý thuyết số.....	7
2.3. Các thuật toán mã hóa công khai phổ biến.....	7
3. Giới thiệu về RSA	7
3.1. Khái niệm về RSA	7
3.2. Cơ sở toán học.....	8
3.3. Cách thức hoạt động	8
3.4. Ưu, nhược điểm của RSA.....	10
4. Ứng dụng của RSA	11
4.1. Một số ứng dụng chính.....	11
4.2. Một số ứng dụng về các lĩnh vực cụ thể tại Việt Nam	11
5. Giới thiệu về chữ ký số điện tử	12
5.1. Khái niệm.....	12
5.2. Vai trò của khóa công khai và khóa bí mật trong chữ ký số.....	12
6. Hàm băm	14
6.1. Khái niệm.....	14
6.2. Các loại hàm băm.....	14
6.3. Cách hoạt động của hàm băm.....	14
6.4. Một số hàm băm phổ biến	15
Chương 2: PHÂN TÍCH THIẾT KẾ.....	16
1. Ý tưởng thiết kế.....	16

2. Quy trình thực hiện.....	16
2.1. Cài đặt thuật toán RSA.....	16
2.2. Cài đặt giao diện người dùng GUI.....	26
2.3. Kết nối giao diện và thuật toán	29
Chương 3: KẾT QUẢ THỰC HIỆN	32
1. Giao diện chương trình demo.....	32
2. Cài đặt và triển khai	32
3. Kết quả đạt được	37
4. Hạn chế.....	37

DANH MỤC HÌNH ẢNH

Hình 3.1: Giao diện chương trình demo	32
Hình 3.2: Tạo 2 số nguyên tố	32
Hình 3.3: Tạo khoá công khai và khoá bí mật tương ứng	33
Hình 3.4: Nhập văn bản để ký	33
Hình 3.5: Kết quả sau khi ký văn bản	33
Hình 3.6: Xuất kết quả và lưu file vào máy tính	33
Hình 3.7: Cửa sổ xuất kết quả	34
Hình 3.8: Nhập văn bản xác thực	34
Hình 3.9: Cửa sổ chọn file xác thực	35
Hình 3.10: Xác thực thành công.....	35
Hình 3.11: Xác thực không thành công.....	36

LỜI CẢM ƠN!

Kính gửi Tiến sĩ Lê Thị Anh,

Nhóm chúng em xin gửi lời cảm ơn chân thành đến Tiến sĩ đã trực tiếp hướng dẫn nhóm chúng em thực hiện đề tài "Ứng dụng của RSA trong chữ ký số điện tử".

Trong suốt quá trình thực hiện đề tài, Tiến sĩ đã cung cấp cho chúng em những kiến thức và kinh nghiệm quý báu về thuật toán RSA và chữ ký số điện tử. Những kiến thức và kinh nghiệm này đã giúp chúng em có thể hiểu rõ hơn về các vấn đề liên quan đến bảo mật thông tin.

Nhờ sự hướng dẫn tận tình của Tiến sĩ, nhóm chúng em đã có thể hoàn thành đề tài của mình một cách xuất sắc và đạt được kết quả tốt. Chúng em đã có thể hiểu rõ hơn về thuật toán RSA và ứng dụng của nó trong chữ ký số điện tử.

Nhóm chúng em xin chân thành cảm ơn Tiến sĩ vì những đóng góp to lớn của Tiến sĩ đối với nhóm chúng em. Chúng em xin hứa sẽ tiếp tục học tập và nghiên cứu để có thể đạt được những thành tích cao hơn nữa.

Xin chân thành cảm ơn!

Chương 1: TỔNG QUAN NỘI DUNG LÝ THUYẾT

1. Lí do chọn lựa đề tài

Trong thời đại công nghệ thông tin phát triển mạnh mẽ, thông tin trở thành tài sản quan trọng của mỗi cá nhân, tổ chức. Do đó, an toàn và bảo mật thông tin là một vấn đề quan trọng cần được quan tâm.

Chữ ký số là một công nghệ bảo mật thông tin được sử dụng rộng rãi trong nhiều lĩnh vực, bao gồm:

- Ký điện tử: Chữ ký số được sử dụng để ký các tài liệu điện tử, chẳng hạn như hợp đồng, hóa đơn,...
- Bảo mật thông tin: Chữ ký số được sử dụng để bảo vệ thông tin nhạy cảm, chẳng hạn như thông tin cá nhân, thông tin tài chính,...
- Xác thực người dùng: Chữ ký số được sử dụng để xác thực người dùng khi truy cập các hệ thống và dịch vụ điện tử.

RSA là một thuật toán mã hóa khóa công khai được sử dụng rộng rãi trong nhiều ứng dụng khác nhau, bao gồm cả chữ ký số. Ứng dụng của RSA trong chữ ký số giúp đảm bảo tính xác thực, tính toàn vẹn và tính không chối bỏ của tài liệu điện tử.

Với những lý do trên, nhóm chúng tôi đã lựa chọn đề tài "Ứng dụng của RSA trong chữ ký số điện tử". Đề tài này có ý nghĩa quan trọng trong việc nâng cao nhận thức của sinh viên về các vấn đề liên quan đến bảo mật thông tin, đặc biệt là chữ ký số.

Ngoài ra, đề tài này cũng giúp nhóm chúng tôi có thể tìm hiểu và nghiên cứu sâu hơn về thuật toán RSA và ứng dụng của nó trong chữ ký số. Những kiến thức và kinh nghiệm mà nhóm chúng tôi học hỏi được từ đề tài này sẽ là nền tảng vững chắc để chúng tôi có thể tiếp tục học tập và nghiên cứu trong tương lai.

2. Mật mã khóa công khai

2.1. Khái niệm

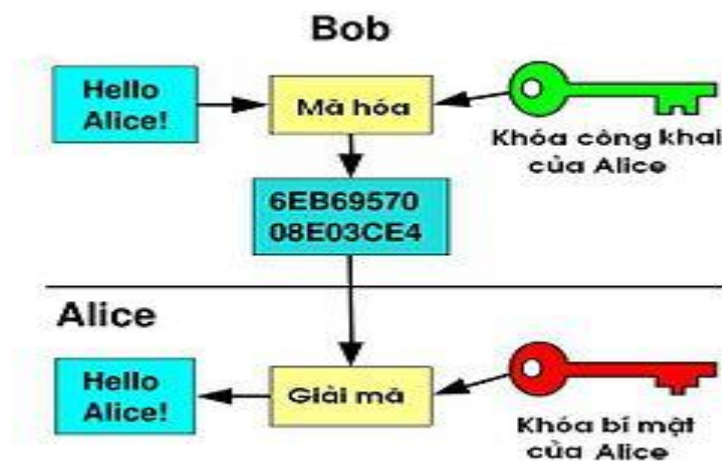
Mật mã khóa công khai là một loại mật mã hóa sử dụng hai khóa để mã hóa và giải mã thông tin.

Để mã hóa thông tin bằng mật mã khóa công khai, người ta sử dụng khóa công khai của người nhận. Khóa công khai được sử dụng để tạo ra một bản mã của thông tin. Bản mã này chỉ có thể được giải mã bằng khóa riêng của người nhận.

Để giải mã thông tin bằng mật mã khóa công khai, người ta sử dụng khóa riêng của mình. Khóa riêng được sử dụng để biến bản mã thành thông tin gốc.



Hình 2.1: Sơ đồ mã hóa công kh



Hình 2.2 Ví dụ minh họa mã hóa công khai

Ví dụ:

Giả sử Alice muốn gửi một tin nhắn bí mật cho Bob. Cô ấy có thể tạo một cặp khóa công khai và khóa riêng. Cô ấy chia sẻ khóa công khai của mình với Bob, nhưng giữ khóa riêng cho riêng mình.

Bob có thể sử dụng khóa công khai của Alice để mã hóa tin nhắn của mình. Anh ta làm điều này bằng cách sử dụng thuật toán mã hóa khóa công khai để biến văn bản rõ ràng thành văn bản mã hóa.

Sau đó, Bob gửi tin nhắn mã hóa cho Alice. Alice có thể sử dụng khóa riêng của mình để giải mã tin nhắn của Bob. Cô ấy làm điều này bằng cách sử dụng thuật toán mã hóa khóa công khai để biến văn bản mã hóa trở lại văn bản rõ ràng.

Tin nhắn hiện đã được giải mã và chỉ Alice mới có thể đọc được.

2.2. Lý thuyết số

Nền tảng toán học của mã hóa công khai dựa trên hai định lý toán học chính:

- Định lý Fermat về số nguyên tố lớn: Định lý này nói rằng, với hai số nguyên tố lớn p và q , không có cách nào để tìm ra các số nguyên tố p và q chỉ bằng cách biết sản phẩm của chúng, pq .
- Định lý Euclid mở rộng: Định lý này cho phép chúng ta tính toán phần dư của phép chia Euclide ngược.

2.3. Các thuật toán mã hóa công khai phổ biến

- Diffie-Hellman: Diffie-Hellman là một thuật toán trao đổi khóa công khai được phát triển bởi Whitfield Diffie và Martin Hellman vào năm 1976. Nó cho phép hai bên tạo ra một khóa bí mật chung mà không cần phải trao đổi thông tin bí mật trước đó.
- RSA: RSA là một thuật toán mã hóa khóa công khai được phát triển bởi Ron Rivest, Adi Shamir và Leonard Adleman vào năm 1977.
- ElGamal: ElGamal là một thuật toán mã hóa khóa công khai được phát triển bởi Taher ElGamal vào năm 1985. Nó tương tự như RSA, nhưng sử dụng một hàm mũ khác.
- ECDSA: ECDSA là một thuật toán chữ ký số khóa công khai được phát triển bởi Certicom vào năm 1998. Nó được sử dụng để tạo chữ ký số cho dữ liệu.

3. Giới thiệu về RSA

3.1. Khái niệm về RSA

RSA là một thuật toán mã hóa khóa công khai (public-key encryption). RSA sử dụng hai khóa khác nhau để mã hóa và giải mã: khóa công khai (public key) và khóa bí

mật (private key). Khóa công khai có thể được chia sẻ cho tất cả mọi người, nhưng khóa bí mật chỉ được giữ bí mật bởi người nhận.

