BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC NHA TRANG**

**KHOA CÔNG NGHỆ THÔNG TIN**

****

**BÁO CÁO BÀI TẬP LỚN LẬP TRÌNH PYTHON**

**Tên đề tài: CÀI ĐẶT THUẬT TOÁN MÃ HÓA RSA VÀ HÀM BĂM SHA-256 TRONG PYTHON**

**Sinh viên thực hiện: Võ Văn Thành**

**Mã số sinh viên: 64132201**

**Giảng viên hướng dẫn: TS. Phạm Văn Nam**

**Học kỳ 1, năm học 2024 - 2025**

Khánh Hòa - 2025

BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC NHA TRANG**

**KHOA CÔNG NGHỆ THÔNG TIN**

****

**BÁO CÁO BÀI TẬP LỚN LẬP TRÌNH PYTHON**

**Tên đề tài: CÀI ĐẶT THUẬT TOÁN MÃ HÓA RSA VÀ HÀM BĂM SHA-256 TRONG PYTHON**

**Sinh viên thực hiện: Võ Văn Thành**

**Mã số sinh viên: 64132201**

**Giảng viên hướng dẫn: TS. Phạm Văn Nam**

**Học kỳ 1, năm học 2024 - 2025**

Khánh Hòa - 2025

**MỤC LỤC**

**[DANH MỤC HÌNH 2](#_Toc20602)**

**[DANH MỤC BẢNG 3](#_Toc472)**

**[Chương 1. GIỚI THIỆU ĐỀ TÀI 4](#_Toc7657)**

**[1.1 LÝ DO CHỌN ĐỀ TÀI 4](#_Toc10856)**

**[1.2 MỤC TIÊU NGHIÊN CỨU 4](#_Toc20297)**

**[1.2.1 Mục tiêu đề tài 4](#_Toc31801)**

**[1.2.2 Đối tượng và phạm vi nghiên cứu 5](#_Toc29343)**

**[1.2.3 Công nghệ sử dụng 5](#_Toc26620)**

**[Chương 2. CƠ SỞ LÝ THUYẾT 6](#_Toc19884)**

**[2.1 MÃ HÓA BẤT ĐỐI XỨNG RSA 6](#_Toc7411)**

**[2.1.1 Giới thiệu 6](#_Toc30989)**

**[2.1.2 Nguyên lý hoạt động 6](#_Toc26148)**

**[2.1.3 Độ an toàn của RSA 10](#_Toc15643)**

**[2.2 HÀM BĂM SHA-256 11](#_Toc13867)**

**[2.2.1 Giới thiệu 11](#_Toc1807)**

**[2.2.2 Nguyên lý hoạt động của SHA-256 12](#_Toc32350)**

**[Chương 3. PHÂN TÍCH YÊU CẦU 18](#_Toc16367)**

**[3.1 YÊU CẦU CHỨC NĂNG VÀ PHI CHỨC NĂNG 18](#_Toc5227)**

**[3.1.1 Yêu cầu chức năng 18](#_Toc23641)**

**[3.1.2 Yêu cầu phi chức năng 18](#_Toc16795)**

**[3.2 CÁC SƠ ĐỒ 18](#_Toc29474)**

**[3.2.1 Use-Case Diagram và mô tả 18](#_Toc12627)**

**[3.2.2 Sequence Diagram cho các chức năng chính 19](#_Toc23134)**

**[Chương 4. THIẾT KẾ 20](#_Toc24726)**

**[4.1 CLASS DIAGRAM 20](#_Toc1367)**

**[4.2 GIAO DIỆN NGƯỜI DÙNG 20](#_Toc32396)**

**[Chương 5. CÀI ĐẶT 22](#_Toc29553)**

**[5.1 CẤU TRÚC SOURCE CODE 22](#_Toc28259)**

**[5.2 CÁC THUẬT TOÁN QUAN TRỌNG 22](#_Toc16154)**

**[5.3 CODE MẪU CHO CÁC CHỨC NĂNG CHÍNH 23](#_Toc23806)**

**[5.3.1 Hàm băm SHA-256 23](#_Toc5418)**

**[5.3.2 Mã hóa RSA 24](#_Toc12619)**

**[Chương 6. KIỂM THỬ 26](#_Toc24680)**

**[6.1 KẾ HOẠCH KIỂM THỬ VÀ KẾT QUẢ 26](#_Toc5187)**

**[6.1.1 Kế hoạch kiểm thử 26](#_Toc4620)**

**[6.1.2 Kết quả kiểm thử 26](#_Toc30167)**

**[6.2 DEMO CÁC CHỨC NĂNG 26](#_Toc30578)**

**[6.2.1 Mã hóa RSA 28](#_Toc24155)**

**[6.2.2 Hàm băm SHA-256 29](#_Toc20068)**

**[Chương 8. KẾT LUẬN 31](#_Toc6187)**

**[8.1 KẾT QUẢ ĐẠT ĐƯỢC 31](#_Toc5278)**

**[8.2 HẠN CHẾ VÀ HƯỚNG PHÁT TRIỂN 31](#_Toc23457)**

**[8.2.1 Hạn chế 31](#_Toc13428)**

**[8.2.2 Hướng phát triển 31](#_Toc14229)**

**[8.3 BÀI HỌC KINH NGHIỆM 31](#_Toc32719)**

**[TÀI LIỆU THAM KHẢO 32](#_Toc765)**

# DANH MỤC HÌNH

*[Hình 2-](#_Toc14611)*[1](#_Toc14611) *[. Mô tả cách mã hóa và giải mã của RSA](#_Toc14611)* [9](#_Toc14611)

*[Hình 2-](#_Toc23790)*[2](#_Toc23790) *[. Quá trình xử lý dữ liệu đầu vào của SHA-256](#_Toc23790)* [13](#_Toc23790)

*[Hình 2-](#_Toc31921)*[3](#_Toc31921) *[. Quá trình mã hóa dữ liệu của SHA-256](#_Toc31921)* [14](#_Toc31921)

*[Hình 3-](#_Toc18109)*[1](#_Toc18109) *[. Sơ đồ Use-Case](#_Toc18109)* [19](#_Toc18109)

*[Hình 3-](#_Toc2440)*[2](#_Toc2440) *[. Sơ đồ trình tự xử lý yêu cầu](#_Toc2440)* [20](#_Toc2440)

*[Hình 4-](#_Toc15430)*[1](#_Toc15430) *[. Sơ đồ các lớp đối tượng](#_Toc15430)* [21](#_Toc15430)

*[Hình 4-](#_Toc3624)*[2](#_Toc3624) *[. Giao diện ban đầu](#_Toc3624)* [21](#_Toc3624)

*[Hình 4-](#_Toc14391)*[3](#_Toc14391) *[. Giao diện chi tiết thuật toán mã hóa RSA](#_Toc14391)* [21](#_Toc14391)

*[Hình 4-](#_Toc28495)*[4](#_Toc28495) *[. Giao diện chi tiết hàm băm SHA-256](#_Toc28495)* [22](#_Toc28495)

*[Hình 5-](#_Toc32468)*[1](#_Toc32468) *[. Cấu trúc cơ bản của source code](#_Toc32468)* [23](#_Toc32468)

*[Hình 5-](#_Toc6667)*[2](#_Toc6667) *[. Thuật toán băm dữ liệu SHA-256](#_Toc6667)* [23](#_Toc6667)

*[Hình 5-](#_Toc8511)*[3](#_Toc8511) *[. Thuật toán mã hóa RSA](#_Toc8511)* [24](#_Toc8511)

*[Hình 5-](#_Toc16671)*[4](#_Toc16671) *[. Chức năng chuyển dữ liệu ban đầu về chuỗi nhị phân](#_Toc16671)* [24](#_Toc16671)

*[Hình 5-](#_Toc17910)*[5](#_Toc17910) *[. Chức năng xử lý dữ liệu ban đầu](#_Toc17910)* [24](#_Toc17910)

*[Hình 5-](#_Toc2472)*[6](#_Toc2472) *[. Chức năng hiển thị kết quả từng vòng băm](#_Toc2472)* [24](#_Toc2472)

*[Hình 5-](#_Toc23197)*[7](#_Toc23197) *[. Chức năng hiển thị dữ liệu cuối cùng](#_Toc23197)* [25](#_Toc23197)

*[Hình 5-](#_Toc27392)*[8](#_Toc27392) *[. Chức năng hiển thị kết quả băm và so sánh](#_Toc27392)* [25](#_Toc27392)

*[Hình 5-](#_Toc4862)*[9](#_Toc4862) *[. Chức năng tạo cặp số nguyên tố lớn](#_Toc4862)* [25](#_Toc4862)

*[Hình 5-](#_Toc17891)*[10](#_Toc17891) *[. Chức năng trực quan hóa quá trình tìm ước chung lớn nhất](#_Toc17891)* [25](#_Toc17891)

*[Hình 5-](#_Toc22862)*[11](#_Toc22862) *[. Chức năng trực quan hóa quá trình tìm phần tử nghịch đảo](#_Toc22862)* [26](#_Toc22862)

*[Hình 5-](#_Toc30416)*[12](#_Toc30416) *[. Chức năng hiển thị khóa, mã hóa và giải mã](#_Toc30416)* [26](#_Toc30416)

*[Hình 6-](#_Toc5041)*[1](#_Toc5041) *[. Giao diện báo lỗi khi chưa có dữ liệu](#_Toc5041)* [28](#_Toc5041)

*[Hình 6-](#_Toc26546)*[2](#_Toc26546) *[. Giao diện báo lỗi khi chưa chọn thuật toán mã hóa](#_Toc26546)* [28](#_Toc26546)

*[Hình 6-](#_Toc25345)*[3](#_Toc25345) *[. Giao diện hiển thị cặp số nguyên tố lớn](#_Toc25345)* [29](#_Toc25345)

*[Hình 6-](#_Toc25792)*[4](#_Toc25792) *[. Giao diện trực quan hóa quá trình tìm ước chung lớn nhất](#_Toc25792)* [29](#_Toc25792)

*[Hình 6-](#_Toc16075)*[5](#_Toc16075) *[. Giao diện trực quan hóa quá trình tìm phần tử nghịch đảo và so sánh](#_Toc16075)* [29](#_Toc16075)

*[Hình 6-](#_Toc14498)*[6](#_Toc14498) *[. Giao diện hiển thị cặp khóa](#_Toc14498)* [29](#_Toc14498)

*[Hình 6-](#_Toc4763)*[7](#_Toc4763) *[. Giao diện hiển thị kết quả mã hóa](#_Toc4763)* [29](#_Toc4763)

*[Hình 6-](#_Toc21461)*[8](#_Toc21461) *[. Giao diện hiển thị kết quả giải mã](#_Toc21461)* [30](#_Toc21461)

*[Hình 6-](#_Toc16506)*[9](#_Toc16506) *[. Giao diện hiển thị dữ liệu chuyển qua nhị phân](#_Toc16506)* [30](#_Toc16506)

*[Hình 6-](#_Toc848)*[10](#_Toc848) *[. Giao diện hiển thị dữ liệu trước khi xử lý](#_Toc848)* [30](#_Toc848)

*[Hình 6-](#_Toc11444)*[11](#_Toc11444) *[. Giao diện hiển thị dữ liệu sau khi xử lý](#_Toc11444)* [30](#_Toc11444)

*[Hình 6-](#_Toc31336)*[12](#_Toc31336) *[. Giao diện hiển thị kết quả từng vòng băm](#_Toc31336)* [31](#_Toc31336)

