TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA KHOA HỌC ỨNG DỤNG**



**HỌ VÀ TÊN SINH VIÊN: NGUYỄN VĂN A**

**BÁO CÁO CUỐI KỲ**

**DỰ ÁN CÔNG NGHỆ THÔNG TIN 2**

KỸ THUẬT PHẦN MỀM

**TP. HỒ CHÍ MINH, THÁNG 4 NĂM 2023**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA KHOA HỌC ỨNG DỤNG**



**HỌ VÀ TÊN SINH VIÊN: NGUYỄN VĂN A**

**BÁO CÁO CUỐI KỲ**

**DỰ ÁN CÔNG NGHỆ THÔNG TIN 2**

KỸ THUẬT PHẦN MỀM

Người hướng dẫn

**Dung Cẩm Quang**

**TP. HỒ CHÍ MINH, THÁNG 4 NĂM 2023**

# **LỜI CẢM ƠN**

Chào thầy Trần Lương Quốc Đại,  
Em xin gửi lời cảm ơn chân thành đến thầy trong suốt quá trình học tập của em. Thầy đã đóng góp rất nhiều cho sự phát triển và tiến bộ của em.  
Thầy là người giáo viên tuyệt vời, không chỉ có kiến thức chuyên môn sâu rộng mà còn rất tâm huyết với công việc giảng dạy. Nhờ có sự hướng dẫn và giúp đỡ của thầy, em đã tiếp cận và hiểu sâu hơn về nhiều khái niệm mới trong lĩnh vực học tập của mình.  
Em mong rằng, những kiến thức và kinh nghiệm mà thầy đã truyền đạt sẽ tiếp tục giúp đỡ và phát triển em trong tương lai. Em lại một lần nữa gửi lời cảm ơn chân thành đến thầy.  
Em xin chân thành cảm ơn.

**CÔNG TRÌNH ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Tôi xin cam đoan đây là công trình nghiên cứu của riêng tôi và được sự hướng dẫn khoa học của TS. Nguyễn Văn A. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong Dự án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

**Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung Dự án của mình**. Trường Đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày … tháng … năm 20..*

*Tác giả*

*(Ký tên và ghi rõ họ tên)*

MỤC LỤC

[LỜI CẢM ƠN 3](#_Toc143100007)

[MỤC LỤC 5](#_Toc143100008)

[NỘI DUNG BÁO CÁO 6](#_Toc143100009)

[I. Tìm Modulo nghịch đảo *n* 6](#_Toc143100010)

[**1. Modulo là gì?** 6](#_Toc143100011)

[**2. Inverse Modulo *n* (Nghịch đảo Modulo *n*)** 6](#_Toc143100012)

[**2.1 Cơ sở toán học:** 6](#_Toc143100013)

[**3. Dùng Extended Euclidean algorithm (thuật toán Euclid mở rộng) để tìm Inverse Modulo *n* (Nghịch đảo Modulo *n*)** 6](#_Toc143100014)

[**3.1 Extended Euclidean algorithm (thuật toán Euclid mở rộng).** 6](#_Toc143100015)

[**3.1.1 Sơ lược Euclid nguyên thủy** 6](#_Toc143100016)

[**3.1.2 Sơ lược Euclid mở rộng** 7](#_Toc143100017)

[**3.2 Áp dụng Euclid mở rộng để tìm Nghịch đảo Modulo n** 7](#_Toc143100018)

[**4. Cài đặt code Python** 9](#_Toc143100019)

[**5. Thử chương trình và xác nhận kết quả** 10](#_Toc143100020)

[II. Mã hóa RSA 12](#_Toc143100021)

[**1. Sơ lược RSA** 12](#_Toc143100022)

[**2. RSA hoạt động như thế nào?** 12](#_Toc143100023)

[**3. Thuật toán Miller-Rabin dùng để kiểm tra số nguyên tố** 13](#_Toc143100024)

[**4. Cài đặt code Python** 14](#_Toc143100025)