3.2. Cơ sở toán học

RSA dựa trên bài toán phân tích số nguyên tố. Bài toán này là tìm các thừa số nguyên tố của một số nguyên lớn. Hiện nay, chưa có phương pháp phân tích số nguyên tố nào có hiệu quả cao cho các số nguyên lớn.

- Số nguyên tố: Số nguyên tố là số nguyên chỉ có hai ước số là 1 và chính nó.
- Thừa số nguyên tố: Thừa số nguyên tố của một số nguyên là số nguyên chia hết cho số nguyên đó và là thừa số nguyên tố của số nguyên đó.
- Định lý nhỏ Fermat: Định lý nhỏ Fermat nói rằng nếu p là một số nguyên tố, thì $a^{(p-1)} \bmod p = 1$ cho mọi số nguyên a không chia hết cho p .
- Bài toán phân tích số nguyên tố: Bài toán phân tích số nguyên tố là tìm các thừa số nguyên tố của một số nguyên lớn.

3.3. Cách thức hoạt động

Quá trình mã hóa dữ liệu bằng RSA được thực hiện như sau:

- Người gửi sử dụng khóa công khai của người nhận để tạo ra một số nguyên lớn ngẫu nhiên.
- Người gửi sử dụng số nguyên ngẫu nhiên đó để mã hóa dữ liệu.

Quá trình giải mã dữ liệu bằng RSA được thực hiện như sau:

- Người nhận sử dụng khóa bí mật của mình để giải mã dữ liệu đã mã hóa.

a. Ví dụ quá trình mã hóa:

Giả sử Alice muốn gửi một thông điệp cho Bob. Alice có khóa công khai của Bob, nhưng không có khóa bí mật của Bob. Để mã hóa thông điệp, Alice thực hiện các bước sau:

1. Alice chọn hai số nguyên lớn ngẫu nhiên p và q .
2. Alice tính $n = p * q$.
3. Alice tính $\phi(n) = (p - 1)(q - 1)$.

4. Alice chọn một số nguyên e sao cho $1 < e < \phi(n)$ và e và $\phi(n)$ là ước số nguyên tố của nhau.
5. Alice tính d sao cho $d * e \bmod \phi(n) = 1$.

Khóa công khai của Alice là cặp (n, e) . Khóa bí mật của Alice là d .

Để mã hóa thông điệp, Alice thực hiện các bước sau:

1. Alice chia thông điệp thành các khối nhỏ hơn.
2. Đối với mỗi khối, Alice sử dụng khóa công khai của Bob để mã hóa khối đó.

Để mã hóa một khối, Alice thực hiện phép nhân modulo n giữa khối đó và khóa công khai của Bob.

Ví dụ: Giả sử Alice muốn mã hóa thông điệp "Hello". Thông điệp "Hello" có độ dài 5 byte, tương đương với 40 bit. Alice chia thông điệp thành 4 khối nhỏ hơn, mỗi khối có độ dài 10 bit, tương đương với 8 byte.

Alice sử dụng khóa công khai của Bob là $(1234567890, 13)$.

Để mã hóa khối đầu tiên của thông điệp, Alice thực hiện phép nhân modulo 1234567890 giữa khối đó và khóa công khai của Bob. Phép nhân modulo 1234567890 tương đương với phép cộng modulo 1234567890.

Ví dụ: Nếu khối đầu tiên của thông điệp là "H", thì Alice thực hiện phép nhân modulo 1234567890 giữa "H" và 13. Kết quả của phép nhân là 10.

Alice lặp lại các bước trên để mã hóa tất cả các khối của thông điệp.

b. Ví dụ quá trình giải mã:

Để giải mã thông điệp, Bob sử dụng khóa bí mật của mình. Bob thực hiện các bước sau:

1. Bob nhận được dữ liệu đã mã hóa từ Alice.
2. Bob chia dữ liệu đã mã hóa thành các khối nhỏ hơn.
3. Đối với mỗi khối, Bob sử dụng khóa bí mật của mình để giải mã khối đó.

Để giải mã một khối, Bob thực hiện phép nhân modulo n giữa khối đó và khóa bí mật của mình.

Ví dụ: Giả sử Bob nhận được dữ liệu đã mã hóa là 10, 20, 30, 40. Bob sử dụng khóa bí mật của mình là 12.

Để giải mã khối đầu tiên của dữ liệu đã mã hóa, Bob thực hiện phép nhân modulo 1234567890 giữa 10 và 12. Kết quả của phép nhân là 120.

Bob lặp lại các bước trên để giải mã tất cả các khối của dữ liệu đã mã hóa.

Kết quả của quá trình giải mã là thông điệp gốc mà Alice đã gửi.

3.4. Ưu, nhược điểm của RSA

a. Ưu điểm

Mạnh mẽ: RSA dựa trên bài toán phân tích số nguyên tố, đây là một bài toán khó giải. Hiện nay, chưa có phương pháp phân tích số nguyên tố nào có hiệu quả cao đối với các số nguyên lớn. Do đó, RSA được cho là một thuật toán mã hóa mạnh và an toàn.

Linh hoạt: Kích thước của khóa RSA có thể được điều chỉnh để phù hợp với nhu cầu bảo mật của ứng dụng. Khóa RSA càng lớn thì càng an toàn, nhưng tốc độ mã hóa và giải mã sẽ chậm hơn.

Ổn định: RSA đã được thử nghiệm và sử dụng trong nhiều năm và được chứng minh là một thuật toán ổn định. RSA không có các lỗ hổng bảo mật nghiêm trọng đã được phát hiện.

b. Nhược điểm

Yêu cầu hai khóa: RSA sử dụng hai khóa khác nhau, khóa công khai và khóa bí mật. Khóa công khai có thể được chia sẻ cho tất cả mọi người, nhưng khóa bí mật chỉ được giữ bí mật bởi người nhận. Điều này có thể gây bất tiện trong một số trường hợp, chẳng hạn như khi người nhận muốn gửi thông điệp cho nhiều người.

Tốc độ mã hóa và giải mã của RSA phụ thuộc vào kích thước của khóa. Khóa RSA càng lớn thì tốc độ mã hóa và giải mã sẽ chậm hơn. Điều này có thể là một vấn đề trong các ứng dụng yêu cầu tốc độ cao, chẳng hạn như truyền dữ liệu trực tiếp.

4. Ứng dụng của RSA

4.1. Một số ứng dụng chính

Mã hóa email và dữ liệu: RSA được sử dụng để mã hóa email và dữ liệu nhạy cảm bằng cách sử dụng khóa công khai của người nhận. Người gửi sử dụng khóa công khai của người nhận để mã hóa dữ liệu, và người nhận sử dụng khóa bí mật của mình để giải mã dữ liệu. Điều này giúp bảo vệ dữ liệu khỏi bị đọc bởi những người không có khóa bí mật.

Chữ ký điện tử: RSA được sử dụng để tạo chữ ký điện tử bằng cách sử dụng khóa bí mật của người ký. Người ký sử dụng khóa bí mật của mình để tạo chữ ký điện tử, và người nhận có thể sử dụng khóa công khai của người ký để xác minh chữ ký. Điều này giúp xác thực người ký và bảo vệ tính toàn vẹn của dữ liệu.

Giao thức TLS/SSL: RSA được sử dụng trong giao thức TLS/SSL để bảo mật các kết nối mạng, chẳng hạn như kết nối web. Giao thức TLS/SSL sử dụng RSA để mã hóa dữ liệu được truyền qua kết nối mạng. Điều này giúp bảo vệ dữ liệu khỏi bị đọc bởi những người không được phép.

Xác thực người dùng: RSA có thể được sử dụng để xác thực người dùng bằng cách sử dụng khóa bí mật của người dùng.

Chứng thực: RSA có thể được sử dụng để chứng thực các tổ chức hoặc cá nhân.

Trao đổi khóa: RSA có thể được sử dụng để trao đổi khóa bí mật giữa hai bên.

4.2. Một số ứng dụng về các lĩnh vực cụ thể tại Việt Nam

Lĩnh vực ngân hàng: RSA được ứng dụng để bảo mật các giao dịch ngân hàng trực tuyến, bao gồm xác thực người dùng, mã hóa dữ liệu giao dịch và chữ ký điện tử. Nhờ đó, các giao dịch ngân hàng trực tuyến tại Việt Nam được đảm bảo an toàn, bảo mật, ngăn chặn các hành vi gian lận, trộm cắp thông tin.

Lĩnh vực thương mại điện tử: RSA cũng được ứng dụng rộng rãi trong lĩnh vực thương mại điện tử, bao gồm xác thực người dùng, mã hóa dữ liệu giao dịch và chữ ký điện tử. Nhờ đó, các giao dịch thương mại điện tử tại Việt Nam được đảm bảo an toàn, bảo mật, giúp người dùng yên tâm mua sắm trực tuyến.

Lĩnh vực Chính phủ điện tử: RSA được ứng dụng trong các ứng dụng Chính phủ điện tử, bao gồm xác thực người dùng, mã hóa dữ liệu giao dịch và chữ ký điện tử. Nhờ đó, các ứng dụng Chính phủ điện tử tại Việt Nam được đảm bảo an toàn, bảo mật, giúp người dân và doanh nghiệp thuận tiện trong việc thực hiện các thủ tục hành chính trực tuyến.

Lĩnh vực y tế: RSA được ứng dụng trong các ứng dụng y tế, bao gồm mã hóa dữ liệu bệnh nhân, chữ ký điện tử và xác thực người dùng. Nhờ đó, các dữ liệu y tế tại Việt Nam được bảo mật, tránh bị rò rỉ hoặc sử dụng trái phép.

Lĩnh vực giáo dục: RSA được ứng dụng trong các ứng dụng giáo dục, bao gồm mã hóa dữ liệu học sinh, chữ ký điện tử và xác thực người dùng. Nhờ đó, các dữ liệu giáo dục tại Việt Nam được bảo mật, tránh bị thay đổi hoặc xóa bỏ trái phép.