*[Hình 6-](#_Toc14047)*[13](#_Toc14047) *[. Giao diện hiển thị kết quả cuối cùng](#_Toc14047)* [31](#_Toc14047)

*[Hình 6-](#_Toc13297)*[14](#_Toc13297) *[. Giao diện hiển thị kết quả mã hóa](#_Toc13297)* [31](#_Toc13297)

# DANH MỤC BẢNG

*[Bảng 2-](#_Toc12693)*[1](#_Toc12693) *[. Bảng liệt kê các mốc phá mã RSA](#_Toc12693)* [12](#_Toc12693)

*[Bảng 2-](#_Toc23976)*[2](#_Toc23976) *[. Bảng giá trị của h\_values](#_Toc23976)* [16](#_Toc23976)

*[Bảng 2-](#_Toc27754)*[3](#_Toc27754) *[. Bảng giá trị của k\_const](#_Toc27754)* [18](#_Toc27754)

# Chương 1. GIỚI THIỆU ĐỀ TÀI

## LÝ DO CHỌN ĐỀ TÀI

Trong xã hội hiện đại ngày nay, Internet là một thứ không thể thiếu đối với tất cả mọi người. Tuy nhiên, bên cạnh những lợi ích cũng như sự tiện lợi mà Internet mang lại cho cuộc sống chúng ta, thì cũng xuất hiện không ít những rủi ro. Trong số những rủi ro đấy thì việc dữ liệu bị đánh cắp vẫn luôn xảy ra với rất nhiều người. Mỗi ngày, có rất nhiều người bị những kẻ lạ mặt mạo danh công an hoặc các cơ quan nhà nước gọi tới thông báo bản thân đang bị vướng vào một vụ điều tra bí mật và yêu cầu nạn nhân phải chuyển tiền và cung cấp một số thông tin cá nhân nếu không thì sẽ bị xử phạt. Hầu hết các nạn nhân đều nghe theo dù đã được cảnh báo là do những kẻ lừa đảo này nắm gần như là toàn bộ thông tin cá nhân của những nạn nhân.

Trong bối cảnh đó, việc bảo mật dữ liệu đang là ưu tiên hàng đầu của tất cả mọi người. Trong số những cách bảo mật thì sự ra đời của các thuật toán mã hóa và hàm băm có thể xem như là một bước thay đổi lớn. Nhờ những thuật toán này, dữ liệu được bảo mật một cách tốt hơn, tránh bị những kẻ lạ mặt đánh cắp.

Chính vì vậy, em quyết định lựa chọn đề tài **“Cài đặt thuật toán mã hóa RSA và hàm băm SHA-256 trong Python”** bởi vì đây đều là những thuật toán có tính ứng dụng tương đối cao, được sử dụng rộng rãi trong rất nhiều lĩnh vực như truyền thông, ngân hàng hay thương mại điện tử,… Bên cạnh đó, việc lựa chọn đề tài này còn là một sự đáp ứng hiệu quả cho sự nguy hiểm của việc dữ liệu bị đánh cắp bởi vì dữ liệu của mỗi người rất quan trọng. Em tin rằng, việc lựa chọn đề tài không chỉ là sự hỗ trợ cho việc bảo mật dữ liệu mà còn cung cấp cơ sở, nền tảng lý thuyết vững chắc để góp phần phát triển kiển thức của bản thân, từ đó em có thể đóng góp những điều tích cực cho cộng đồng bảo mật thông tin.

## MỤC TIÊU NGHIÊN CỨU

### Mục tiêu đề tài

1. Tìm hiểu lý thuyết về các thuật toán mã hóa và hàm băm.
2. Nắm vững các khái niệm về an toàn và bảo mật thông tin, bao gồm mã hóa đối xứng, mã hóa bất đối xứng và hàm băm.
3. Hiểu được thuật toán mã hóa bất đối xứng RSA và hàm băm SHA-256.
4. Minh họa kết quả bằng giao diện cụ thể.

### Đối tượng và phạm vi nghiên cứu

1. Đối tượng nghiên cứu: Thuật toán mã hóa RSA, hàm băm SHA-256 và ngôn ngữ lập trình Python.
2. Phạm vi nghiên cứu: Ứng dụng nghiên cứu vào thực tế.

### Công nghệ sử dụng

Trong project này em sử dụng ngôn ngữ lập trình Python với sự hỗ trợ của thư viện Tkinter để hiển thị giao diện người dùng (GUI) với công cụ để lập trình là PyCharm.

# Chương 2. CƠ SỞ LÝ THUYẾT

## 2.1 MÃ HÓA BẤT ĐỐI XỨNG RSA

### 2.1.1 Giới thiệu

Giải thuật mã hóa RSA là một giải thuật mã hóa dữ liệu không đối xứng vô cùng nổi tiếng được công bố bởi các nhà khoa học Ron Rivest, Adi Shamir và Leonard Adleman vào năm 1977 tại học viện MIT và được đặt tên từ các chữ cái đầu tiên trong tên của họ. Sở dĩ giải thuật này không đối xứng là do nó hoạt động dựa trên 2 khóa khác nhau là ***khóa công khai - public key và khóa riêng - private key***. Như tên gọi của chúng, khóa công khai cho phép tất cả mọi người đều có thể sử dụng thì khóa riêng lại chỉ cho phép duy nhất một người sử dụng.

Một số ví dụ về việc ứng dụng RSA có thể kể đến như việc một máy khách truy cập vào một trình duyệt bất kỳ và gửi một khóa công khai cho máy chủ để yêu cầu dữ liệu trong hệ thống client - server. Khi đó, máy chủ sẽ mã hóa dữ liệu cần gửi dựa trên khóa riêng mà máy khách gửi tới. Khi đã mã hóa xong, máy chủ sẽ gửi dữ liệu đã mã hóa cho máy khách. Cuối cùng, khi đã nhận được dữ liệu từ máy chủ thì máy khách chỉ cần việc giải mã là xong. Chính bởi vì việc mã hóa này không đối xứng nên không một ai có thể giải mã được trừ khi có một bên thứ ba có được cái khóa công khai từ trình duyệt. Điều này sẽ giúp bảo mật dữ liệu một cách tốt hơn.

### 2.1.2 Nguyên lý hoạt động

#### 2.1.2.1 Một số khái niệm về lý thuyết số

Trước khi bước vào nguyên lý hoạt động của RSA, ta cần phải có một số kiến thức về lý thuyết số, cụ thể là phép chia modulo.

1. *Phép chia modulo*

Phép chia modulo chính là phép chia lấy phần dư. Ví dụ như hoặc . Một cách tổng quát hơn, ta có

Nếu hai số a và b cùng có số dư trong phép chia cho n thì ta nói a và b đồng dư trong phép chia modulo cho n và kí hiệu là

Nếu thì ta nói a chia hết cho n hay n là ước số của a.

1. *Một số tính chất của phép chia modulo*

Cho là các số nguyên dương. Khi đó phép chia modulo có các tính chất sau

* Phép cộng
* Phép trừ
* Phép nhân

1. *Phần tử nghịch đảo của phép nhân modulo*

Nếu hai số nguyên và nguyên tố cùng nhau thì tồn tại số nguyên w thỏa mãn

Ta gọi w là phần tử nghịch đảo của trong phép modulo cho và kí hiệu là .

**Ví dụ:** Do và là hai số nguyên tố cùng nhau nên ta tìm được vì

1. *Định lý Fermat*

**Định lý:** Nếu là số nguyên tố và là số nguyên không chia hết cho thì

1. *Phép lũy thừa modulo*

Ta định nghĩa phép lũy thừa modulo để tính y từ các số nguyên như sau

1. *Thuật toán Euclid*

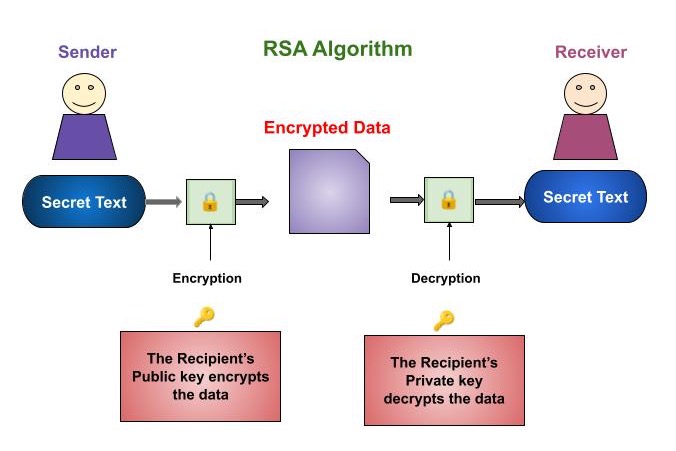
Thuật toán Euclid dùng để tìm ước số chung lớn nhất của hai số nguyên và . Ta kí hiệu ước số chung lớn nhất này là . Thuật toán này dựa trên định lý sau

**Định lý:** Với mọi số nguyên và thì ta có

Ngoài ra còn có thuật toán Euclid mở rộng. Đúng như tên gọi, thuật toán này mở rộng thuật toán Euclid ở chỗ trong trường hợp và nguyên tố cùng nhau, nghĩa là với . Khi đó thuật toán cho biết thêm giá trị nghịch đảo của trong phép chia modulo , hay

#### 2.1.2.2 Nguyên tắc hoạt động của RSA

*a. Ý tưởng*



*Hình 2-1**. Mô tả cách mã hóa và giải mã của RSA*

Về mặt tổng quát thì RSA là một phương pháp mã hóa theo khối, trong đó bản rõ M và bản mã C là các số nguyên từ 0 đến với i là số bits của khối và thường là 1024 bits. RSA sử dụng hàm một chiều là vấn đề phân tích một số thành thừa số nguyên tố. Và để thực hiện mã hóa và giải mã, RSA dùng phép lũy thừa modulo.

*b. Các bước thực hiện*

1. Chọn hai số nguyên tố lớn và tính . Cần chọn sao cho

Với thì là một số nguyên dài khoảng 309 chữ số.

1. Tính .
2. Tìm một số nguyên nguyên tố cùng nhau với .
3. Tìm một số nguyên là nghịch đảo của trong phép modulo cho .
4. Chọn khóa công khai , khóa riêng .
5. Việc mã hóa được thực hiện theo công thức

* Phương án 1: Mã hóa bảo mật .
* Phương án 2: Mã hóa chứng thực .

1. Việc giải mã được thực hiện theo công thức

* Phương án 1: Mã hóa bảo mật .
* Phương án 1: Mã hóa chứng thực .

**Ví dụ về mã hóa RSA với kích thước khóa là 6 bits:**

1. Chọn thì .
2. .
3. Chọn nguyên tố cùng nhau với .
4. Tính nghịch đảo của trong phép modulo cho thì ta được .
5. Khóa công khai , khóa riêng .