[**5. Xác minh kết quả và giải thích:** 15](#_Toc143100026)

[**4.1 Xác minh kết quả** 15](#_Toc143100027)

[**4.2 Giải thích** 15](#_Toc143100028)

[**6. Phân tích hiệu quả và tính bảo mật của RSA đã triển khai** 15](#_Toc143100029)

[**7. Thảo luận về các mối đe dọa bảo mật tiềm ẩn và những hạn chế của hệ thống mật mã RSA.** 15](#_Toc143100030)

[**8. Kết luận** 16](#_Toc143100031)

NỘI DUNG BÁO CÁO

## **I. Tìm Modulo nghịch đảo *n***

### **1. Modulo là gì?**

**Định nghĩa:** Trong toán học, **Modulo** là một toán tử dùng để tìm phần dư khi lấy một số nguyên này chia cho một số nguyên khác, thường được kí hiệu là *%* hoặc *mod*

- Phép chia lấy dư được thực hiện trên các số nguyên dương. Khi thực hiện phép chia lấy dư, chúng ta luôn nhận được một số nguyên không âm và nhỏ hơn số chia được cho.

### **2. Inverse Modulo *n* (Nghịch đảo Modulo *n*)**

#### **2.1 Cơ sở toán học:**

- **Nghịch đảo Modulo (Inverse Modulo)** bắt nguồn từ Số học đồng dư (Modular arithmetic), nó dùng để xác định phép chia trong *Số học đồng dư* có tồn tại hay không.

- Trong *Số học đồng dư*. Các phép **cộng, trừ, nhân** có thể được xác định rõ ràng như sau:

- Tuy nhiên với phép **chia**, nó không đơn giản như vậy do phép chia trong *Số học đồng dư* không được xác định cho mọi số:

+ Trong *Số học thông thường*, phép chia cho 0 không xác định được vì vậy 0 không thể là một ước số. Nhưng trong Số học đồng dư ta lại có trường hợp đồng dư:

### **3. Dùng Extended Euclidean algorithm (thuật toán Euclid mở rộng) để tìm Inverse Modulo *n* (Nghịch đảo Modulo *n*)**

Từ ví dụ ở mục 2.2 ta có thể tìm Nghịch đảo Modulo *n* bằng cách thử lần lượt các số cho đến khi tìm được nghiệm thỏa mãn phương trình nhưng cách này không hiệu quả khi nghiệm cần tìm quá lớn, vì vậy trong phần này chúng ta sẽ dùng **Extended Euclidean algorithm (thuật toán Euclid mở rộng)** đểtìm Nghịch đảo Modulo *n*.

#### **3.1 Extended Euclidean algorithm (thuật toán Euclid mở rộng).**

##### **3.1.1 Sơ lược Euclid nguyên thủy**

- **Thuật toán Euclid** là một phương pháp cổ điển dùng để tìm ước số chung lớn nhất của hai số nguyên. Nó được xuất bản lần đầu tiên trong Quyển VII của Euclid's Elements vào khoảng năm 300 trước công nguyên.

**Mô tả ngắn gọn thuật toán:** Gọi **a, b** là 2 số nguyên cần tìm ước chung lớn nhất. Thuật toán này chỉ đơn giản là sẽ lặp lại việc cập nhật b bằng cách lấy **a** chia lấy dư cho **b**, cho đến khi **b = 0** thì **a** chính là ước số chung lớn nhất.

##### **3.1.2 Sơ lược Euclid mở rộng**

**-** Thuật toán **Euclid mở rộng** là một phương pháp để tìm ước chung lớn nhất của hai số nguyên và cũng có thể dùng để tìm các hệ số **Bézout**, là một biểu diễn của ước chung lớn nhất như một tổ hợp tuyến tính của hai số nguyên.