Ngoài ra, RSA còn được ứng dụng trong nhiều lĩnh vực khác tại Việt Nam, bao gồm:

- Lĩnh vực quân sự
- Lĩnh vực vận tải
- Lĩnh vực truyền thông
- Lĩnh vực sản xuất
- Lĩnh vực viễn thông
- Lĩnh vực nghiên cứu

5. Giới thiệu về chữ ký số điện tử

5.1. Khái niệm

Chữ ký số điện tử là một dạng chữ ký điện tử được sử dụng để xác thực tính toàn vẹn và tính xác thực của dữ liệu. Nó được tạo ra bằng cách sử dụng thuật toán mã hóa khóa công khai.

Chữ ký số điện tử bao gồm hai phần:

- Dữ liệu được ký: Đây là dữ liệu mà người ký muốn xác thực.
- Chữ ký số: Đây là một chuỗi ký tự được tạo ra bằng cách sử dụng khóa bí mật của người ký.

5.2. Vai trò của khóa công khai và khóa bí mật trong chữ ký số

Chữ ký số sử dụng cặp khóa công khai và khóa bí mật để tạo và xác thực chữ ký. Người gửi sử dụng khóa bí mật để tạo chữ ký, trong khi người nhận sử dụng khóa công khai để xác thực tính toàn vẹn và nguồn gốc của dữ liệu.

a. Vai trò của khóa công khai (Public key)

Public key là một khóa công khai, có thể được chia sẻ với bất kỳ ai. Public key được sử dụng để xác minh chữ ký số.

Khi người nhận nhận được chữ ký số, họ sử dụng public key của người ký để xác minh chữ ký số. Nếu chữ ký số được xác minh thành công, thì người nhận có thể tin tưởng rằng chữ ký số được tạo bởi người ký có private key tương ứng.

Việc sử dụng public key để xác minh chữ ký số giúp đảm bảo rằng chỉ có người sở hữu private key mới có thể tạo được chữ ký số đó. Điều này giúp xác thực người ký và ngăn chặn việc giả mạo chữ ký số.

b. Vai trò của khóa bí mật (Private key)

Trong chữ ký số, private key đóng vai trò quan trọng trong việc xác thực người ký và bảo vệ tính toàn vẹn của dữ liệu.

– Xác thực người ký

Khi người ký tạo chữ ký số, họ sử dụng private key để tạo một chữ ký số. Chữ ký số này sau đó được gửi cho người nhận.

Người nhận sử dụng public key của người ký để xác minh chữ ký số. Nếu chữ ký số được xác minh thành công, thì người nhận có thể tin tưởng rằng chữ ký số được tạo bởi người ký có private key tương ứng.

Việc sử dụng private key để tạo chữ ký số giúp đảm bảo rằng chỉ có người sở hữu private key mới có thể tạo được chữ ký số đó. Điều này giúp xác thực người ký và ngăn chặn việc giả mạo chữ ký số.

– Bảo vệ tính toàn vẹn của dữ liệu

Chữ ký số cũng giúp bảo vệ tính toàn vẹn của dữ liệu. Khi người ký tạo chữ ký số, họ sử dụng private key để tạo một chữ ký số dựa trên dữ liệu cần được bảo vệ.

Nếu dữ liệu bị thay đổi sau khi được ký, thì chữ ký số sẽ không còn hợp lệ. Điều này giúp đảm bảo rằng dữ liệu không bị thay đổi kể từ khi được ký.

6. Hàm băm

6.1. Khái niệm

Hàm băm là một hàm toán học biến một chuỗi ký tự có độ dài bất kỳ thành một chuỗi ký tự có độ dài cố định. Chuỗi ký tự có độ dài cố định này được gọi là giá trị băm. Giá trị băm có thể được sử dụng để kiểm tra tính toàn vẹn của dữ liệu, xác thực người dùng và tạo chữ ký số.

6.2. Các loại hàm băm

Có hai loại chính của hàm băm: hàm băm mật mã và hàm băm không mật mã.

- Hàm băm mật mã được thiết kế để đáp ứng các yêu cầu bảo mật cao. Chúng phải có các tính chất sau:
 - Khả năng chống xung đột: Điều này có nghĩa là không thể tìm thấy hai chuỗi ký tự đầu vào khác nhau có cùng giá trị băm.
 - Khả năng chống đảo ngược: Điều này có nghĩa là không thể tính ngược lại chuỗi ký tự đầu vào từ giá trị băm của nó.
- Hàm băm không mật mã không được thiết kế để đáp ứng các yêu cầu bảo mật cao. Chúng thường có kích thước nhỏ hơn và nhanh hơn so với hàm băm mật mã.

6.3. Cách hoạt động của hàm băm

- Phân chia dữ liệu đầu vào thành các phần nhỏ:

Đầu tiên, dữ liệu đầu vào được phân chia thành các phần nhỏ có độ dài cố định. Độ dài của các phần nhỏ này thường là 32 bit hoặc 64 bit.

- Thực hiện các phép toán toán học trên các phần nhỏ:

Các phần nhỏ sau đó được xử lý bằng các phép toán toán học. Các phép toán toán học này thường được thiết kế để tạo ra một giá trị băm duy nhất cho mỗi chuỗi ký tự đầu vào.

- Kết hợp các giá trị băm của các phần nhỏ:

Cuối cùng, các giá trị băm của các phần nhỏ được kết hợp lại để tạo thành giá trị băm cuối cùng.

6.4. Một số hàm băm phổ biến

- MD5: MD5 là một hàm băm mật mã 128 bit được phát triển bởi Rivest, Shamir và Adleman.
- SHA-1: SHA-1 là một hàm băm mật mã 160 bit được phát triển bởi Cơ quan An ninh Quốc gia Hoa Kỳ.
- SHA-2: SHA-2 là một họ hàm băm mật mã gồm các hàm 224, 256, 384 và 512 bit.

Chương 2: PHÂN TÍCH THIẾT KẾ

1. Ý tưởng thiết kế

Với ý tưởng thiết kế, nhóm lên ý tưởng với ba bước sau:

- Bước 1: Cài đặt thuật toán RSA: Sử dụng thư viện mã hóa RSA trong ngôn ngữ lập trình Python để thực hiện các phép toán RSA như tạo khóa, mã hóa, giải mã và tạo chữ ký số điện tử.
- Bước 2: Tạo GUI (Graphical User Interface - Giao diện người dùng đồ họa) để demo sản phẩm minh họa
- Bước 3: Chạy thử sản phẩm

2. Quy trình thực hiện

2.1. Cài đặt thuật toán RSA

Tạo các hàm để tạo khóa, mã hoá, giải mã và tạo chữ ký số điện tử

2.2.1. Mô tả chi tiết về thuật toán RSA

Thuật toán RSA (Rivest-Shamir-Adleman) là một thuật toán mã hoá khóa công khai được sử dụng rộng rãi trong bảo mật thông tin. Nó dựa trên việc tính toán các phép toán số lớn, đặc biệt là phân tích số nguyên tố. Các bước chính của thuật toán RSA bao gồm:

- Lựa chọn hai số nguyên tố lớn ngẫu nhiên p và q .
- Tính $n = p * q$, gọi là module.
- Tính hàm số Euler của n , $\varphi(n) = (p - 1) * (q - 1)$.
- Lựa chọn một số nguyên e sao cho $1 < e < \varphi(n)$ và e là số nguyên tố cùng nhau với $\varphi(n)$.
- Tìm số nguyên d sao cho $(d * e) \equiv 1 \pmod{\varphi(n)}$, tức là $d * e$ chia hết cho $\varphi(n)$ khi chia lấy dư.
- Khóa công khai là cặp (e, n) và khóa bí mật là cặp (d, n) .

2.2.2. Phân tích các bước để tạo khóa công khai và khóa bí mật:

- a. Lựa chọn hai số nguyên tố lớn p và q :

Bước đầu tiên của quá trình tạo khóa RSA là lựa chọn hai số nguyên tố lớn ngẫu nhiên, thường có độ dài tương đối như nhau.

Điều quan trọng là hai số nguyên tố này phải được giữ bí mật và không được tiết lộ cho bất kỳ ai, bởi vì tính bảo mật của thuật toán RSA dựa trên sự khó khăn trong việc phân tích ngược số nguyên tố lớn.

b. Tính $n = p * q$:

Sau khi đã chọn hai số nguyên tố p và q , tính tích của chúng để thu được giá trị n .

Giá trị n sẽ là module trong quá trình mã hoá và giải mã.

c. Tính hàm số Euler của n , $\varphi(n) = (p - 1) * (q - 1)$:

Hàm số Euler (phi function) của một số nguyên dương n là số nguyên tố cùng nhau với n và nhỏ hơn n .

Đối với RSA, tính giá trị của hàm số Euler của n , ký hiệu là $\varphi(n)$, bằng tích của $(p - 1)$ và $(q - 1)$.

d. Lựa chọn một số nguyên e :

Bước tiếp theo là lựa chọn một số nguyên e , thường là một số nguyên tố, sao cho $1 < e < \varphi(n)$ và e là số nguyên tố cùng nhau với $\varphi(n)$.

Số nguyên tố cùng nhau (coprime) có nghĩa là chúng không có ước chung lớn hơn 1, trừ 1.

e. Tính số nguyên d sao cho $(d * e) \equiv 1 \pmod{\varphi(n)}$:

Bước cuối cùng trong quá trình tạo khóa là tìm một số nguyên d sao cho tích của d và e khi chia lấy dư cho $\varphi(n)$ sẽ cho kết quả là 1.

Điều này có thể thực hiện bằng cách sử dụng thuật toán Euclid mở rộng.

f. Khóa công khai và khóa bí mật:

Khóa công khai là cặp giá trị (e, n) .

Khóa bí mật là cặp giá trị (d, n) .

Quá trình tạo khóa RSA yêu cầu sự chọn lựa cẩn thận của các số nguyên tố lớn và bước tính toán phức tạp để đảm bảo tính bảo mật của hệ mã hoá.