Nếu thực hiện theo phương án 1:

1. Mã hóa bản rõ
2. Giải mã bản mã

Nếu thực hiện theo phương án 2:

1. Mã hóa bản rõ
2. Giải mã bản mã

### 2.1.3 Độ an toàn của RSA

Sau đây ta sẽ xem xét một số các tấn công phương pháp RSA

1. Vét cạn khóa: Cách tấn công này thủ tất cả các khóa có thể có để tìm ra bản giải mã có ý nghĩa, tương tự như cách thử khóa K của mã hóa đối xứng. Tất nhiên, với N đủ lớn thì việc tấn công là bất khả thi.
2. Phân tích N thành thừa số nguyên tố : Chúng ta đã nói rằng việc phân tích phải là bất khả thi thì mới là hàm một chiều. Đấy là nguyên tắc hoạt động của RSA. Tuy nhiên, nhiều thuật toán phân tích mới đã được đề xuất, cùng với tốc độ xử lý của máy tính ngày càng nhanh đã làm cho việc phân tích N thành thừa số nguyên tố không còn quá khó như trước. Năm 1977, các tác giả của RSA đã treo giải thưởng cho ai phá được RSA có kích thước của N vào khoảng 428 bits, tương đương 129 chữ số. Các tác giả này ước đoán phải mất 40 nghìn triệu năm mới có thể giải được. Tuy nhiên, vào năm 1994, câu đố này đã được giải chỉ trong vòng 8 tháng. Bảng sau đây liệt kê kích thước N của các RSA đã được phá mã cho đến hiện nay.

|  |  |  |  |
| --- | --- | --- | --- |
| ***Số chữ số của N*** | ***Số bits*** | ***Năm phá mã*** | ***Thuật toán*** |
| 100 | 332 | 1991 | Quadratic sieve |
| 110 | 365 | 1992 | Quadratic sieve |
| 120 | 398 | 1993 | Quadratic sieve |
| 129 | 428 | 1994 | Quadratic sieve |
| 130 | 431 | 1996 | GNFS |
| 140 | 465 | 1999 | GNFS |
| 155 | 512 | 1999 | GNFS |
| 160 | 530 | 2003 | Lattice sieve |
| 174 | 576 | 2003 | Lattice sieve |
| 200 | 633 | 2003 | Lattice sieve |

*Bảng 2-1**. Bảng liệt kê các mốc phá mã RSA*

Dĩ nhiên việc phá mã trên chỉ được thực hiện trong phòng thí nghiệm. Tuy nhiên người ta cho rằng kích thước của N phải khoảng 1024 bits (tức khoảng 309 chữ số) thì mới đảm bảo an toàn thật sự.

1. Đo thời gian: Đây là một phương pháp phá mã không dựa vào mặt toán học của thuật toán RSA, mà dựa vào một “hiệu ứng lề” sinh ra bởi quá trình giải mã RSA. Hiệu ứng lề đó là thời gian thực hiện giải mã. Giả sử người phá mã có thể đo được thời giải mã dùng thuật toán bình phương liên tiếp. Trong thuật toán bình phương liên tiếp, nếu một bits của d là 1 thì xảy ra hai phép modulo, nếu bits đó là 0 thì chỉ có một phép modulo, do đó thời gian thực hiện giải mã là khác nhau. Bằng một số phép thử chosen-plaintext, người phá mã có thể biết được các bits của d là 0 hay 1 và từ đó biết được d.

Phương pháp mã hóa này là một ví dụ cho thấy việc thiết kế một hệ mã an toàn rất phức tạp. Người thiết kế phải lường trước được hết các tình huống có thể xảy ra.

## 2.2 HÀM BĂM SHA-256

### 2.2.1 Giới thiệu

Hàm băm là một thuật toán nhận một chuỗi dữ liệu đầu vào và xuất ra một chuỗi kí tự với độ dài cố định và nhỏ. Chuỗi kí tự đầu ra đóng vai trò như “dấu vân tay” của dữ liệu đầu vào, nghĩa là một chuỗi dữ liệu đầu vào sẽ cho ra duy nhất một chuỗi kí tự đầu ra. Tuy nhiên, hàm băm là một chiều nên không thể suy được dữ liệu ban đầu từ kết quả sau khi băm. Việc thay đổi một kí tự trong dữ liệu vào sẽ cho ra một chuỗi kí tự khác rất nhiều so với chuỗi kí tự ban đầu. Tuy nhiên, do việc dữ liệu ra bị giới hạn về độ dài nên chắc chắn sẽ không tránh được việc 2 dữ liệu vào khác nhau cho ra cùng một dữ liệu ra.

SHA-256 (Secure Hash Algorithm 256) là một hàm băm mã hóa được sử dụng rộng rãi trong mã hóa và bảo mật thông tin. Nó thuộc họ thuật toán băm SHA-2, được thiết kế bởi Cơ quan An ninh Quốc gia Hoa Kỳ (NSA) và được phát triển bởi Viện Tiêu chuẩn và Công nghệ Quốc gia (NIST). Đúng như tên gọi, SHA-256 nhận giá trị đầu vào và tạo ra kết quả băm dài 256 bit. SHA-256 được ứng dụng trong rất nhiều lĩnh vực khác nhau như

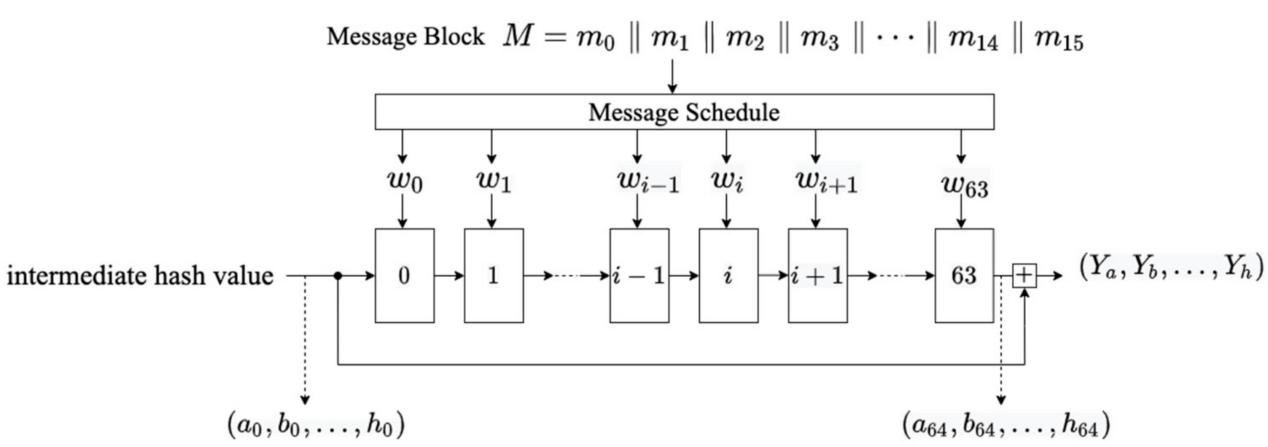
1. Bảo mật dữ liệu: SHA-256 tạo kết quả băm duy nhất cho dữ liệu, giúp xác định tính toàn vẹn của dữ liệu. Bất kỳ thay đổi nào trong dữ liệu sẽ dẫn đến thay đổi kết quả ban đầu, giúp phát hiện sự thay đổi trái phép.
2. Xác thực mật khẩu: Lưu trữ mật khẩu dưới dạng mã hash sẽ giúp bảo vệ mật khẩu gốc. Khi đăng nhập, mật khẩu nhập vào sẽ được băm và so sánh với kết quả băm đã được lưu trữ. Việc này sẽ giúp nâng cao bảo mật, ngăn chặn tiết lộ mật khẩu trong trường hợp hệ thống bị tấn công.
3. Chứng thực dữ liệu: Tạo mã hash cho dữ liệu để xác nhận tính toàn vẹn của dữ liệu khi truyền tải. Bên nhận có thể kiểm tra tính chính xác của dữ liệu bằng cách so sánh mã hash.
4. Blockchain: SHA-256 tạo mã hash cho các khối dữ liệu trong blockchain. Các khối dữ liệu này được liên kết với nhau tạo thành chuỗi, đảm bảo tính toàn vẹn và chống giả mạo.
5. Bảo mật SSL/TLS: SHA-256 được sử dụng trong mã hóa và xác thực trong giao thức SSL/TLS, giúp đảm bảo việc bảo mật cho các kết nối web và bảo vệ dữ liệu truyền tải khỏi bị tấn công.

### 2.2.2 Nguyên lý hoạt động của SHA-256

SHA-256 hoạt động dựa trên 2 bước là xử lý dữ liệu đầu vào và băm dữ liệu. Một số kí hiệu được sử dụng ở phần này bao gồm

1. biểu diễn cho toán tử XOR.
2. biểu diễn cho toán tử AND.
3. biểu diễn cho toán tử NOT.
4. Phép cộng ở đây là cộng nhị phân modulo .

#### 2.2.2.1 Xử lý dữ liệu đầu vào



*Hình 2-2**. Quá trình xử lý dữ liệu đầu vào của SHA-256*

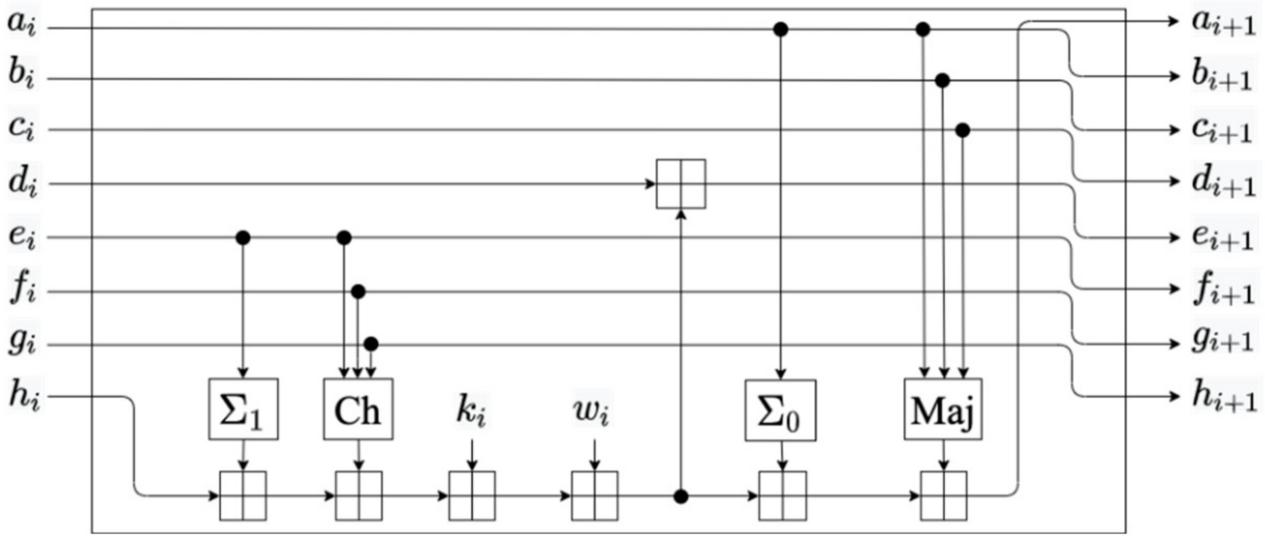
Dữ liệu đầu vào sẽ được chuyển thành chuỗi nhị phân. Sau đó, ta thêm vào chuỗi nhị phân trên một bit ‘1’ và các bit ‘0’ sao cho khi thêm 64 bit là độ dài của chuỗi nhị phân ban đầu ở dạng nhị phân thì ta được kết quả là một chuỗi nhị phân có độ dài là bội số của 512. Tiếp đến, ta sẽ chia chuỗi nhị phân trên thành các khối có độ dài 32 bit và khởi tạo một mảng w gồm 64 phần tử được xác định như sau: với thì và với thì

trong đó

và

với là phép quay phải chuỗi nhị phân , còn là phép dịch phải chuỗi nhị phân .