- Thuật toán **Euclid mở rộng** cũng được dùng để tính ước chung lớn nhất như **Euclid** nhưng nó sẽ mở rộng ra thêm 2 biến **x** và **y** sao cho **x** và **y** thỏa mãn phương trình  
**Bézout**:

**ax + by = GCD(a, b) (1.3)**

- Cặp x, y được gọi là hệ số **Bézout** của **a** và **b**

#### **3.2 Áp dụng Euclid mở rộng để tìm Nghịch đảo Modulo n**

- Như đã trình bày ở công thức 1.1, để tìm **Nghịch đảo Modulo *n*** ta sẽ tìm **a-1**, từ công thức 1.1 gọi **x** là **a-1** ta có thể viết lại công thức dưới dạng *Số học thông thường*:

**ax**  **= kn + 1 (1.4)**

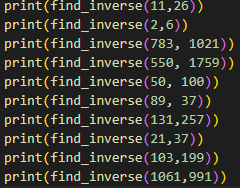
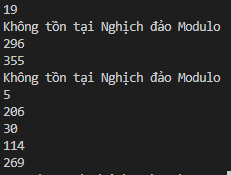
Trong đó: **k** là một số nguyên bất kì.

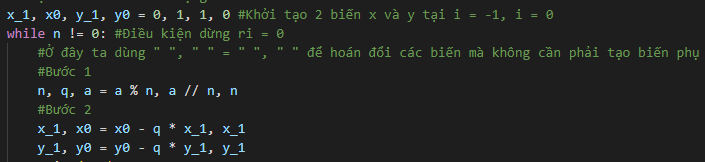
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **i** | **ri** | **qi** | **xi** | **yi** |
| -1 | 26 |  | 0 | 1 |
| 0 | 11 |  | 1 | 0 |
| 1 | 4 | 2 | -2 | 1 |
| 2 | 3 | 2 | 5 | -2 |
| 3 | 1 | 1 | -7 | 3 |
| 4 | 0 | 3 | 26 | -11 |
| 4 | 0 | 3 | 26 | -11 |
| 4 | 0 | 3 | 26 | -11 |
| 4 | 0 | 3 | 26 | -11 |
| 4 | 0 | 3 | 26 | -11 |
| 4 | 0 | 3 | 26 | -11 |
| 4 | 0 | 3 | 26 | -11 |
| 4 | 0 | 3 | 26 | -11 |
| 4 | 0 | 3 | 26 | -11 |
| 4 | 0 | 3 | 26 | -11 |
| 4 | 0 | 3 | 26 | -11 |
| 4 | 0 | 3 | 26 | -11 |
| 4 | 0 | 3 | 26 | -11 |
| 4 | 0 | 3 | 26 | -11 |
| 4 | 0 | 3 | 26 | -11 |
| 4 | 0 | 3 | 26 | -11 |
| 4 | 0 | 3 | 26 | -11 |
| 4 | 0 | 3 | 26 | -11 |
| 4 | 0 | 3 | 26 | -11 |
| 4 | 0 | 3 | 26 | -11 |
| 4 | 0 | 3 | 26 | -11 |
| 4 | 0 | 3 | 26 | -11 |
| 4 | 0 | 3 | 26 | -11 |
| 4 | 0 | 3 | 26 | -11 |
| 4 | 0 | 3 | 26 | -11 |
| 4 | 0 | 3 | 26 | -11 |
| 4 | 0 | 3 | 26 | -11 |
| 4 | 0 | 3 | 26 | -11 |
| 4 | 0 | 3 | 26 | -11 |
| 4 | 0 | 3 | 26 | -11 |
| 4 | 0 | 3 | 26 | -11 |

### **4. Cài đặt code Python**

- Ở phần này chúng ta sẽ cài đặt thuật toán Euclid mở rộng theo phương pháp tổng quát:  
Hàm **find\_inverse(a, n)** sẽ nhận vào 2 biến **a** là số cần tìm nghịch đảo, **n** là modulo *n*.