2.2.3. Xem xét các yếu tố quan trọng trong thiết kế RSA

Thuật toán phép tính số lớn: RSA sử dụng các phép tính số lớn như lũy thừa và chia lấy dư trên các số nguyên lớn để thực hiện mã hóa và giải mã. Các phép tính này đảm bảo tính toán hiệu quả và chính xác trong quá trình mã hóa và giải mã thông điệp.

Quá trình sinh số nguyên tố: RSA yêu cầu việc tìm kiếm và lựa chọn hai số nguyên tố lớn để tạo khóa. Điều này đảm bảo tính bảo mật của thuật toán. Quá trình sinh số nguyên tố phải đảm bảo tính ngẫu nhiên và khó khăn trong việc phân tích ngược, điều này đảm bảo rằng việc tìm ra khóa bí mật từ khóa công khai là không thực tế.

Bảo mật khóa: RSA yêu cầu bảo mật khóa bí mật (d , n) để đảm bảo chỉ người được ủy quyền mới có thể giải mã thông điệp. Việc bảo mật khóa bí mật là rất quan trọng để ngăn chặn việc xâm nhập và truy cập trái phép vào thông điệp đã được mã hoá.

Độ dài khóa: Độ dài của các số nguyên p , q và e trong thuật toán RSA có tác động trực tiếp đến độ bảo mật của hệ mã hoá. Độ dài khóa phải đủ lớn để chống lại các phương pháp tấn công thông qua phân tích số nguyên tố hoặc tìm kiếm không gian khóa.

Hiệu suất tính toán: Hiệu suất tính toán để thực hiện mã hóa và giải mã trong RSA là một yếu tố quan trọng, đặc biệt khi xử lý các thông điệp lớn hoặc ứng dụng thời gian thực. Cần xem xét và tối ưu hóa thuật toán để đạt được hiệu suất tốt nhất mà không làm giảm tính bảo mật của hệ mã hoá.

Bảo mật hàm băm: RSA yêu cầu sử dụng hàm băm để tạo chữ ký số điện tử và đảm bảo tính toàn vẹn của thông điệp. Việc chọn một hàm băm an toàn và chống va đập là quan trọng để ngăn chặn việc tạo ra các chữ ký giả mạo.

2.2.4. Tạo chữ ký

Bước 1: Người gửi chuyển đổi thông điệp thành một số nguyên m .

Bước 2: Sử dụng hàm băm an toàn như SHA-1 để tạo giá trị băm của thông điệp:

$$\text{hash_m} = \text{SHA-1}(m).$$

Bước 3: Sử dụng khóa bí mật (d , n) để mã hóa giá trị băm:

$$c \equiv \text{hash_m}^d \pmod{n}. \text{ Số nguyên } c \text{ được tạo thành chữ ký số điện tử.}$$

2.2.5. Xác minh chữ ký

Bước 1: Người nhận nhận được thông điệp và chữ ký số điện tử.

Bước 2: Sử dụng khóa công khai (e, n) của người gửi để giải mã chữ ký:

$\text{hash_c} \equiv c^e \pmod{n}$. Số nguyên hash_c được nhận.

Bước 3: Sử dụng hàm băm an toàn như SHA-1 tạo giá trị băm của thông điệp ban đầu:

$\text{hash_m} = \text{SHA-1}(m)$.

Bước 4: So sánh giá trị băm hash_m với giá trị băm hash_c để xác minh tính toàn vẹn của chữ ký. Nếu hai giá trị này khớp nhau, chữ ký được xác minh.

2.2.6. Xem xét các yếu tố quan trọng trong thiết kế chữ ký số điện tử:

Bảo mật khóa: Trong thiết kế chữ ký số điện tử sử dụng RSA, cần đảm bảo bảo mật của cặp khóa: khóa công khai và khóa bí mật. Khóa bí mật phải được bảo vệ cẩn thận và chỉ được sử dụng bởi chủ khóa. Cần áp dụng các biện pháp bảo mật mạnh như lưu trữ an toàn và quản lý khóa chặt chẽ.

Hàm băm SHA-1: Sử dụng hàm băm an toàn như SHA-1 để tạo giá trị băm của thông điệp. Hàm băm an toàn đảm bảo tính không thể đoán trước và không thể tái tạo của giá trị băm, giúp bảo vệ tính toàn vẹn của thông điệp.

Thuật toán SHA-1 nhận thông điệp ở đầu vào có chiều dài $k < 2^{64}$ bit, thực hiện xử lý và đưa ra thông điệp thu gọn (message digest) có chiều dài cố định 160 bits. Quá trình tính toán cũng thực hiện theo từng khối 512bits, nhưng bộ đệm xử lý dùng 5 thanh ghi 32-bits. Thuật toán này chạy tốt với các bộ vi xử lý có cấu trúc 32 bits.

Quản lý khóa: Quản lý khóa trong thiết kế chữ ký số điện tử RSA rất quan trọng. Cần thiết lập quy trình và hệ thống quản lý khóa an toàn để đảm bảo rằng khóa không bị rò rỉ hoặc sử dụng sai mục đích. Đồng thời, cần thực hiện các biện pháp bảo mật mạnh như mã hóa khóa, lưu trữ an toàn và kiểm soát truy cập vào khóa.

Cơ sở hạ tầng công khai: Trong thiết kế chữ ký số điện tử RSA, cơ sở hạ tầng công khai (PKI) là một yếu tố quan trọng để xác minh tính toàn vẹn và xác thực của chữ ký. Cần triển khai và quản lý PKI một cách chính xác, bao gồm cấp phát chứng chỉ số, quản lý danh sách từ chối và phân phối khóa công khai.

Chuẩn quốc tế: Trong thiết kế chữ ký số điện tử RSA, cần tuân thủ các chuẩn quốc tế liên quan đến RSA và chữ ký số. Ví dụ, chuẩn RSA PKCS#1 định nghĩa các thuật toán và quy trình cho mã hóa RSA và chữ ký số. Tuân thủ các chuẩn này đảm bảo tính tương thích và đáng tin cậy của chữ ký số điện tử.

2.2.7. Cài đặt bằng ngôn ngữ lập trình python

```
def is_prime(num):  
    """Kiểm tra xem một số có phải là số nguyên tố hay không."""  
    if num < 2:  
        return False  
    for i in range(2, int(num ** 0.5) + 1):  
        if num % i == 0:  
            return False  
    return True
```

- Hàm is_prime() được sử dụng để kiểm tra xem một số có phải là số nguyên tố hay không.
- Hàm này có hai tham số:
 - num: Số cần kiểm tra
 - return: Giá trị trả về của hàm, là True nếu num là số nguyên tố, False nếu num không phải là số nguyên tố
- Hàm hoạt động như sau:
 - Đầu tiên, hàm kiểm tra xem num có nhỏ hơn 2 hay không. Nếu num nhỏ hơn 2, thì num không thể là số nguyên tố. Trong trường hợp này, hàm trả về False.
 - Nếu num lớn hơn hoặc bằng 2, thì hàm sẽ duyệt qua các số từ 2 đến căn bậc hai của num.
 - Nếu num chia hết cho một trong các số trong vòng lặp, thì num không thể là số nguyên tố. Trong trường hợp này, hàm trả về False.
 - Nếu num không chia hết cho bất kỳ số nào trong vòng lặp, thì num là số nguyên tố. Trong trường hợp này, hàm trả về True

```
def modinv(a, m):  
    """Tìm nghịch đảo modulo."""  
    m0, x0, x1 = m, 0, 1  
    while a > 1:
```

```

q = a // m
m, a = a % m, m
x0, x1 = x1 - q * x0, x0
return x1 + m0 if x1 < 0 else x1

```

- Hàm `modinv()` để tìm nghịch đảo modulo của một số a modulo một số m .
 - Nguyên lý hoạt động của hàm này dựa trên phương pháp Euclide mở rộng. Phương pháp này sử dụng định lý Euclide, theo đó nếu a và m là hai số nguyên dương không chia hết cho nhau, thì tồn tại hai số nguyên x và y sao cho $ax + my = 1$.
 - Hàm `modinv()` bắt đầu bằng việc khởi tạo hai biến `m0` và `x1` với giá trị là m và 1 . Sau đó, hàm sẽ lặp lại cho đến khi a bằng 1 . Trong mỗi vòng lặp, hàm sẽ thực hiện các phép tính sau:
 - Tính $q = a // m$.
 - Cập nhật m và a thành $a \% m$ và m .
 - Cập nhật $x0$ và $x1$ thành $x1 - q * x0$ và $x0$.
 - Khi a bằng 1 , thì $x1$ chính là nghịch đảo modulo của a modulo m . Tuy nhiên, nếu $x1$ nhỏ hơn 0 , thì cần phải cộng thêm `m0` để đảm bảo rằng $x1$ là số nguyên dương.

```

def generate_large_prime():
    """Sinh một số nguyên tố lớn."""
    while True:
        num = random.randint(10**14, 10**15)
        if is_prime(num):
            return num

```

- Hàm `generate_large_prime()` có chức năng sinh ra một số nguyên tố lớn. Hàm này sử dụng một vòng lặp `while True` để lặp đi lặp lại cho đến khi tìm thấy một số nguyên tố. Trong mỗi lần lặp, hàm sẽ sinh ra một số nguyên tố ngẫu nhiên trong khoảng từ 10^{14} đến 10^{15} . Nếu số nguyên tố ngẫu nhiên này là số nguyên tố, thì hàm sẽ trả về số đó.
- Hàm này hoạt động theo nguyên tắc sau:
 - Khởi tạo một vòng lặp `while True`.
 - Trong mỗi lần lặp, sinh ra một số nguyên tố ngẫu nhiên.