#### 2.2.2.2 Mã hóa dữ liệu



*Hình 2-3**. Quá trình mã hóa dữ liệu của SHA-256*

Trong quá trình mã hóa dữ liệu của SHA-256 có sử dụng đến các đại lượng sau:

1. Mảng giá trị băm gồm 8 phần tử là 32 bit đầu tiên trong phần thập phân của căn bậc 2 của 8 số nguyên tố đầu tiên.

|  |  |  |
| --- | --- | --- |
| ***Biến*** | ***Giá trị hệ nhị phân*** | ***Giá trị hệ 16*** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

*Bảng 2-2**. Bảng giá trị của h\_values*

1. Mảng hằng số gồm 64 phần tử là 32 bit đầu tiên trong phần thập phân của căn bậc 3 của 64 số nguyên tố đầu tiên.

|  |  |  |
| --- | --- | --- |
| ***Biến*** | ***Giá trị hệ nhị phân*** | ***Giá trị hệ 16*** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  | 0110000100100 |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

*Bảng 2-3**. Bảng giá trị của k\_const*

- Các biến được khởi tạo với giá trị ban đầu lần lượt từ đến .

Ta sẽ thực hiện 1 vòng lặp với chạy từ đến , trong đó tại mỗi bước ta sẽ tính toán như sau

1. *Khai báo các biến*

Ta khai báo các biến như sau

1. *Cập nhật giá trị*

Ta cập nhật giá trị các biến như sau:

Sau khi kết thúc vòng lặp, ta cập nhật lại giá trị của như sau

#### 2.2.2.3 Kết quả mã hóa

Sau khi kết thúc quá trình mã hóa, kết quả mã hóa chính là giá trị của các biến trong hệ 16 rồi nối lại với nhau.

# Chương 3. PHÂN TÍCH YÊU CẦU

## 3.1 YÊU CẦU CHỨC NĂNG VÀ PHI CHỨC NĂNG

### 3.1.1 Yêu cầu chức năng

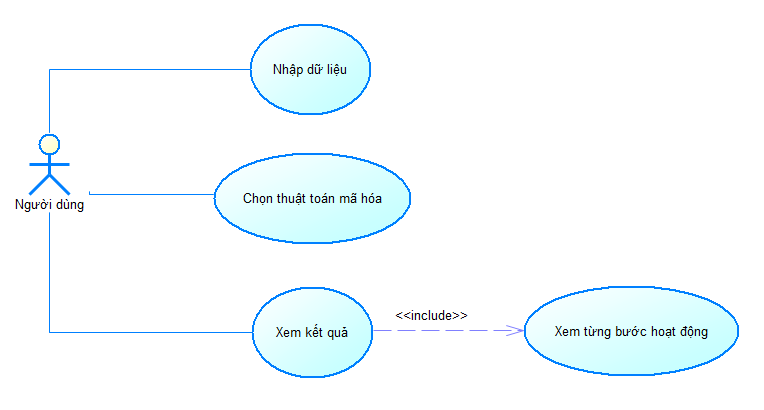
1. Tạo cặp khóa công khai và riêng tư trong RSA.
2. Mã hóa dữ liệu và giải mã với RSA.
3. Băm dữ liệu với SHA-256.

### 3.1.2 Yêu cầu phi chức năng

1. Đảm bảo về tốc độ cũng như hiệu năng.
2. Độ bảo mật cao.
3. Dễ dàng trong việc bảo trì.
4. Giao diện dễ sử dụng.

## 3.2 CÁC SƠ ĐỒ

### 3.2.1 Use-Case Diagram và mô tả

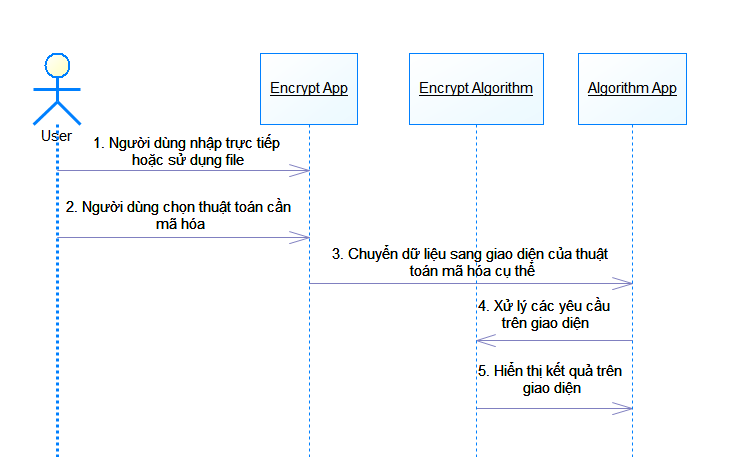


*Hình 3-1**. Sơ đồ Use-Case*

Các Use-Case gồm:

1. Nhập dữ liệu: Người dùng có thể nhập dữ liệu trực tiếp hoặc sử dụng file.
2. Chọn cách mã hóa: Người dùng có thể chọn một trong hai cách mã hóa là RSA hoặc hàm băm SHA-256.
3. Hiển thị kết quả: Người dùng có thể xem kết quả sau khi mã hóa cũng như từng bước hoạt động của thuật toán mã hóa mà người dùng chọn.

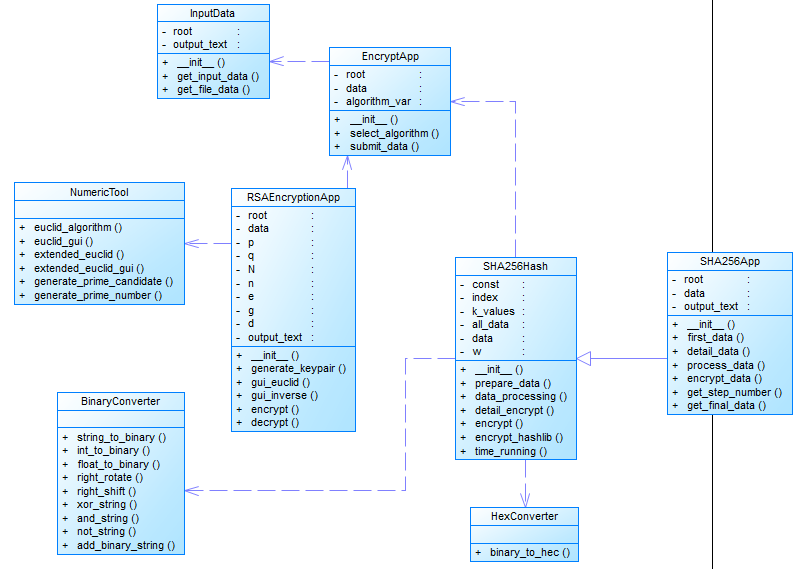
### 3.2.2 Sequence Diagram cho các chức năng chính