### **5. Thử chương trình và xác nhận kết quả**

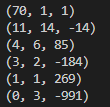
**Test:**   
**Kết quả:**  
**  
Xác nhận kết quả:**1) 11 \* 19 mod 26 = 1 (Đúng)  
2) GCD(2, 6) = 2 ⇒ Không tồn tại Nghịch đảo Modulo (Đúng)  
3) 783 \* 296 mod 1021 = 1 (Đúng)  
4) 550 \* 355 mod 1759 = 1 (Đúng)

**Giải thích cách thuật toán chạy:  
  
Ví dụ ở test số 10:**  
**Tại vòng lặp đầu tiên (a = 1061, n = 991)**  
Bước 1:  
n = a % n = 1061 % 991 = 70  
q = a // n = 1061 // 991 = 1 (cập nhật qi tiếp theo);  
a = n = 991 (cập nhật ri-2 tiếp theo);  
Bước 2:  
x\_1 = x0 – q \* x\_1 = 1 – 1 \* 0 = 1  
x0 = x\_1 = 0 (cập nhật lại xi-2 tiếp theo)

y tương tự.  
**Tại vòng lặp tiếp theo (a = 991, n = 70)**  
Bước 1:

n = a % n = 991 % 70 = 11  
q = a // n = 991 // 70 = 14 (cập nhật qi tiếp theo);  
a = n = 11 (cập nhật ri-2 tiếp theo);  
Bước 2:  
x\_1 = x0 – q \* x\_1 = 0 – 14 \* 1 = -14  
x0 = x\_1 = 1 (cập nhật lại xi-2 tiếp theo)

y tương tự.

**Cứ tiếp tục như thế cho đến khi n = 0:**  
Ta được bảng sau:  
(r, q, x)  




Kiểm tra điều kiện, ta thấy cột đầu tiên có tồn tại r = 1, tại đó x = 269  
⇒ x = 269  
**Ví dụ ở test số 2:**

Tương tự như trên ta có bảng (r, q, x):

Ta có thể thấy không có GCD(a, n) nào thỏa mãn = 1  
⇒ x = Không tồn tại Nghịch đảo Modulo

## **II. Mã hóa RSA**

### **1. Sơ lược RSA**

- RSA là một trong những thuật toán mã hóa công khai (public key cryptography) phổ biến nhất được sử dụng để bảo vệ thông tin. Đây là thuật toán đầu tiên phù hợp để tạo chữ ký số. Nó đánh dấu một tiến bộ đáng kể trong lĩnh vực mật mã học trong việc sử dụng mật mã hóa công khai. Thuật toán này được đặt tên theo tên ba nhà khoa học tạo ra nó vào năm 1977: Ron Rivest, Adi Shamir và Leonard Adleman**.**

### **2. RSA hoạt động như thế nào?**

Thuật toán RSA sử dụng một cặp khóa, bao gồm khóa công khai (public key) và khóa bí mật (private key). Khóa công khai có thể chia sẻ với người khác, khóa bí mật được giữ lại để giải mã thông tin từ khóa công khai.

**Các bước để mã hóa và giải mã:**

**Bước 1: Sinh khóa**

**Để sinh khóa ta có thể làm theo các bước sau:**1. Chọn hai số nguyên tố **p** và **q**  
2. Tính **N = pq**3. Tính **φ(N) = (p-1) \* (q-1)**  
4. Chọn một số nguyên **e** trong khoảng **(1, φ(N))** sao cho **e** và **φ(N)** cùng là số nguyên tố hay nói cách khác **GCD(e, φ(N)) = 1.**5. Tính **d** sao cho **d** là Nghịch đảo modulo của **e**, tức là **e \* d ≡ 1 (mod φ(N)).**  
6. Khóa công khai (public key) sẽ là cặp **(N, e)** và khóa bí mật (private key) là cặp **(N, d).**