- Kiểm tra xem số nguyên tố ngẫu nhiên đó có phải là số nguyên tố hay không.
- Nếu là số nguyên tố, thì trả về số đó.
- Nếu không phải là số nguyên tố, thì lặp lại vòng lặp.

```
def generate_keypair():
    """Sinh khoá công khai và bí mật."""
    p = generate_large_prime()
    q = generate_large_prime()
    n = p * q
    phi = (p - 1) * (q - 1)

    # Chọn số e
    e = random.randint(2, phi - 1)
    while gcd(e, phi) != 1:
        e = random.randint(2, phi - 1)

    # Sử dụng thuật toán Euclide mở rộng để tính số d
    d = modinv(e, phi)

    return ((n, e), (n, d))
```

- Đoạn code trên là một hàm sinh cặp khóa công khai và bí mật theo thuật toán RSA. Thuật toán RSA là một thuật toán mật mã hóa khóa công khai dựa trên bài toán khó phân tích số nguyên tố.
- Hàm này có hai tham số đầu vào là p và q , là hai số nguyên tố lớn ngẫu nhiên. Từ p và q , hàm tính ra n , là tích của p và q , và ϕ , là hàm phi của n .
- Tiếp theo, hàm chọn e , là một số nguyên tố nhỏ hơn ϕ . Hàm sử dụng vòng lặp `while` để đảm bảo rằng e và ϕ là ước số nguyên tố của nhau.
- Cuối cùng, hàm sử dụng thuật toán Euclide mở rộng để tính d , là số nghịch đảo của e modulo ϕ .
- Hàm này trả về một cặp khóa, bao gồm khóa công khai $((n, e))$ và khóa bí mật $((n, d))$. Khóa công khai (n, e) có thể được chia sẻ với tất cả mọi người. Khóa bí mật (n, d) chỉ được giữ bí mật bởi chủ sở hữu khóa.
- Khóa công khai được sử dụng để mã hóa dữ liệu. Khóa bí mật được sử dụng để giải mã dữ liệu đã được mã hóa bằng khóa công khai.

```
def sign(message, private_key):
    """Ký bản tin."""
    n, d = private_key
    signature = pow(message, d, n)
    return signature
```

- Dòng `def sign(message, private_key):`: khai báo một hàm có tên `sign` nhận hai tham số `message` (bản tin) và `private_key` (khóa riêng).
- Dòng `"""Ký bản tin."""` là chú thích, giải thích mục đích của hàm.
- Dòng `n, d = private_key` giải nén khóa riêng thành hai thành phần `n` (mô đun) và `d` (thành phần bí mật).
- Dòng `signature = pow(message, d, n)` là dòng chính thực hiện việc ký bản tin. Dòng này tính toán tổng mũ $\text{message}^d \bmod n$ và gán kết quả cho biến `signature`. Đây là cách tạo chữ ký RSA cơ bản.
- Dòng cuối cùng `return signature` trả về chữ ký đã được tạo ở dòng 4.

```
def verify(message, signature, public_key):
    """Kiểm tra chữ ký."""
    n, e = public_key
    decrypted_signature = pow(signature, e, n)
    return decrypted_signature == message
```

- Dòng `def verify(message, signature, public_key):`: khai báo một hàm có tên `verify` nhận ba tham số `message` (bản tin), `signature` (chữ ký) và `public_key` (khóa công khai).
- Dòng `"""Kiểm tra chữ ký."""` là chú thích, giải thích mục đích của hàm.
- Dòng `n, e = public_key` giải nén khóa công khai thành hai thành phần `n` (mô đun) và `e` (thành phần công khai).
- Dòng `decrypted_signature = pow(signature, e, n)` là dòng chính thực hiện việc kiểm tra chữ ký. Dòng này tính toán tổng mũ $\text{signature}^e \bmod n$ và gán kết quả cho biến `decrypted_signature`. Đây là cách giải mã chữ ký RSA cơ bản.

- Dòng cuối cùng `return decrypted_signature == message` so sánh `decrypted_signature` với `message` và trả về kết quả. Nếu `decrypted_signature` bằng `message`, thì chữ ký hợp lệ, ngược lại chữ ký không hợp lệ.

```
def decrypt(ciphertext, public_key):
    """Giải mã bản tin."""
    n, e = public_key
    plaintext = pow(ciphertext, e, n)
    return plaintext
```

- Dòng `def decrypt(ciphertext, public_key):` khai báo một hàm có tên `decrypt` nhận hai tham số `ciphertext` (bản tin đã mã hóa) và `public_key` (khóa công khai).
- Dòng `"""Giải mã bản tin."""` là chú thích, giải thích mục đích của hàm.
- Dòng `n, e = public_key` giải nén khóa công khai thành hai thành phần `n` (mô đun) và `e` (thành phần công khai).
- Dòng `plaintext = pow(ciphertext, e, n)` là dòng chính thực hiện việc giải mã bản tin. Dòng này tính toán tổng mũ $\text{ciphertext}^e \bmod n$ và gán kết quả cho biến `plaintext`. Đây là cách giải mã bản tin RSA cơ bản.
- Dòng cuối cùng `return plaintext` trả về bản tin đã giải mã.

```
# Example usage
if __name__ == "__main__":
    # Bước 1: Sinh khóa
    public_key, private_key = generate_keypair()
    print("Public Key:", public_key)
    print("Private Key:", private_key)

    # Bước 2: Ký và kiểm tra chữ ký
    message = random.randint(1, public_key[0] - 1)
    print("Original Message:", message)

    signature = sign(message, private_key)
    print("Signature:", signature)

    verification_result = verify(message, signature, public_key)
    print("Verification Result:", verification_result)

    # Bước 3: Giải mã khi xác thực thành công
    if verification_result:
        decrypted_message = decrypt(signature, public_key)
        print("Decrypted Message:", decrypted_message)
```

– Cách sử dụng thuật toán RSA để thực hiện các thao tác ký, kiểm tra chữ ký và giải mã. Đoạn code sử dụng các thư viện sau:

- random: Thư viện tạo số ngẫu nhiên
- hashlib: Thư viện tạo giá trị băm

Bước 1: Sinh khóa

- Đoạn code trong trên sinh ra cặp khóa công khai và khóa riêng. Khóa công khai được sử dụng để mã hóa bản tin, còn khóa riêng được sử dụng để giải mã bản tin.
- Đoạn code sử dụng hàm generate_keypair() để sinh ra cặp khóa. Hàm này nhận một tham số là độ dài của khóa, và trả về một cặp khóa gồm hai thành phần:

n: Mô đun

e: Thành phần công khai

Bước 2: Ký và kiểm tra chữ ký

- Bước tiếp theo, đoạn code sử dụng khóa riêng để ký một bản tin. Quá trình ký bao gồm các bước sau:

- Nâng bản tin lên lũy thừa d modulo n .
- Trả về kết quả.
- Trong đoạn code, hàm `sign()` thực hiện các bước trên. Hàm này nhận hai tham số là bản tin và khóa riêng.
- Để kiểm tra chữ ký, đoạn code sử dụng khóa công khai. Quá trình kiểm tra chữ ký bao gồm các bước sau:
- Nâng chữ ký lên lũy thừa e modulo n .
- So sánh kết quả với bản tin gốc.
- Trong đoạn code, hàm `verify()` thực hiện các bước trên. Hàm này nhận ba tham số là bản tin, chữ ký và khóa công khai.

Bước 3: Giải mã

- Cuối cùng, đoạn code sử dụng khóa công khai để giải mã một bản tin đã mã hóa. Quá trình giải mã bao gồm các bước sau:
- Nâng bản tin lên lũy thừa e modulo n .
- Trả về kết quả.

Trong đoạn code, hàm `decrypt()` thực hiện các bước trên. Hàm này nhận hai tham số là bản tin đã mã hóa và khóa công khai.

2.2. Cài đặt giao diện người dùng GUI

2.2.1. Cài đặt hàm `setupUi (self, MainWindow):`

a. Framework:

Đoạn code sử dụng framework Qt của Qt Company. Qt là một framework đa nền tảng, được sử dụng để tạo giao diện người dùng đồ họa (GUI) cho các ứng dụng máy tính. Nó cung cấp một bộ công cụ và thư viện phong phú để giúp lập trình viên tạo GUI đẹp mắt và hiệu quả.

b. Mục đích:

Đoạn code tạo giao diện người dùng chính của một ứng dụng liên quan đến việc tạo và xử lý khóa mật mã. Ứng dụng này có ba tính năng chính:

- Tạo khóa mật mã mới
- Mã hóa văn bản bằng khóa mật mã
- Giải mã văn bản được mã hóa bằng khóa mật mã

c. Cấu trúc:

Đoạn code được chia thành hai phần chính:

- Hàm `setupUi(self, MainWindow)`: Thiết lập giao diện người dùng chính của ứng dụng.
- Các thành phần GUI chính: Bao gồm ba nhóm đối tượng: `groupBox`, `sinh_khoa_title`, `groupBox_2`, và `groupBox_3`. Mỗi nhóm chứa các thành phần liên quan đến một tính năng cụ thể của ứng dụng.

d. Các thành phần GUI chính

❖ Thành phần `groupBox`

Thành phần `groupBox` là một nhóm chứa các thành phần GUI khác. Nó được sử dụng để nhóm các thành phần có liên quan với nhau về mặt chức năng. Trong ứng dụng này, `groupBox` chứa các thành phần sau:

- `label`: Nhãn văn bản "Tạo khóa mật mã".
- `label_2`: Nhãn văn bản "p".
- `label_3`: Nhãn văn bản "q".
- `label_4`: Nhãn văn bản "Khóa bí mật".
- `label_5`: Nhãn văn bản "Khóa công khai".
- `ngau_nhien`: Nút "Tạo ngẫu nhiên".
- `sinh_khoa`: Nút "Tạo khóa".
- `khoa_bi_mat`: Trường nhập liệu để nhập khóa bí mật.
- `khoa_cong_khai`: Trường nhập liệu để nhập khóa công khai.

Các thành phần này được sử dụng để tạo khóa mật mã mới. Nút "Tạo ngẫu nhiên" sẽ tạo ngẫu nhiên hai số nguyên tố p và q . Sau đó, các số này được sử dụng để tính toán khóa

bí mật và khóa công khai. Nút "Tạo khóa" sẽ xác nhận các giá trị p và q đã nhập và tạo khóa mật mã mới.

Thành phần `sinh_khoa_title`: Thành phần `sinh_khoa_title` là một nhóm chứa các thành phần liên quan đến việc tạo khóa mật mã. Nó được sử dụng để làm nổi bật các thành phần này.