*Hình 3-2**. Sơ đồ trình tự xử lý yêu cầu*

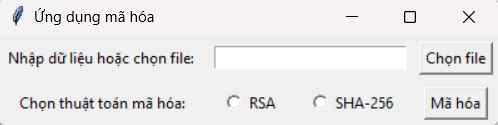
# Chương 4. THIẾT KẾ

## 4.1 CLASS DIAGRAM

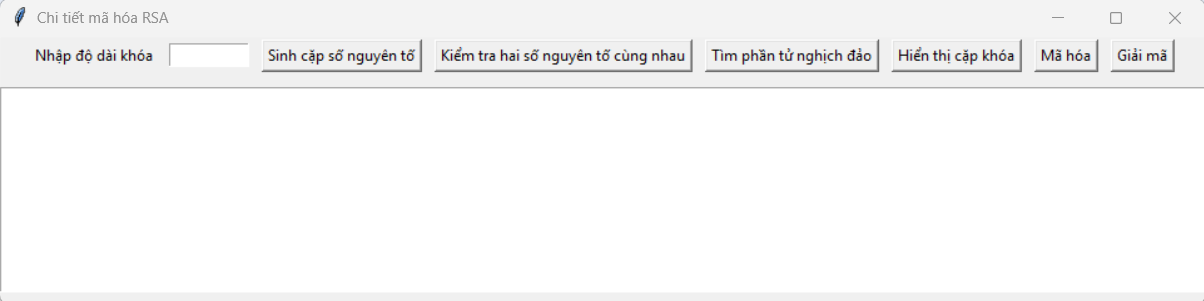


*Hình 4-1**. Sơ đồ các lớp đối tượng*

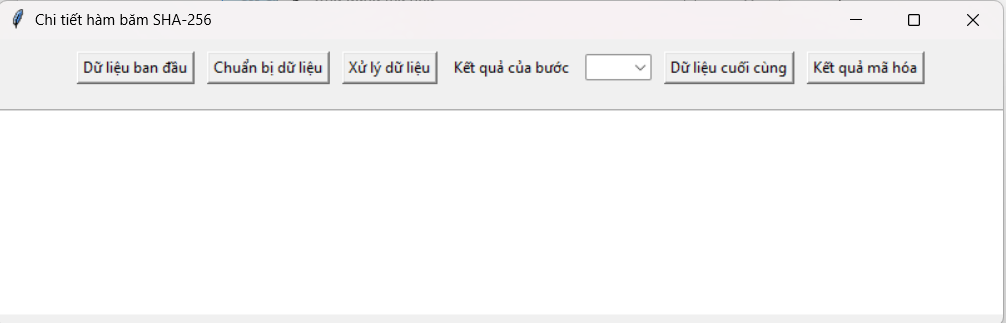
## 4.2 GIAO DIỆN NGƯỜI DÙNG



*Hình 4-2**. Giao diện ban đầu*



*Hình 4-3**. Giao diện chi tiết thuật toán mã hóa RSA*



*Hình 4-4**. Giao diện chi tiết hàm băm SHA-256*

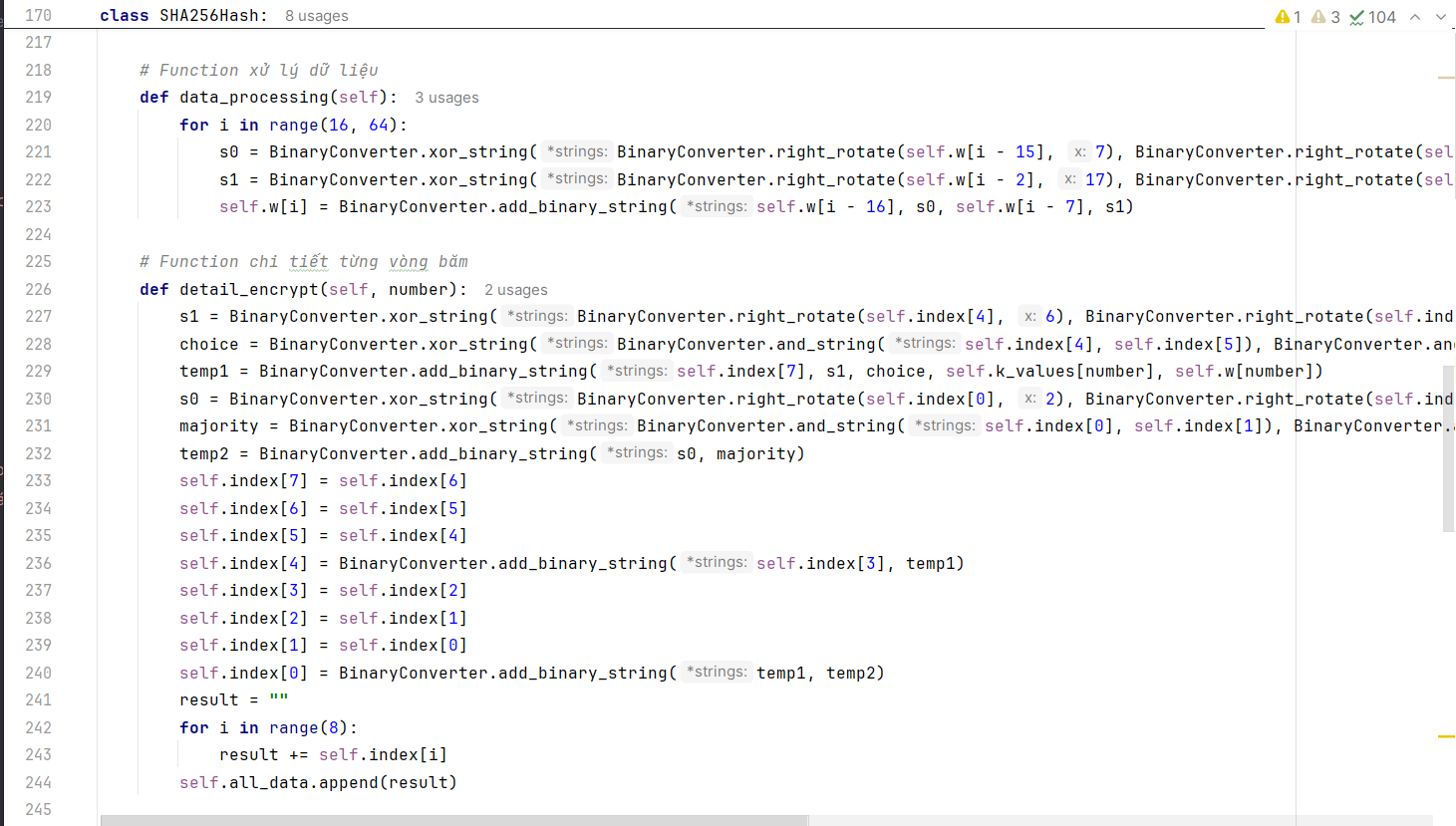
# Chương 5. CÀI ĐẶT

## 5.1 CẤU TRÚC SOURCE CODE

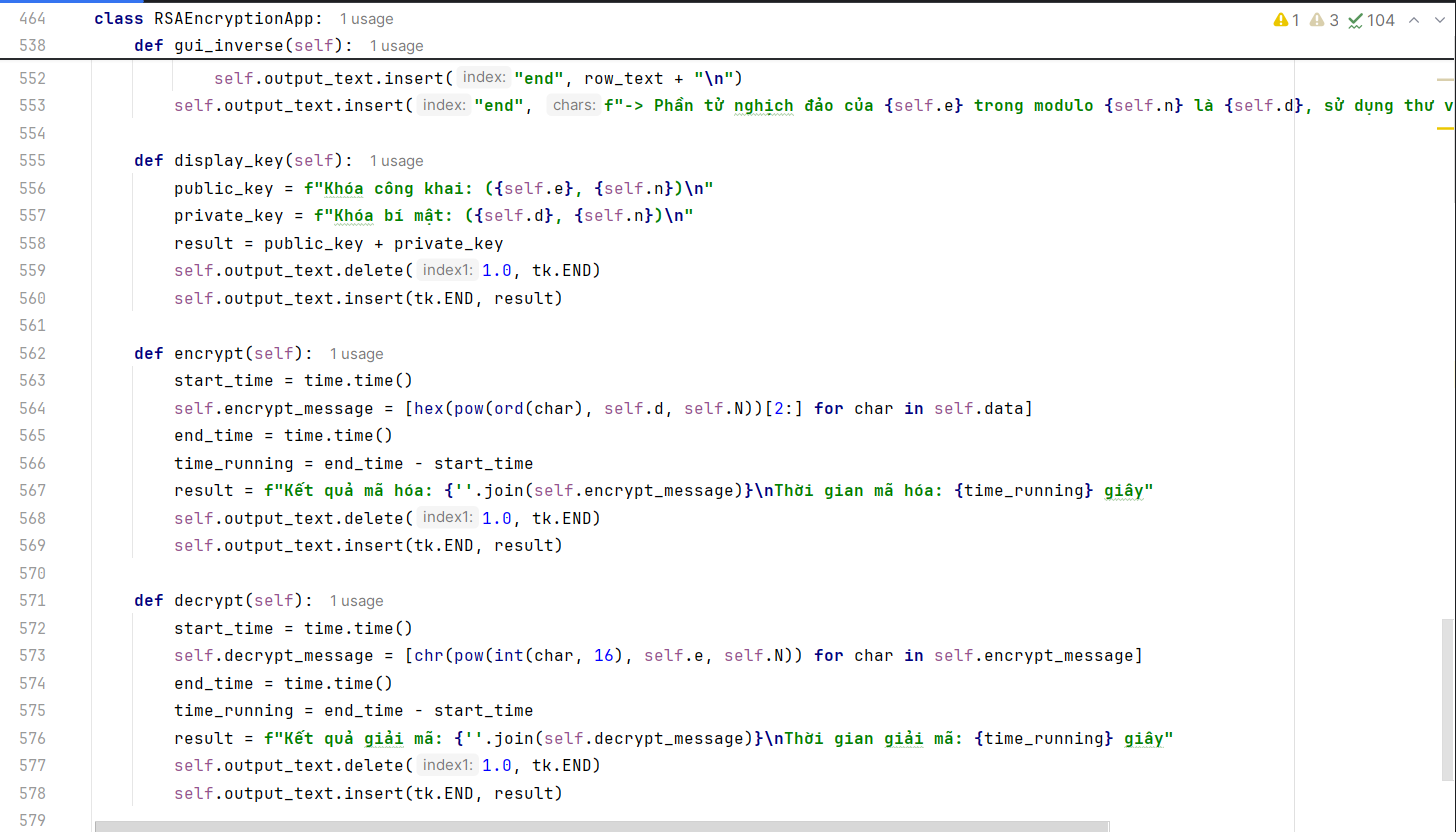


*Hình 5-1**. Cấu trúc cơ bản của source code*

## 5.2 CÁC THUẬT TOÁN QUAN TRỌNG



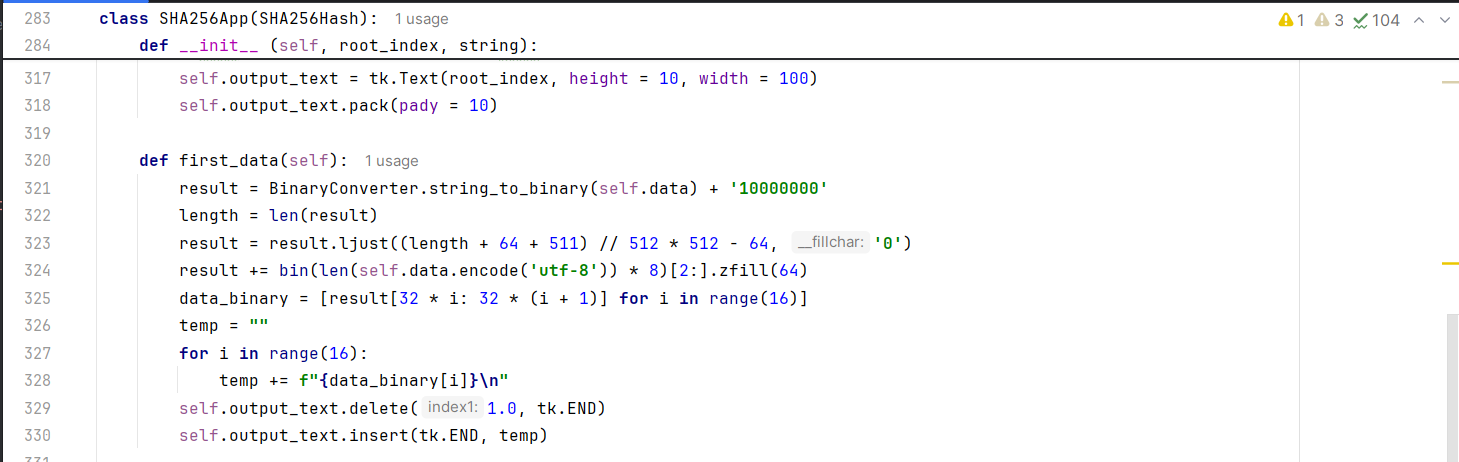
*Hình 5-2**. Thuật toán băm dữ liệu SHA-256*