**Ví dụ:**

1. Chọn hai số nguyên tố: **p=17** và **q=11.**  
2. Tính **N = p \* q = 187.**  
3. Tính **φ(N) = (p-1) \* (q-1) = 160.**  
4. Chọn **e = 7**, **e** thuộc **(1, 160)**, **GCD(7, 160) = 1**  
5. Gọi **d** là Nghịch đảo modulo của tức là **e \* d ≡ 1 (mod φ(N))**, ta có thể dùng thuật toán Euclid mở rộng đã trình bày ở phần 1.3.1 ở trên để tìm **d**. **7 \* 23 ≡ 1 (mod 160) ⇒ d = 23**.  
6. Khóa công khai là **(N, e) = (187, 7)**. Khóa bí mật là **(N, d) = (187, 23).**

**Bước 2: Mã hóa**

Giả sử chúng ta có tin nhắn (message): **“HELLO”**  
1. Ta sẽ dùng một bảng mã nào đó để chuyển chuỗi kí tự thành chuỗi số. Ví dụ ở đây ta sẽ dùng bảng mã **ASCII: H = 72, E = 69, L = 76, O = 79**  
Sau khi mã hóa và nối chúng lại ta được chuỗi số: **M = 7269767679.**  
2. Bây giờ ta sẽ sử dụng cặp khóa công khai **(N, e) = (187, 7)** để tạo bảng mã (ciphertext) bằng cách lấy **C = Me mod N** ⇒ **M7 mod 187 = 59**

**Bước 3: Giải mã**

Để giải mã người nhận sẽ dùng cặp khóa bí mật **(N, d) = (187, 23)  
M = Cd mod N** ⇒ **5923 mod 187 = 7269767679**Dựa vào bảng mã ASCII ta có thể dịch ngược **7269767679** thành **“HELLO”**

### **3. Thuật toán Miller-Rabin dùng để kiểm tra số nguyên tố**

**-** Từ ví dụ trên ta đã có thể hiểu cách hoạt động của RSA, tuy nhiên 2 số nguyên tố được chọn quá nhỏ nên thuật toán không an toàn. Trong thực tế người ta sẽ chọn p và q là 2 số nguyên tố rất lớn, mà số lớn thì rất khó để kiểm tra nó có phải là số nguyên tố hay không vì vậy thuật toán này được đưa ra để làm điều đó.

- **Thuật toán Miller-Rabin** là một thuật toán kiểm tra số nguyên tố sử dụng phổ biến trong các ứng dụng mã hóa và bảo mật. (Không chính xác 100%)

- Nếu một số nguyên **n** là số nguyên tố, thì với mọi số nguyên **a** không chia hết cho **n**, ta có thể xem **a** là một "chứng nhận" cho **n** là số nguyên tố. Nghĩa là, nếu chọn ngẫu nhiên một số nguyên **a** không phải là ước của **n,** và a thỏa mãn một số **điều kiện**, ta có thể kết luận với độ tin cậy cao rằng **n** là số nguyên tố.

**Ví dụ:**Giả sử ta muốn kiểm tra xem số nguyên **n = 17**Bước 1: Chọn một số nguyên a ngẫu nhiên trong khoảng [2, n-2], ví dụ a = 3.Bước 2: Tính b = ad mod n, với d = (n-1)/2. Ta có b = 38 mod 17 = 1.Bước 3: Nếu b = 1 hoặc b = n-1, ta có thể kết luận rằng n có thể là số nguyên tố và kết thúc thuật toán.

Để tăng độ tin cậy, ta có thể lặp lại thuật toán thêm nhiều lần nữa, chẳng hạn sau khi lặp lại 10 lần, nếu bằng 1 hoặc n-1 trong mọi lần lặp, ta có thể kết luận rằng n là số nguyên tố với độ tin cậy cao.

### **4. Cài đặt code Python**

- Trong phần này chúng ta sẽ sử dụng thư viện PyCryptodome để cài đặt mã hóa RSA

- PyCryptodome là một bản phát triển tiếp theo của PyCrypto, một thư viện mật mã hóa được viết bằng Python. PyCryptodome bao gồm các thư viện cung cấp các chức năng mã hóa và giải mã cho một loạt các thuật toán mã hóa khác nhau như AES, Blowfish, Triple DES, RSA, ECC, và nhiều hơn nữa.

