**Modulo Arithmetic (Đại Số Modular):**

1. **Khái Niệm Cơ Bản:**

* Modulo Arithmetic là một nhánh của đại số đối với các phép toán thực hiện trên các phần dư.
* Được biểu diễn bởi ký hiệu *a* mod *b*, trong đó *a* là số cần chia và *b* là số chia.

2. **Công Thức Cơ Bản:**

* Cho mọi số nguyên *a*, *b* (trong đó *b* không bằng 0), phần dư của *a* khi chia cho *b* được ký hiệu là *a* mod *b*.
* Công thức: *a* mod *b* = *c*, với 0 ≤ *c* <∣*b*∣.
  + Nghĩa là có một số *q* sao cho *a*= *bq* + *c*, và *c* chính là phần dư.

3. **Ví Dụ Cụ Thể:**

* **Ví Dụ 1:**
  + 17mod  5=217mod5=2, vì 17=5×3+217=5×3+2.
* **Ví Dụ 2:**
  + −23mod  7=4−23mod7=4, vì −23=7×(−4)+4−23=7×(−4)+4.
* **Ví Dụ 3:**
  + 10mod  3=110mod3=1, vì 10=3×3+110=3×3+1.

4. **Ứng Dụng Thực Tế:**

* Modulo Arithmetic thường được sử dụng trong việc xác định chu kỳ, trong các hệ thống mã hóa, và trong lĩnh vực kiểm soát dữ liệu số.

5. **Chú Ý Quan Trọng:**

* Nếu *a* chia hết cho *b*, tức là *a* mod *b*=0.

6. **Tính Chất Quan Trọng:**

* **Tính Chất Cộng:** (*a*+*b*) mod *n*=(( *a* mod *n*)+(*b* mod *n*)) mod *n*
* **Tính Chất Nhân:** (*a*×*b*) mod *n*=((*a* mod *n*)×( *b*mod *n*)) mod *n*

**Ví Dụ Thêm:**

1. **Xác Định Chu Kỳ:**

* Trong hệ thống giờ đồng hồ 12 giờ, 15 mod 12 = 3, vì 15 giờ là 3 giờ chiều nay.

2. **Mã Hóa Số Nguyên Tố:**

* Trong mật mã RSA, tính toán modulo được sử dụng để bảo vệ dữ liệu thông qua một khối số nguyên tố lớn.

3. **Kiểm Tra Số Chẵn Lẻ:**

* 37 mod 2 = 1, vì 37 là số lẻ.

4. **Xử Lý Dữ Liệu Điện Tử:**

* Trong máy tính, modulo thường được sử dụng để xác định vị trí của pixel trên màn hình.

**Kết Luận:**

Modulo Arithmetic cung cấp một cách hiệu quả để làm việc với phần dư, và tính toán modulo thường xuyên xuất hiện trong nhiều lĩnh vực khác nhau. Đây là một công cụ

quan trọng và linh hoạt trong toán học và ứng dụng thực tế.

**RSA Algorithm (Thuật toán RSA):**

1. **Giới Thiệu:**

* RSA là một thuật toán mã hóa và giải mã công khai, được phát triển bởi Ronald Rivest, Adi Shamir và Leonard Adleman vào năm 1977.
* RSA sử dụng hai khóa khác nhau: khóa công khai (public key) và khóa riêng tư (private key).

2. **Nguyên Tắc Cơ Bản:**

* **Tạo Khóa:**
  + Chọn hai số nguyên tố lớn ngẫu nhiên *p* và *q*.
  + Tính *n* = *pq*, *phi*(*n*)=(*p*−1)(*q*−1) (hàm Euler).
  + Chọn một số nguyên *e* sao cho 1<*e*< *ϕ*(*n*) và *e* là số nguyên tố cùng nhau với *ϕ*(*n*).
  + Tính *d* sao cho *d* ≡ *e* – 1 mod *ϕ*(*n*).
  + Khóa công khai là (*e*,*n*), khóa riêng tư là (*d*,*n*).