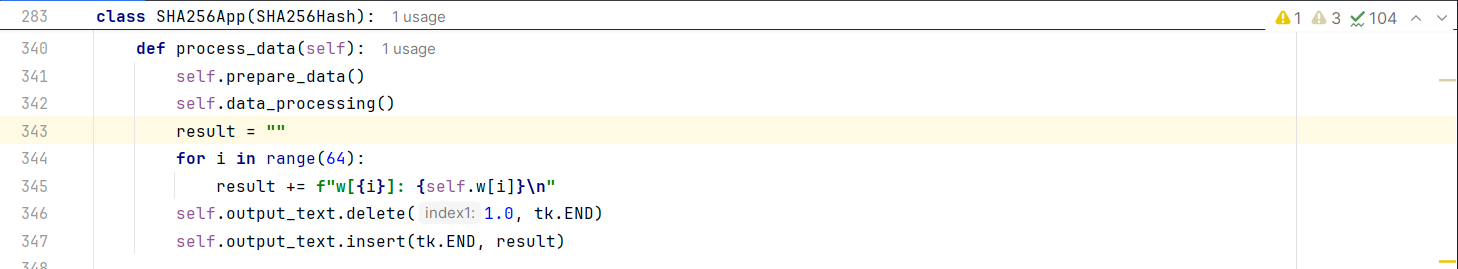
*Hình 5-3**. Thuật toán mã hóa RSA*

## 5.3 CODE MẪU CHO CÁC CHỨC NĂNG CHÍNH

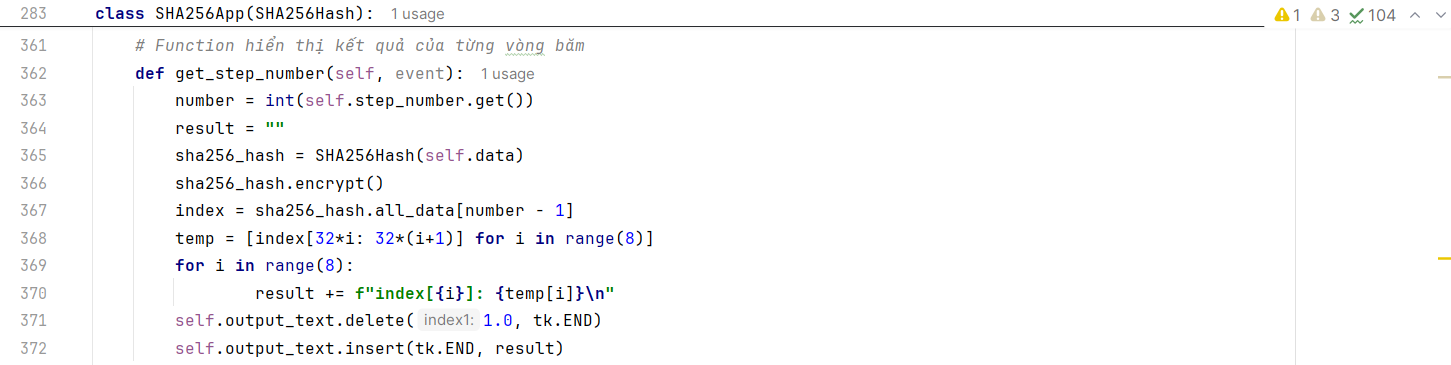
### 5.3.1 Hàm băm SHA-256



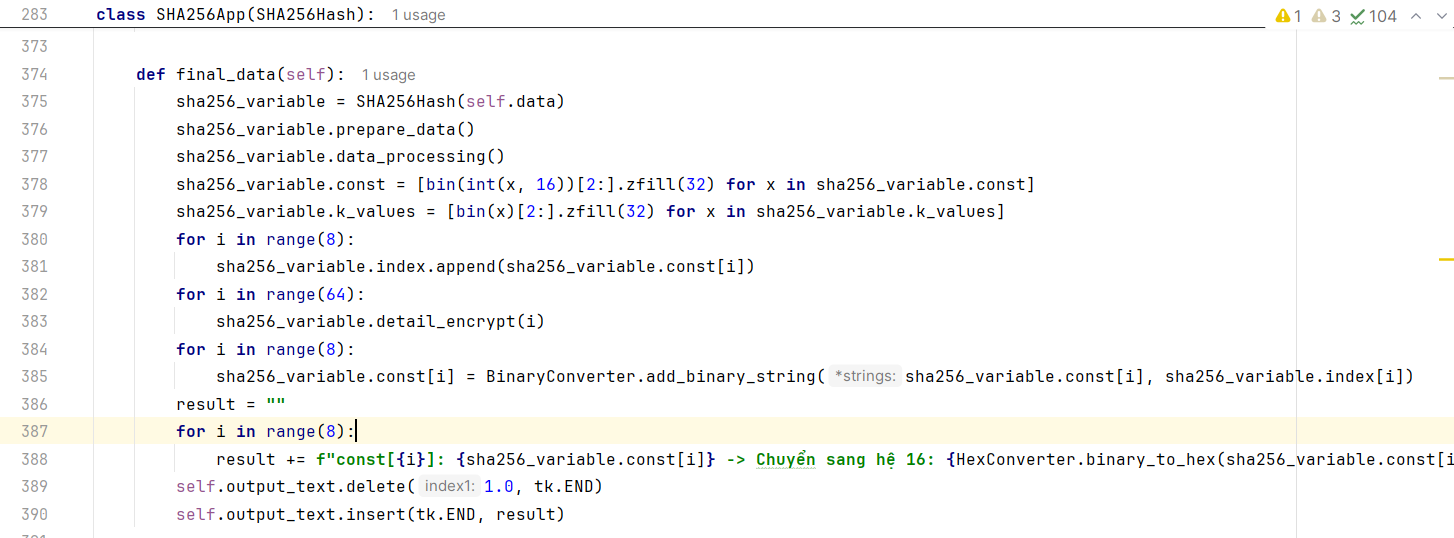
*Hình 5-4**. Chức năng chuyển dữ liệu ban đầu về chuỗi nhị phân*



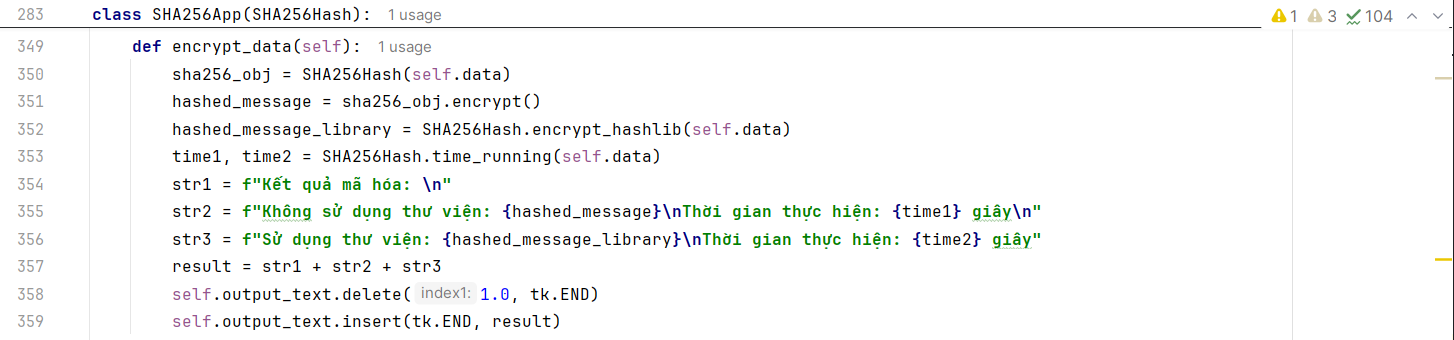
*Hình 5-5**. Chức năng xử lý dữ liệu ban đầu*