❖ Thành phần `groupBox_2`

Thành phần `groupBox_2` là một nhóm chứa các thành phần liên quan đến việc mã hóa văn bản. Nó chứa các thành phần sau:

- `label_6`: Nhãn văn bản "Văn bản gốc".
- `label_7`: Nhãn văn bản "Văn bản đã mã hóa".
- `chon1`: Nút "Chọn văn bản".
- `xuat_kq1`: Nút "Xuất kết quả".
- `ky_van_ban`: Nút "Ký văn bản".
- `van_ban1`: Trường nhập liệu để nhập văn bản gốc.
- `ket_qual`: Trường nhập liệu để hiển thị văn bản đã mã hóa.

Các thành phần này được sử dụng để mã hóa văn bản bằng khóa mật mã. Nút "Chọn văn bản" sẽ mở một hộp thoại cho phép người dùng chọn văn bản gốc. Nút "Ký văn bản" sẽ sử dụng khóa mật mã để mã hóa văn bản gốc. Văn bản đã mã hóa sẽ được hiển thị trong trường nhập liệu `ket_qual`.

❖ Thành phần `groupBox_3`

Thành phần `groupBox_3` là một nhóm chứa các thành phần liên quan đến việc giải mã văn bản. Nó chứa các thành phần sau:

- `label_8`: Nhãn văn bản "Văn bản đã mã hóa".
- `label_9`: Nhãn văn bản "Văn bản gốc".
- `chon2`: Nút "Chọn văn bản".
- `xuat_kq2`: Nút "Xuất kết quả".
- `xac_thuc`: Nút "Xác thực".

- van_ban2: Trường nhập liệu để nhập văn bản đã mã hóa.
- ket_qua2: Trường nhập liệu để hiển thị văn bản gốc.

Các thành phần này được sử dụng để giải mã văn bản được mã hóa bằng khóa mật mã. Nút "Chọn văn bản" sẽ mở một hộp thoại cho phép người dùng chọn văn bản đã mã hóa. Nút "Xác thực" sẽ sử dụng khóa công khai để xác thực văn bản đã mã hóa. Nếu văn bản đã mã hóa hợp lệ, thì văn bản gốc sẽ được hiển thị trong trường nhập liệu ket_qua2.

2.2.2. Cài đặt hàm *retranslateUi (self, MainWindow)*

a. Mục đích:

Hàm này có nhiệm vụ dịch các văn bản và chuỗi ký tự trong giao diện ứng dụng sang ngôn ngữ khác phù hợp với cài đặt ngôn ngữ của hệ thống.

Nó đảm bảo giao diện ứng dụng hiển thị chính xác ngôn ngữ cho người dùng.

b. Cách hoạt động:

- Lấy chức năng dịch từ QApplication: Hàm lấy đối tượng `_translate` từ `QtCore.QCoreApplication.translate` để sử dụng cho việc dịch văn bản.
- Dịch tiêu đề cửa sổ: Đặt tiêu đề cửa sổ chính thành "MainWindow" bằng cách gọi `MainWindow.setWindowTitle(_translate("MainWindow", "MainWindow"))`
- Dịch tiêu đề và định dạng các nhóm đối tượng: Dịch tiêu đề của các nhóm đối tượng (`groupBox`, `sinh_khoa_title`, `groupBox_2`, `groupBox_3`) sang ngôn ngữ phù hợp.
- Đặt định dạng viền, màu sắc, độ cong góc và khoảng cách cho các nhóm đối tượng.
- Dịch văn bản của các nhãn và nút: Dịch văn bản của các nhãn (`label`, `label_2`, ..., `label_9`) và nút (`ngau_nhien`, `sinh_khoa`, ..., `xac_thuc`) sang ngôn ngữ phù hợp.

2.3. Kết nối giao diện và thuật toán

2.2.1. Cấu trúc tổng quan

- Import các thư viện cần thiết:
 - `sys`: Truy cập các đối số dòng lệnh và thoát ứng dụng.

- PyQt6.QtWidgets: Tạo các thành phần GUI và tương tác với hệ thống.
- Class RSAApp:
 - Lớp chính của ứng dụng, kế thừa từ QMainWindow để tạo cửa sổ chính.
 - Chứa các phương thức để thiết lập giao diện, kết nối các sự kiện, và thực hiện các chức năng của ứng dụng.
- Hàm main():
 - Tạo đối tượng ứng dụng RSAApp.
 - Hiển thị cửa sổ chính của ứng dụng.
 - Chạy vòng lặp sự kiện chính của ứng dụng.

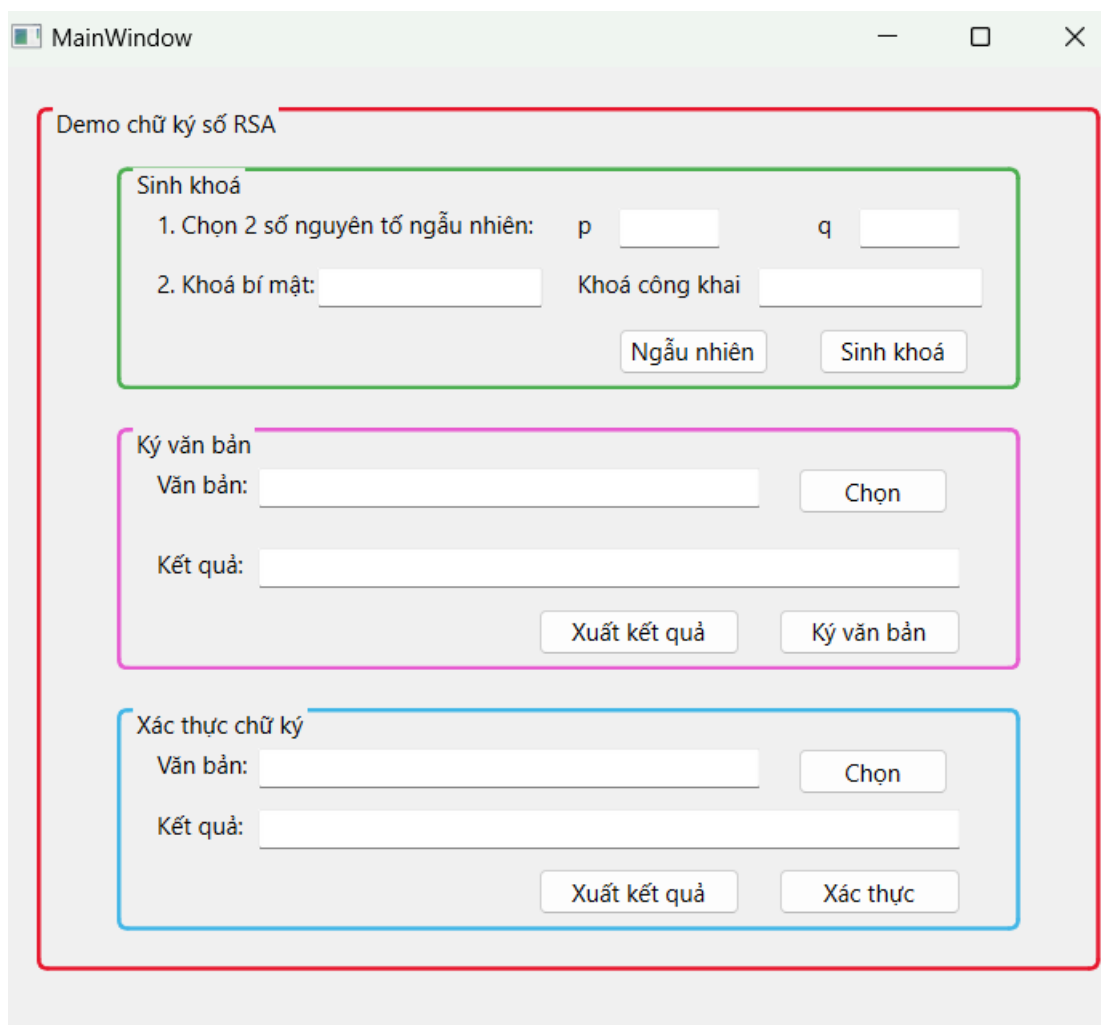
2.2.2. Các phương thức và chức năng chính

1. `__init__(self)`:
 - Khởi tạo lớp RSAApp, thiết lập giao diện người dùng và kết nối các sự kiện với các hàm xử lý.
2. `generate_random_primes(self)`:
 - Tạo hai số nguyên tố ngẫu nhiên p và q, hiển thị chúng trên giao diện.
3. `generate_keypair(self)`:
 - Gọi hàm `generate_keypair()` (được định nghĩa bên ngoài class này) để tạo ra cặp khóa công khai và khóa bí mật.
 - Hiển thị khóa bí mật và khóa công khai trên giao diện.
4. `load_text_file1(self)`:
 - Mở hộp thoại để chọn file văn bản cần ký.
 - Đọc nội dung file và hiển thị trong trường nhập liệu tương ứng.
5. `sign_text(self)`:
 - Lấy văn bản cần ký từ trường nhập liệu.
 - Gọi hàm `sign()` (được định nghĩa bên ngoài class này) để ký văn bản bằng khóa bí mật.

- Hiện thị chữ ký số trên giao diện.
- 6. `export_signed_text(self)`:
 - Lưu chữ ký số vào một file văn bản.
- 7. `load_text_file2(self)`:
 - Mở hộp thoại để chọn file văn bản chứa chữ ký số cần xác thực.
 - Đọc nội dung file và hiển thị trong trường nhập liệu tương ứng.
- 8. `verify_signature(self)`:
 - Lấy văn bản gốc và chữ ký số từ các trường nhập liệu.
 - Gọi hàm `verify()` (được định nghĩa bên ngoài class này) để xác thực chữ ký số bằng khóa công khai.
 - Hiện thị kết quả xác thực và giải mã văn bản nếu xác thực thành công.
- 9. `export_decrypted_text(self)`:
 - Lưu văn bản đã giải mã vào một file văn bản.