- Một số tính năng khác của PyCryptodome bao gồm:

1. Hỗ trợ các hàm băm và chữ ký số

2. Hỗ trợ các chức năng chống lại tấn công khóa bằng cách sử dụng các chức năng băm, bảo vệ khóa bằng cách sử dụng các chức năng kích hoạt ngẫu nhiên và các chức năng bảo vệ bằng cách sử dụng các chức năng bảo vệ.

3. Có thể hoạt động trên nhiều nền tảng, bao gồm Windows, macOS và Linux.

- Dưới đây là code được áp dụng để mã hóa RSA: 

### 

### **5. Xác minh kết quả và giải thích:**

#### **4.1 Xác minh kết quả**

**Kết quả thu được:**

Khóa công khai (public key): -----BEGIN PUBLIC KEY-----

MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEArqVp9DoD2akfaEe1PXbD

0qzf1PMGUHmAtZ9bMEHgbmgbMWqJVyvI3f6FzQhPbj94pZLPIvvN6v7qrB5jxQez

pSZxd+UOtzwZk9j3EY13gp8J0avx1Q2ON5OfgNMPQGZ19C7ejpkn+7sErhiEwGdb

OepIw2sbQyER/b8JkY/tNpS5KV6JTMcFpbX1CyDzdWyX8yq60HmFv/nopoqT0lgd

4XG6cGLGKnCnHTPKN3JDKsdlQc5zqImfcc9CzrSby4bdMz8HAiQ0P2Z5tL6Mw+K4

8lk63nAwXe0Mkv78CwuoTFFM2pkDAy/D4xElJmAWHZ2tjkpDQyWW2D8W2N3ae22b

BQIDAQAB

⇒ Thuật toán chính xác

#### **4.2 Giải thích**

- Giải mã ciphertext bằng khóa private ta thu được kết quả:

### **6. Phân tích hiệu quả và tính bảo mật của RSA đã triển khai**

Hiệu quả và bảo mật của hệ thống mật mã RSA được triển khai phụ thuộc vào một số yếu tố như kích thước khóa, sơ đồ đệm và triển khai thuật toán.

Trong mã được triển khai, kích thước khóa được đặt thành 2048 bit, được coi là an toàn cho hầu hết các mục đích thực tế. Tuy nhiên, kích thước khóa lớn hơn có thể được yêu cầu cho các ứng dụng nhạy cảm hơn.

Thuật toán được sử dụng là PKCS1\_OAEP, là một thứ được sử dụng rộng rãi và an toàn.

Thuật toán này thêm các byte ngẫu nhiên vào tin nhắn để làm cho nó có khả năng chống lại các cuộc tấn công tốt hơn.  
Việc triển khai thuật toán trong thư viện pycryptodome được tối ưu hóa tốt cho hiệu suất, làm cho nó phù hợp để sử dụng trong các ứng dụng mà hiệu quả là một mối quan tâm.

Nhìn chung, hệ thống mật mã RSA được triển khai là an toàn và hiệu quả cho hầu hết các mục đích thực tế. Tuy nhiên, điều quan trọng cần lưu ý là tính bảo mật của hệ thống cũng phụ thuộc vào việc xử lý đúng các khóa và sử dụng các kênh liên lạc an toàn để ngăn chặn các cuộc tấn công như tấn công trung gian.