*Hình 5-6**. Chức năng hiển thị kết quả từng vòng băm*

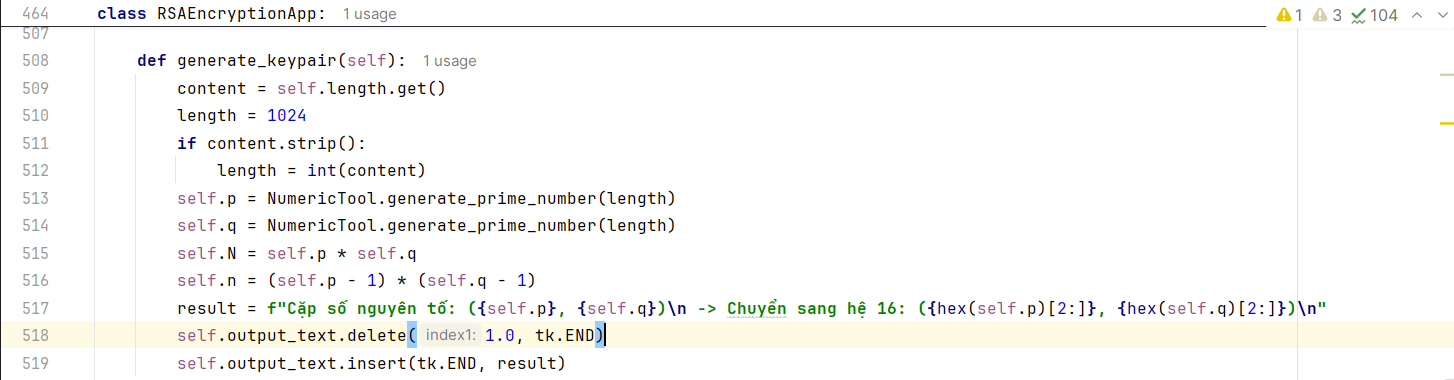


*Hình 5-7**. Chức năng hiển thị dữ liệu cuối cùng*

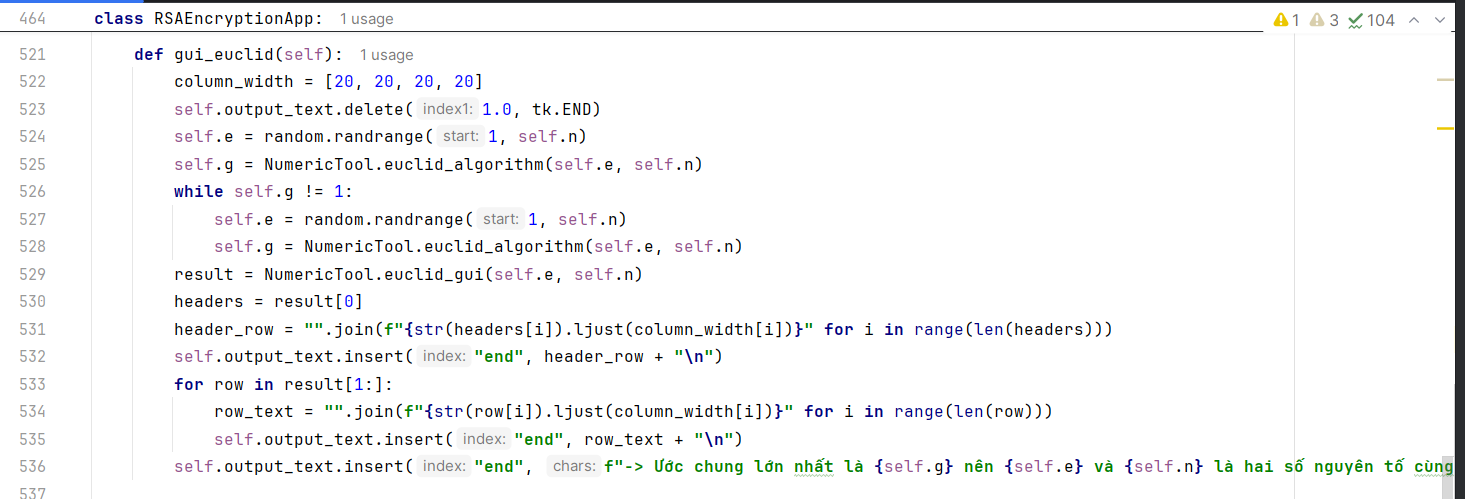


*Hình 5-8**. Chức năng hiển thị kết quả băm và so sánh*

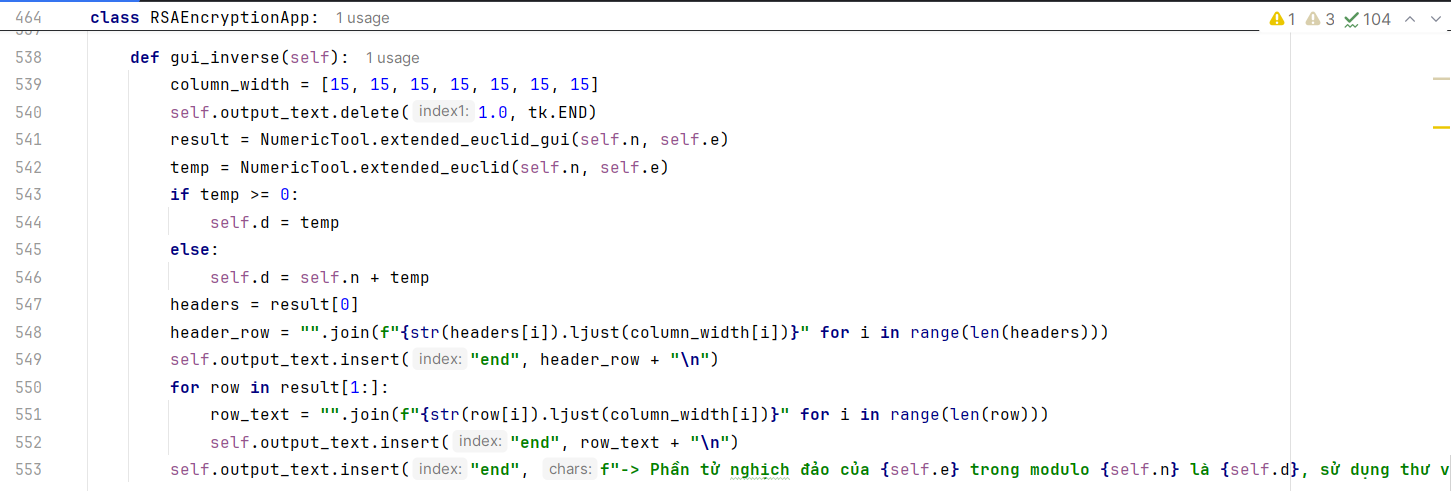
### 5.3.2 Mã hóa RSA



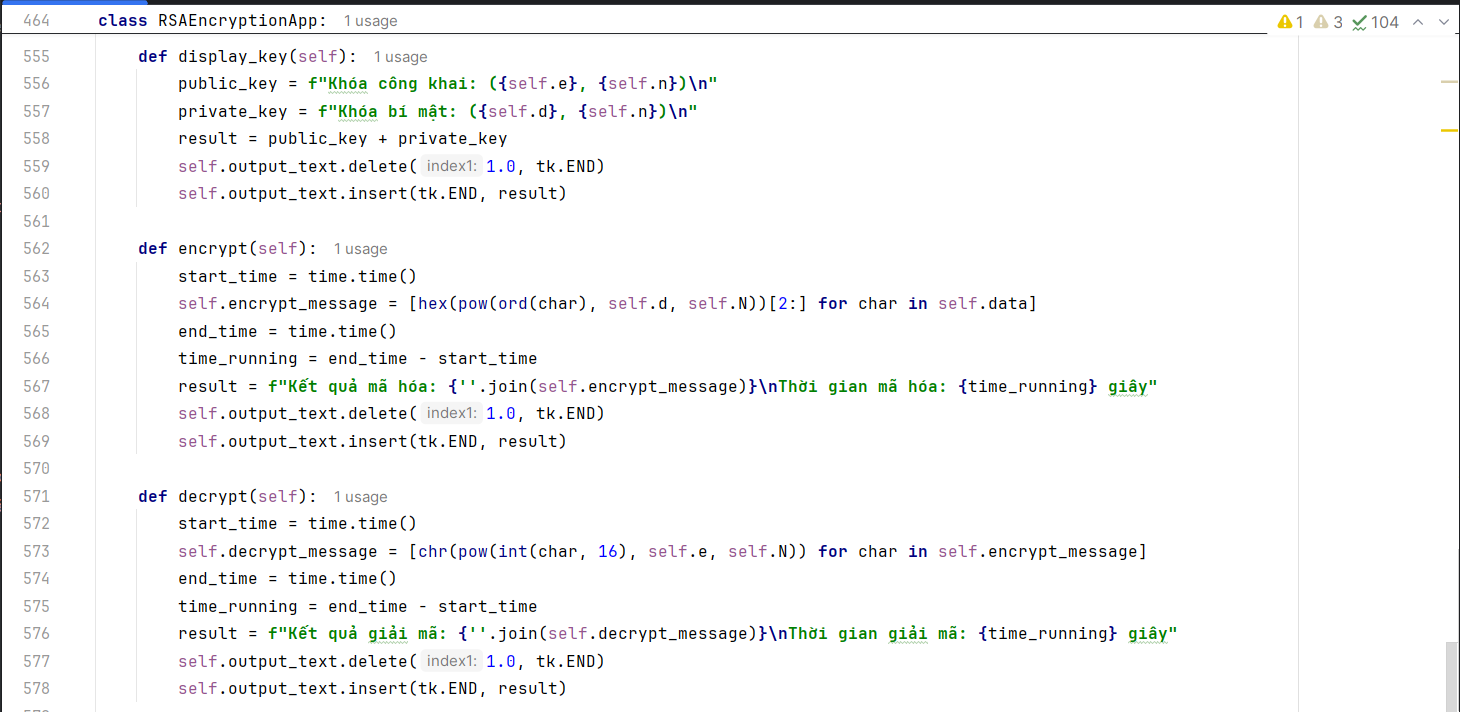
*Hình 5-9**. Chức năng tạo cặp số nguyên tố lớn*



*Hình 5-10**. Chức năng trực quan hóa quá trình tìm ước chung lớn nhất*



*Hình 5-11**. Chức năng trực quan hóa quá trình tìm phần tử nghịch đảo*



*Hình 5-12**. Chức năng hiển thị khóa, mã hóa và giải mã*

# Chương 6. KIỂM THỬ

## 6.1 KẾ HOẠCH KIỂM THỬ VÀ KẾT QUẢ

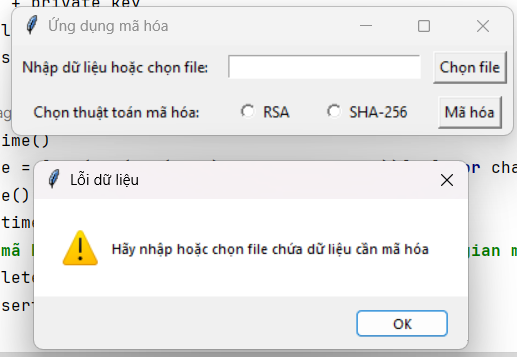
### 6.1.1 Kế hoạch kiểm thử

1. Tạo dữ liệu đầu vào mẫu, bao gồm văn bản đơn giản và phức tạp.
2. Áp dụng thuật toán mã hóa cho dữ liệu đầu vào.
3. Giải mã dữ liệu đã mã hóa và so sánh với dữ liệu ban đầu.
4. Kiểm tra kết quả giải mã có đúng không.
5. Kiểm tra các trường hợp biên (dữ liệu rỗng, dữ liệu đặc biệt như ký tự không phải chữ cái, số, hoặc ký tự Unicode).
6. So sánh thời gian chạy so với việc sử dụng thư viện có sẵn.

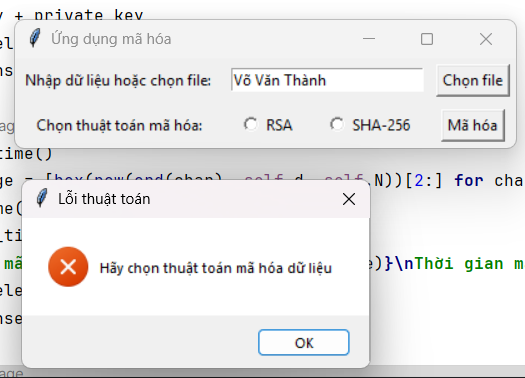
### 6.1.2 Kết quả kiểm thử

1. Tất cả các trường hợp kiểm thử đều thành công: Sau khi mã hóa và giải mã, dữ liệu trả về hoàn toàn giống với dữ liệu đầu vào ban đầu. Các ký tự đặc biệt, số, và ký tự Unicode đều được mã hóa và giải mã chính xác.
2. Mã hóa và giải mã dữ liệu với kích thước nhỏ thực hiện rất nhanh và không có sự cố. Với dữ liệu lớn, thời gian mã hóa và giải mã tăng lên nhưng vẫn trong phạm vi chấp nhận được.
3. Dữ liệu rỗng được xử lý đúng cách (không có lỗi xảy ra). Các ký tự đặc biệt, chẳng hạn như dấu chấm hỏi, dấu chấm than, và ký tự Unicode đều được xử lý chính xác.

## 6.2 DEMO CÁC CHỨC NĂNG

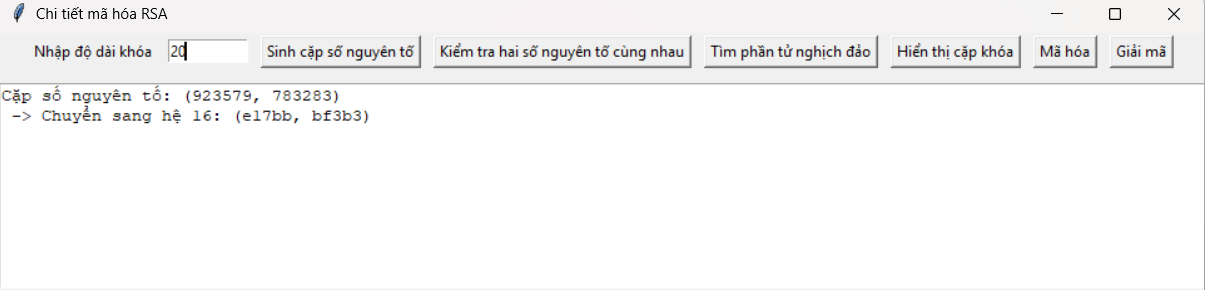


*Hình 6-1**. Giao diện báo lỗi khi chưa có dữ liệu*

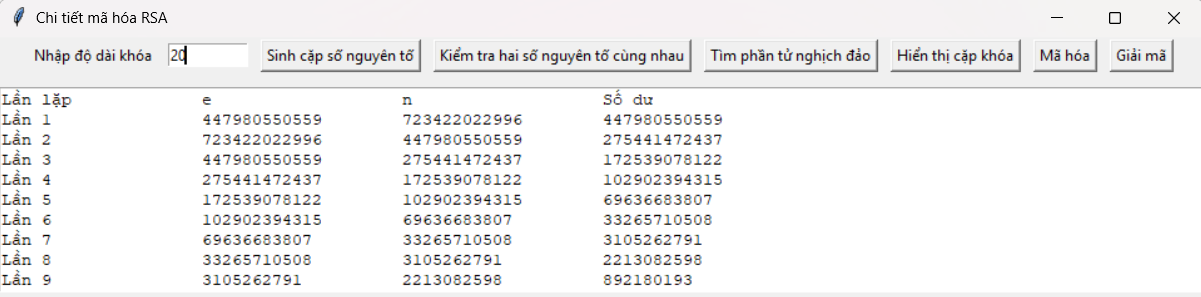


*Hình 6-2**. Giao diện báo lỗi khi chưa chọn thuật toán mã hóa*

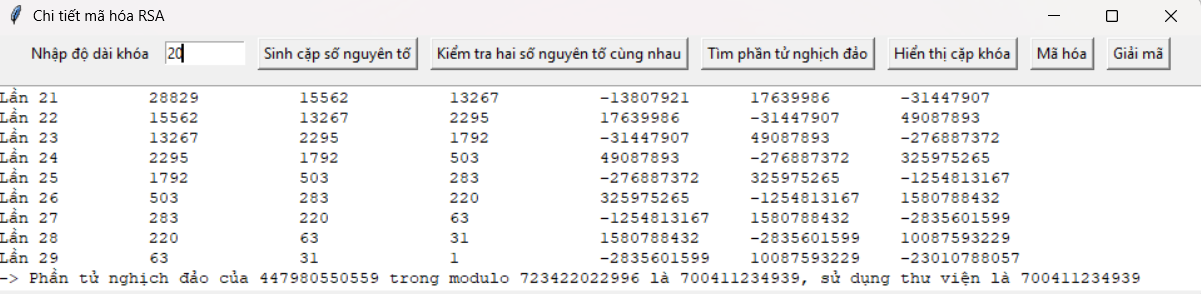
### 6.2.1 Mã hóa RSA



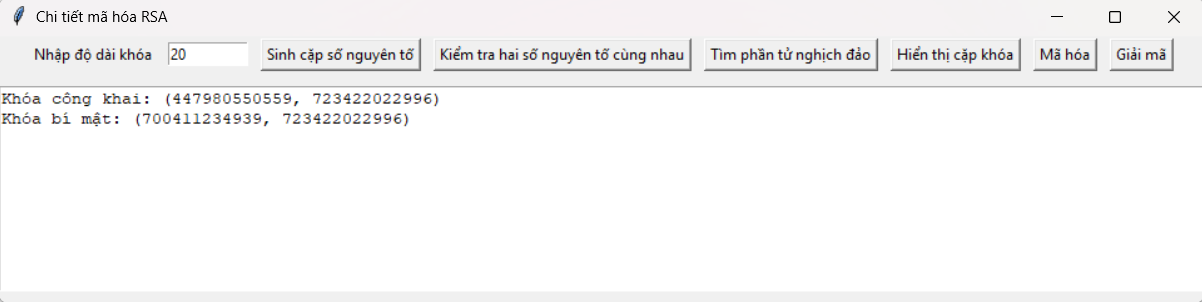
*Hình 6-3**. Giao diện hiển thị cặp số nguyên tố lớn*