Chương 3: KẾT QUẢ THỰC HIỆN

1. Giao diện chương trình demo

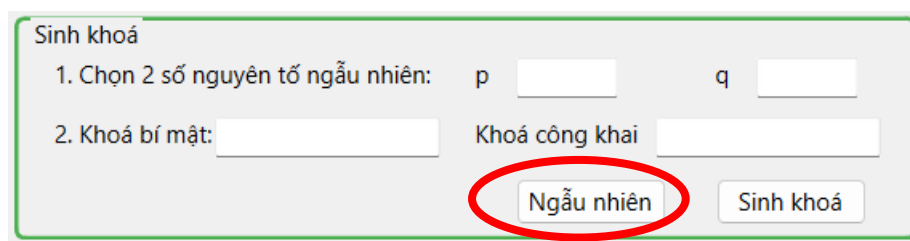


The screenshot shows a window titled "MainWindow" with a red border. Inside, the title "Demo chữ ký số RSA" is at the top. The interface is divided into three main sections, each with a colored border: "Sinh khoá" (green), "Ký văn bản" (pink), and "Xác thực chữ ký" (blue). The "Sinh khoá" section contains two input fields for prime numbers p and q , two input fields for private and public keys, and two buttons: "Ngẫu nhiên" and "Sinh khoá". The "Ký văn bản" section has a text input field, a "Chọn" button, a result output field, and two buttons: "Xuất kết quả" and "Ký văn bản". The "Xác thực chữ ký" section also has a text input field, a "Chọn" button, a result output field, and two buttons: "Xuất kết quả" and "Xác thực".

Hình 3.1: Giao diện chương trình demo

2. Cài đặt và triển khai

- Bước 1: Bấm vào nút ngẫu nhiên để chọn 2 số nguyên tố ngẫu nhiên



This image is a close-up of the "Sinh khoá" section from the previous screenshot. It shows the input fields for p and q , the private and public key fields, and the "Ngẫu nhiên" and "Sinh khoá" buttons. The "Ngẫu nhiên" button is circled in red, indicating it is the focus of the current step.

Hình 3.2: Tạo 2 số nguyên tố

- Bước 2: bấm nút sinh khoá để tạo khoá bí mật và khoá công khai tương ứng

Sinh khoá

1. Chọn 2 số nguyên tố ngẫu nhiên: p q

2. Khoá bí mật: Khoá công khai:

Ngẫu nhiên Sinh khoá

Hình 3.3: Tạo khoá công khai và khoá bí mật tương ứng

- Bước 3: Nhập văn bản. Người dùng có thể nhập thủ công hoặc bấm nút “chọn” để lấy ngẫu nhiên một văn bản (file.txt) sẵn có trên máy tính của mình.

Ký văn bản

Văn bản: Chọn

Kết quả:

Xuất kết quả Ký văn bản

Hình 3.4: Nhập văn bản để ký

- Bước 4: Ký văn bản. Người dùng bấm vào nút ký văn bản để có kết quả. Kết quả của văn bản đã ký là một dãy số

Ký văn bản

Văn bản: Chọn

Kết quả:

Xuất kết quả Ký văn bản

Hình 3.5: Kết quả sau khi ký văn bản

- Bước 5: Xuất kết quả. Lưu file

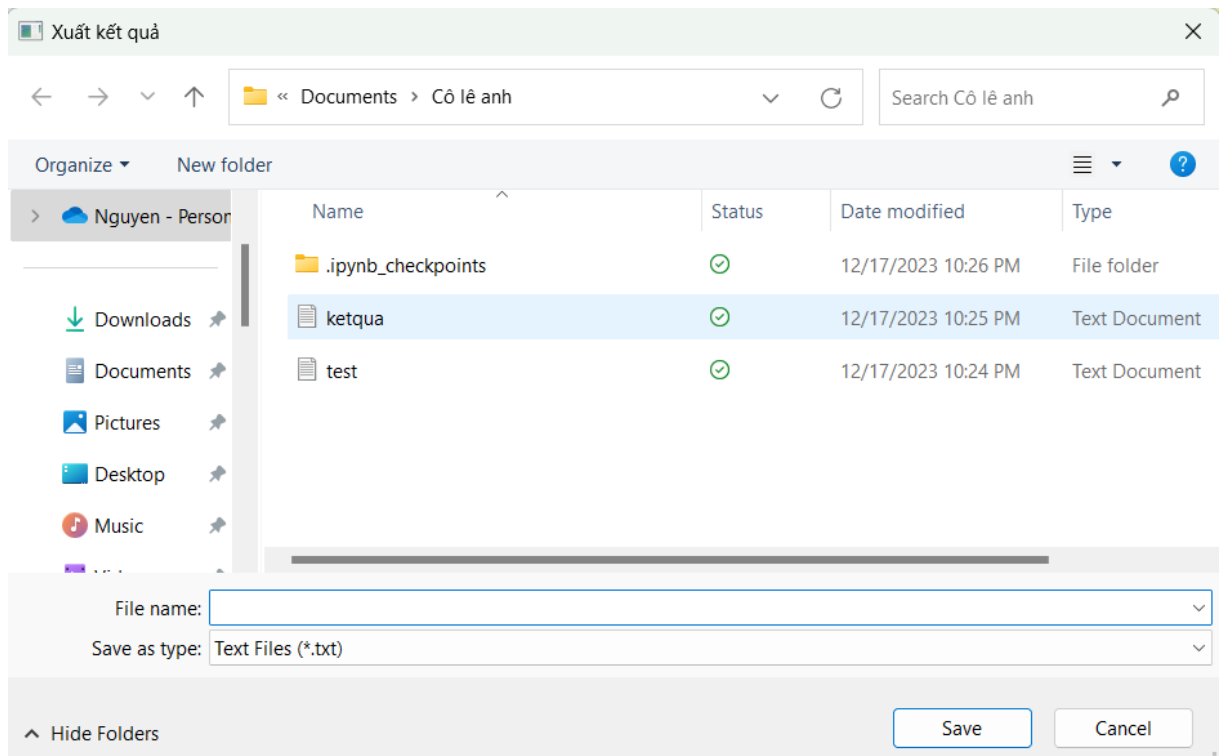
Ký văn bản

Văn bản: Chọn

Kết quả:

Xuất kết quả Ký văn bản

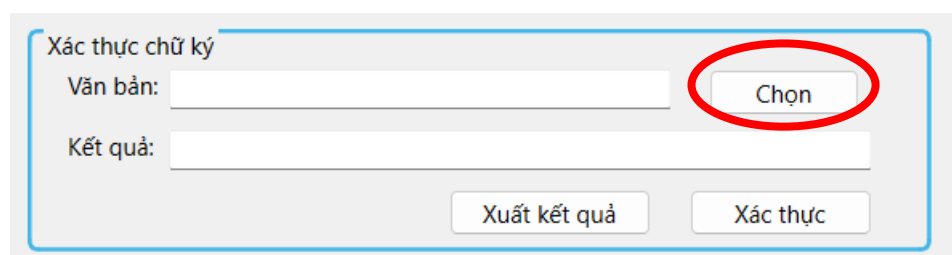
Hình 3.6: Xuất kết quả và lưu file vào máy tính



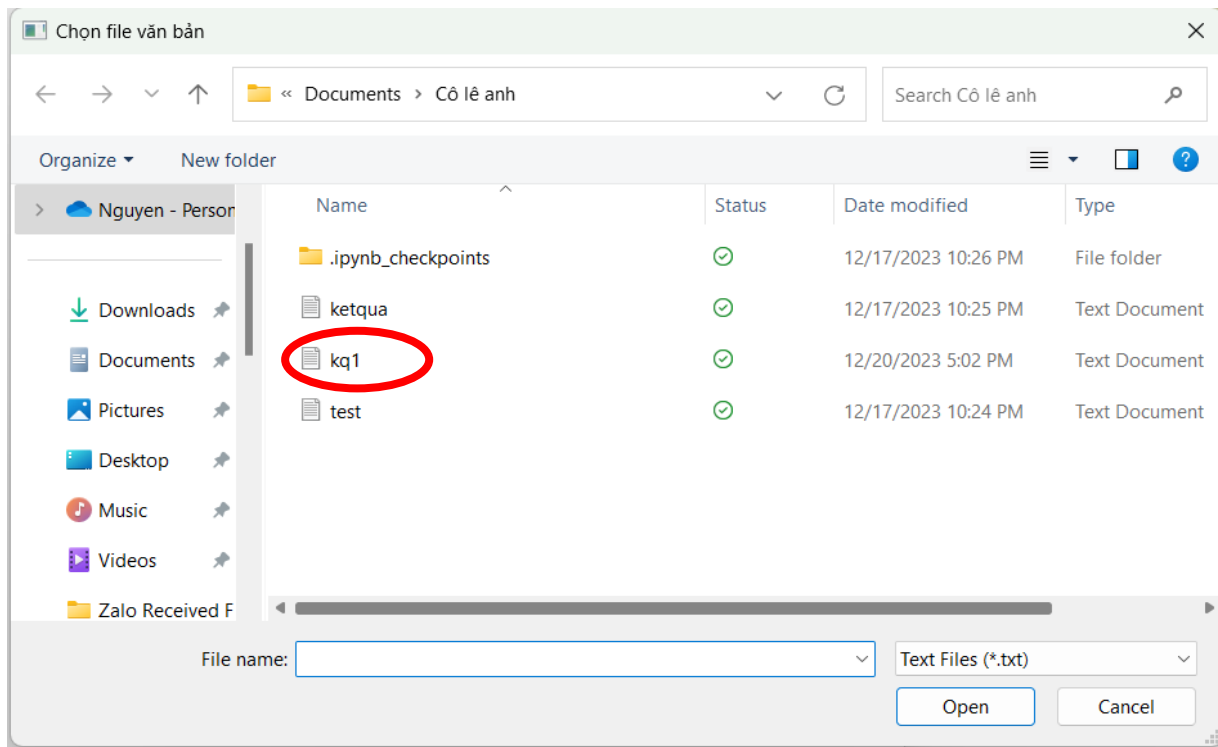
Hình 3.7: Cửa sổ xuất kết quả

Sau khi bấm vào xuất kết quả sẽ có một cửa sổ mới hiện ra tên là xuất kết quả. Tại trường File name đặt tên phù hợp cho kết quả. File sẽ được lưu dưới dạng file txt

- Bước 6: xác thực chữ ký. Bấm vào nút chọn để nhập văn bản. Văn bản này chính là kết quả của mục ký văn bản nên trên.