### **7. Thảo luận về các mối đe dọa bảo mật tiềm ẩn và những hạn chế của hệ thống mật mã RSA.**

**Hạn chế:  
Kích thước khóa:** Tính bảo mật của hệ thống mật mã RSA liên quan trực tiếp đến kích thước của khóa. Khi sức mạnh tính toán tăng lên, việc bẻ khóa các khóa nhỏ hơn trở nên dễ dàng hơn, tạo ra các khóa lớn hơn cần thiết để duy trì bảo mật. Tuy nhiên, các khóa lớn hơn đòi hỏi nhiều sức mạnh tính toán hơn để tạo, truyền và xử lý, điều này có thể ảnh hưởng đến hiệu suất

**Mối đe dọa bảo mật tiềm ẩn:**

**Các cuộc tấn công vào việc tạo số nguyên tố ngẫu nhiên:** Tạo số ngẫu nhiên là rất quan trọng đối với tính bảo mật của các hệ thống mật mã RSA. Nếu kẻ tấn công có thể dự đoán hoặc kiểm soát quá trình tạo số ngẫu nhiên, chúng có khả năng phát hiện ra khóa riêng. Thực hiện cẩn thận và thử nghiệm quá trình tạo số ngẫu nhiên là rất quan trọng để duy trì tính bảo mật của các hệ thống mật mã RSA.

**Máy tính lượng tử:** Hệ thống mật mã RSA dựa trên độ khó của việc chi phí giải mã lớn. Tuy nhiên, với sự ra đời của máy tính lượng tử, có thể tính toán một số lượng lớn chỉ trong nháy mắt, có thể vét cạn toàn bộ RSA. Mặc dù máy tính lượng tử vẫn còn trong giai đoạn sơ khai và máy tính lượng tử thực tế chưa có sẵn, điều quan trọng là phải xem xét tác động tiềm tàng đối với RSA và các hệ thống mật mã khác.

### **8. Kết luận**

- Dựa trên các hạn chế và các mối đe dọa ở trên, dưới đây là một số khuyến nghị để cải thiện RSA:

+ Sử dụng kích thước khóa dài hơn: Khi sức mạnh tính toán tăng lên, kích thước khóa dài hơn cần thiết để duy trì bảo mật. Kích thước khóa được đề nghị ít nhất là 2048 bits hoặc dài hơn

+ Thiết kế cẩn thận và triển khai an toàn với các thuật toán sinh số nguyên tố ngẫu nhiên.

+ Cập nhật hệ thống thường xuyên, điều này có thể giúp giải quyết các lỗ hổng đã biết và đảm bảo hệ thống được cập nhật với các biện pháp bảo mật mới nhất.

+ Quản lý Private key cẩn thận, private key là quan trọng nhất đối với hệ thống RSA. Nếu làm lộ Private key thì coi như thuật toán bị giải mã hoàn toàn.

TÀI LIỆU THAM KHẢO

1. [Chapter 3. Modular Arithmetic](https://www.doc.ic.ac.uk/~mrh/330tutor/ch03.html#:~:text=The%20division%20theorem%20tells%20us,%3D%203*5%2B2.)
2. [**https://brilliant.org/wiki/modular-arithmetic/#:~:text=Modular%20arithmetic%20is%20a%20system,modulus)%20to%20leave%20a%20remainder**](https://brilliant.org/wiki/modular-arithmetic/#:~:text=Modular%20arithmetic%20is%20a%20system,modulus)%20to%20leave%20a%20remainder)**.**
3. [Số học 4.5 - Nghịch đảo modulo (vnoi.info)](https://vnoi.info/wiki/algo/math/modular-inverse.md)
4. [Thuật toán Euclid và thuật toán Euclid mở rộng (di-mgt.com.au)](https://www.di-mgt.com.au/euclidean.html#:~:text=The%20Extended%20Euclidean%20Algorithm%20is,use%20in%20a%20computer%20program.)
5. [Modular multiplicative inverse - Wikipedia](https://en.wikipedia.org/wiki/Modular_multiplicative_inverse#Extended_Euclidean_algorithm)
6. [RSA (mã hóa) – Wikipedia tiếng Việt](https://vi.wikipedia.org/wiki/RSA_%28m%C3%A3_h%C3%B3a%29)
7. [Hệ mã hóa RSA và chữ ký số (viblo.asia)](https://viblo.asia/p/he-ma-hoa-rsa-va-chu-ky-so-6J3ZgkgMZmB)