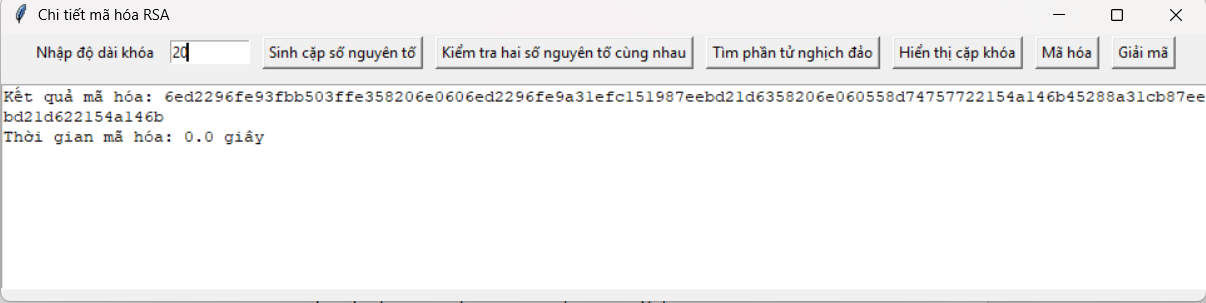
*Hình 6-4**. Giao diện trực quan hóa quá trình tìm ước chung lớn nhất*



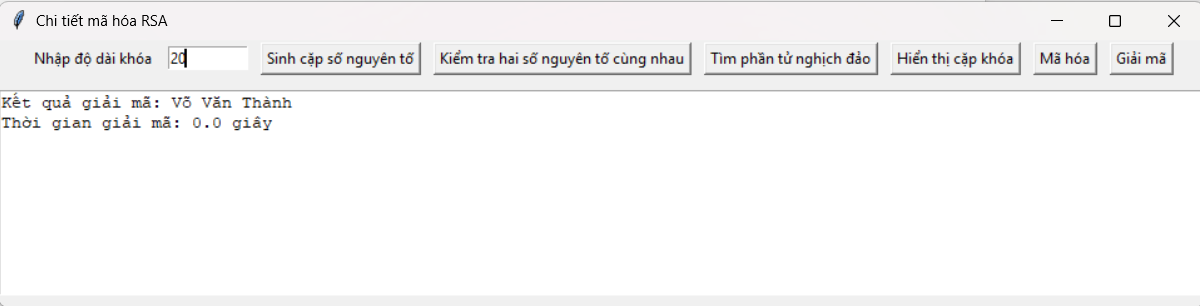
*Hình 6-5**. Giao diện trực quan hóa quá trình tìm phần tử nghịch đảo và so sánh*



*Hình 6-6**. Giao diện hiển thị cặp khóa*

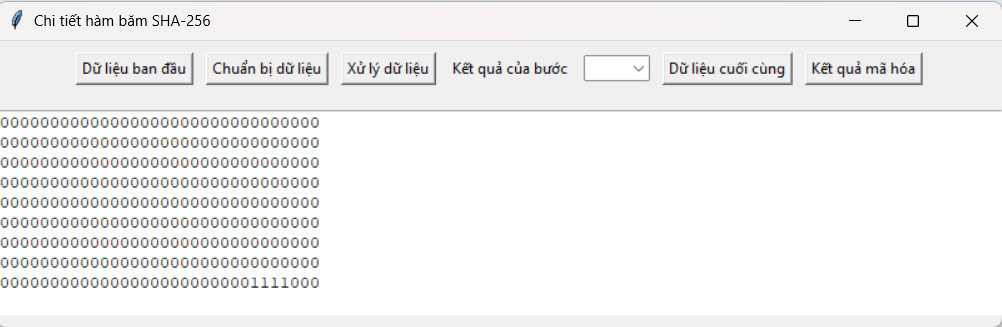


*Hình 6-7**. Giao diện hiển thị kết quả mã hóa*

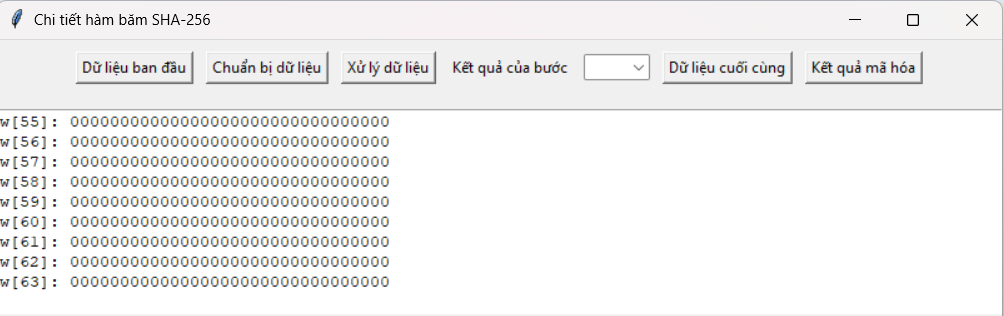


*Hình 6-8**. Giao diện hiển thị kết quả giải mã*

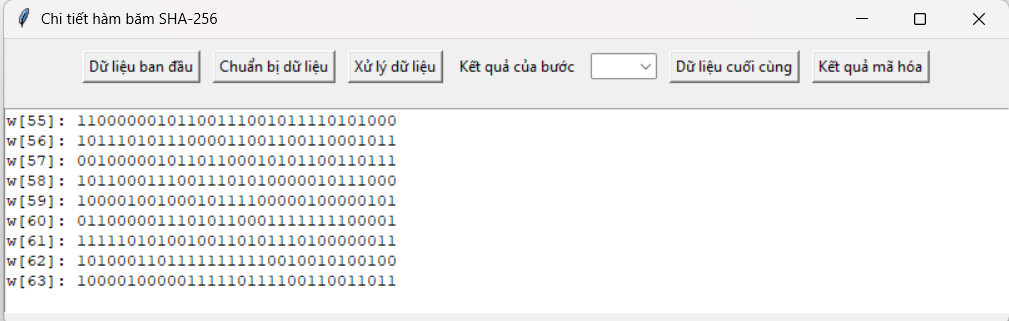
### 6.2.2 Hàm băm SHA-256



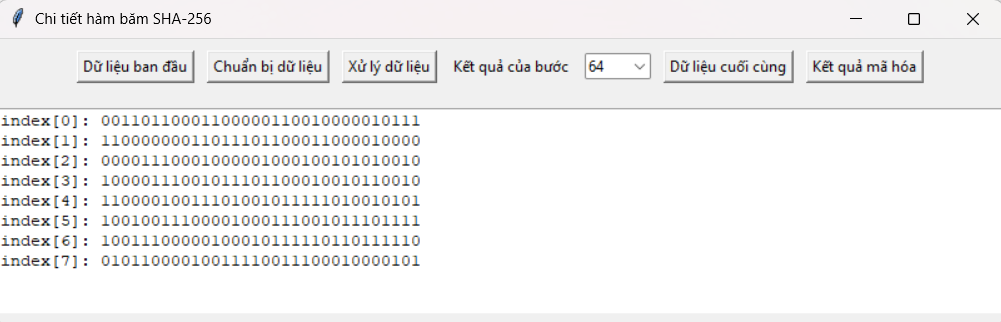
*Hình 6-9**. Giao diện hiển thị dữ liệu chuyển qua nhị phân*



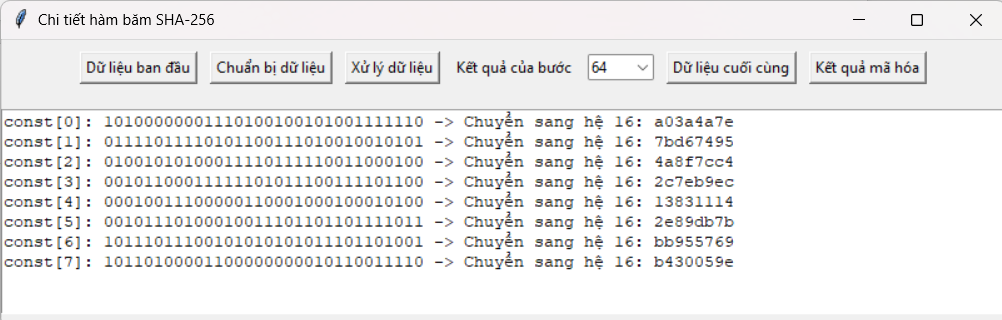
*Hình 6-10**. Giao diện hiển thị dữ liệu trước khi xử lý*



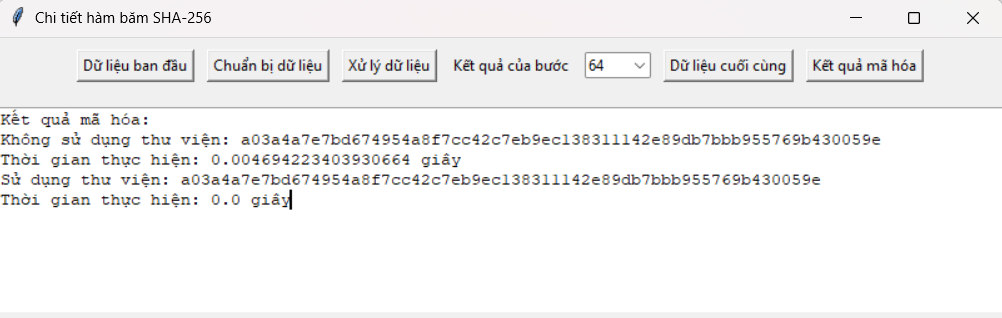
*Hình 6-11**. Giao diện hiển thị dữ liệu sau khi xử lý*



*Hình 6-12**. Giao diện hiển thị kết quả từng vòng băm*



*Hình 6-13**. Giao diện hiển thị kết quả cuối cùng*



*Hình 6-14**. Giao diện hiển thị kết quả mã hóa*

# Chương 8. KẾT LUẬN

## 8.1 KẾT QUẢ ĐẠT ĐƯỢC

1. Mô phỏng được các thuật toán mã hóa.
2. Kiểm tra được tính đúng đắn của các thuật toán.
3. Tạo được giao diện người dùng dễ sử dụng, tương thích với nhiều thiết bị.
4. Code dễ bảo trì cũng như tái sử dụng.

## 8.2 HẠN CHẾ VÀ HƯỚNG PHÁT TRIỂN

### 8.2.1 Hạn chế

1. Giao diện còn đơn giản, chưa có nhiều chức năng.
2. Chưa xuất được kết quả mã hóa thành file.
3. Chưa có chức năng lưu cặp khóa của RSA thành file.

### 8.2.2 Hướng phát triển

Có thể phát triển bằng cách thêm nhiều thuật toán mã hóa khác như DES, AES,…

## 8.3 BÀI HỌC KINH NGHIỆM

1. Cần hiểu rõ nguyên lý hoạt động của từng thuật toán mã hóa.
2. Cần kiểm tra tính đúng đắn của kết quả bằng cách so sánh với thư viện.
3. Biết xử lý các trường hợp ngoại lệ.
4. So sánh thời gian chạy cũng như hiệu suất.

# TÀI LIỆU THAM KHẢO

1. Phạm Văn Nam, *Bài giảng Lập trình Python*
2. Trần Minh Văn, *Giáo trình An toàn và bảo mật thông tin*
3. Lê Thị Bích Hằng - Nguyền Đình Hưng, *Phân tích và thiết kế hướng đối tượng sử dụng UML*