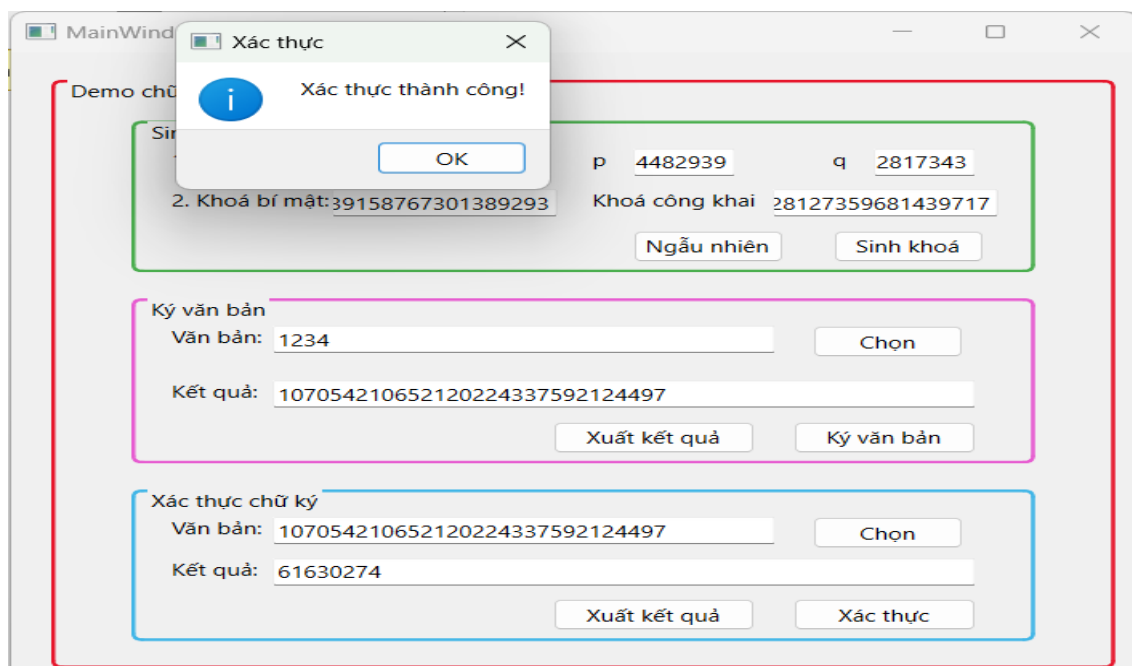
Hình 3.8: Nhập văn bản xác thực



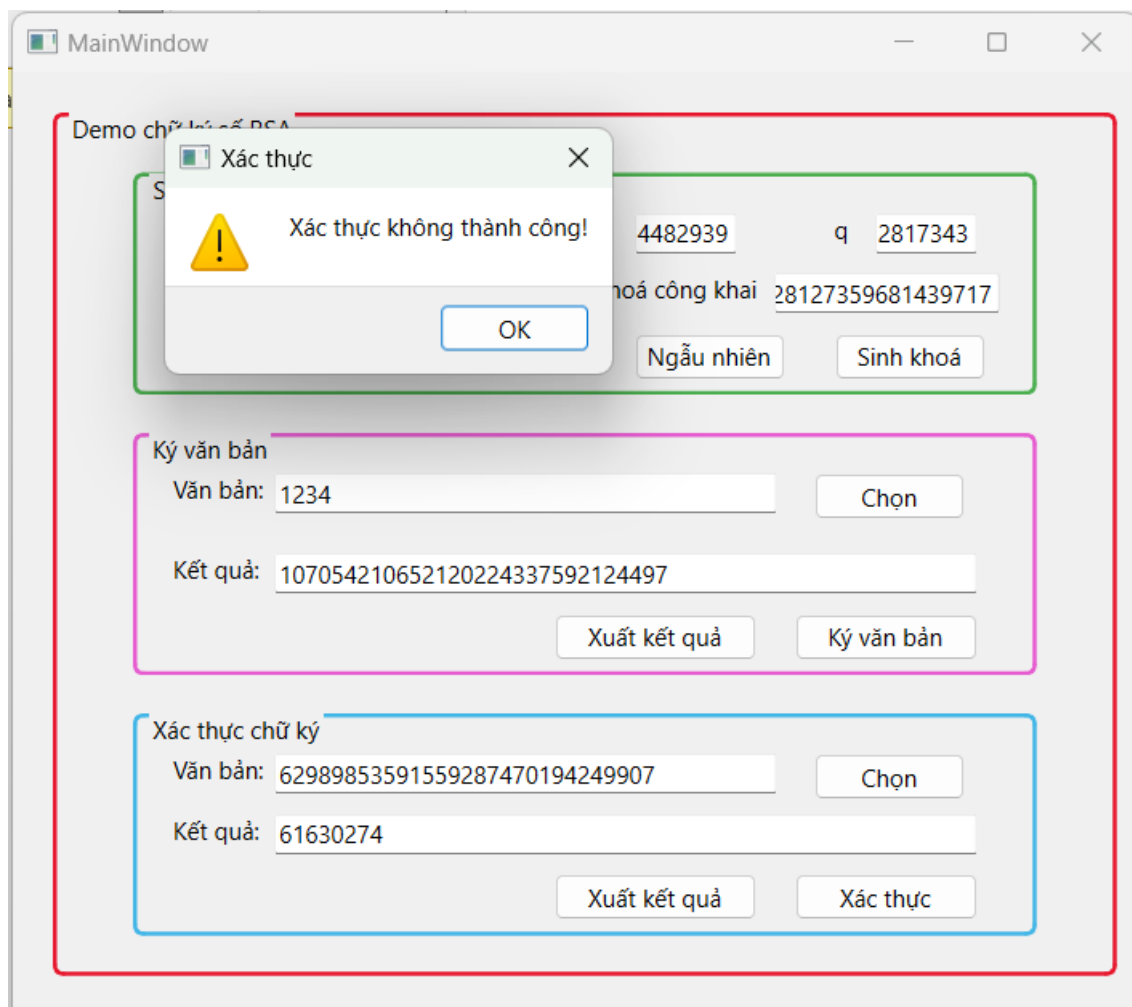
Hình 3.9: Cửa sổ chọn file xác thực

Sau khi bấm vào nút chọn, một cửa sổ mới tên là “Chọn file văn bản” sẽ xuất hiện. Ta chọn đúng file kết quả mình vừa đặt.

- Bước 7: Bấm xác thực chữ ký. Sẽ có cửa sổ xác thực chữ ký hiện lên. Nếu thành công sẽ có thông báo “Xác thực chữ ký thành công!”. Nếu bạn chọn sai file văn bản sẽ có cửa sổ xác thực chữ ký không thành công hiện lên.



Hình 3.10: Xác thực thành công



Hình 3.11: Xác thực không thành công

Kết quả của ký văn bản và xác thực chữ ký lại khác nhau vì các thao tác này sử dụng các khóa khác nhau.

Ký văn bản sử dụng khóa bí mật để tạo ra một chuỗi số nguyên ngẫu nhiên, gọi là chữ ký số. Chữ ký số này được gắn vào văn bản gốc để tạo thành văn bản được ký.

Xác thực chữ ký sử dụng khóa công khai để kiểm tra xem chữ ký số có khớp với văn bản gốc hay không. Nếu khớp, thì chữ ký được coi là hợp lệ và văn bản gốc được coi là không bị thay đổi.

Do đó, kết quả của ký văn bản và xác thực chữ ký sẽ khác nhau vì chúng sử dụng các khóa khác nhau.

Trong ứng dụng RSA được đề cập ở trên, khóa bí mật được sử dụng để ký văn bản, còn khóa công khai được sử dụng để xác thực chữ ký. Do đó, kết quả của ký văn bản và xác thực chữ ký sẽ khác nhau.

3. Kết quả đạt được

Nhóm chúng em đã nắm được các khái niệm cơ bản về mật mã khóa công khai, bao gồm:

- Số nguyên tố
- Module
- Mật mã RSA

Nhóm cũng đã có thể thực hiện các thao tác mật mã RSA cơ bản, bao gồm:

- Tạo số nguyên tố
- Tạo cặp khóa
- Ký văn bản
- Xác thực chữ ký

Nhóm sinh viên đã có thể tạo ra một ứng dụng Python sử dụng framework Qt để thực hiện các thao tác mật mã RSA.

4. Hạn chế

Ứng dụng chỉ sử dụng các số nguyên tố nhỏ để tạo cặp khóa. Điều này có thể làm giảm độ an toàn của hệ thống.

Ứng dụng không sử dụng các kỹ thuật bảo mật bổ sung, chẳng hạn như băm mật khẩu, mã hóa dữ liệu, v.v. Điều này có thể làm tăng nguy cơ tấn công.

Ứng dụng chỉ hỗ trợ ký và xác thực chữ ký cho văn bản. Cần phải bổ sung tính năng hỗ trợ ký và xác thực chữ ký cho các tệp tin, v.v.

PHỤ LỤC

TÀI LIỆU THAM KHẢO

<https://bizflycloud.vn/tin-tuc/rsa-la-gi-20220506153312797.htm>

<https://www.youtube.com/watch?v=qph77bTKJTM>

<https://vietnix.vn/rsa/>

<https://codelearn.io/sharing/hash-la-gi-va-hash-dung-de-lam-gi>

<https://nacis.gov.vn/nghien-cuu-trao-doi/-/view-content/213743/gioi-thieu-ve-mot-so-ham-bam-va-ung-dung-trong-mot-so-san-pham-mat-ma-dan-su>

<https://academy.binance.com/vi/articles/what-is-public-key-cryptography>

<https://www.gate.io/vi/learn/articles/what-is-public-key-cryptography-pkc/311> <https://itnavi.com.vn/blog/rsa-la-gi>

<https://vusta.vn/ung-dung-he-ma-hoa-rsa-vao-bao-mat-he-thong-cong-nghe-thong-tin-p69962.html>