3. **Quy Trình Mã Hóa và Giải Mã:**

* **Mã Hóa:**
  + Một thông điệp *M* được biểu diễn bằng một số nguyên và 0≤*M*<*n*.
  + Mã hóa thông điệp bằng cách tính *C* ≡ *Me* mod*n*.
  + *C* là mã hóa của *M*.
* **Giải Mã:**
  + Giải mã *C* để thu được *M* ≡ *Cd* mod *n*.
  + *M* là thông điệp gốc.

4. **Ví Dụ Cụ Thể:**

* **Bước 1: Tạo Khóa:**
  + Chọn *p*=61 và *q*=53, *n*=*pq*=3233.
  + Tính *ϕ*(*n*)=60×52=3120.
  + Chọn *e*=17 (số nguyên tố cùng nhau với *ϕ*(*n*)).
  + Tính *d*≡17−1 mod 3120 => *d*=2753.
  + Khóa công khai là (17,3233), khóa riêng tư là (2753,3233).
* **Bước 2: Mã Hóa và Giải Mã:**
  + **Mã Hóa:**
    - Chọn *M*=123.
    - *C*≡12317 mod 3233=855.
    - *C* là mã hóa của *M*.
  + **Giải Mã:**
    - *M*≡8552753 mod 3233 =123.
    - *M* là thông điệp gốc.

5. **Ứng Dụng Trong Bảo Mật Thông Tin:**

* RSA thường được sử dụng để bảo vệ thông tin trong quá trình truyền tải trên mạng.
* Chữ ký số và xác thực nhận diện dựa trên RSA để đảm bảo tính toàn vẹn và xác thực thông tin.

6. **Khả Năng An Toàn:**

* Sự an toàn của RSA phụ thuộc vào khả năng khó khăn của việc phân tích một số nguyên lớn thành các thừa số nguyên tố (vấn đề NP-khó).

7. **Chú Ý:**

* Thường có các cải tiến như thêm các bước padding để tăng cường an toàn.

**Kết Luận:**

RSA là một trong những thuật toán quan trọng nhất trong lĩnh vực an toàn thông tin, cung cấp cơ sở cho nhiều ứng dụng mật mã hiện đại và được sử dụng rộng rãi trong bảo mật công nghệ thông tin.

* Sự an toàn của RSA phụ thuộc vào khả năng khó khăn của việc phân tích một số nguyên lớn thành các thừa số nguyên tố (vấn đề NP-Hard).
* Vấn Đề NP-Khó:
* Vấn đề phân tích số nguyên thành các thừa số nguyên tố được xem là một vấn đề NP - Hard.
* NP - Hard là một loại vấn đề trong lý thuyết độ phức tạp tính toán, và nếu một vấn đề thuộc loại này có thể được giải trong thời gian hợp lý, thì tất cả các vấn đề trong lớp NP cũng có thể được giải một cách hợp lý.

Github SSH keys

RSA (Rivest-Shamir-Adleman) là một thuật toán mã hóa khóa công khai được sử dụng trong nhiều ứng dụng an toàn thông tin, bao gồm cả việc bảo vệ dữ liệu truyền tải qua mạng và xác thực người dùng. Trong ngữ cảnh của GitHub và SSH keys, RSA thường được sử dụng để tạo các cặp khóa công khai và khóa riêng tư để xác thực và bảo vệ giao tiếp giữa máy cục bộ của bạn và máy chủ GitHub.

Quy trình sử dụng RSA cho SSH keys thường như sau:

1. **Tạo cặp khóa**: Một cặp khóa RSA bao gồm khóa công khai và khóa riêng tư. Khóa công khai được thêm vào máy chủ, trong khi khóa riêng tư được giữ trên máy tính cá nhân của bạn.
2. **Thêm khóa công khai vào tài khoản GitHub**: Khóa công khai sẽ được thêm vào tài khoản GitHub của bạn thông qua trang cài đặt SSH keys. Điều này giúp máy chủ GitHub xác nhận rằng máy tính của bạn có quyền truy cập vào tài khoản và có thể tương tác với các kho lưu trữ của bạn.
3. **Sử dụng khóa riêng tư để xác thực**: Khi bạn thực hiện các hoạt động như git push, git pull, hoặc git clone, máy tính cá nhân của bạn sẽ sử dụng khóa riêng tư tương ứng để xác nhận danh tính của bạn đối với máy chủ GitHub.
4. **Mã hóa thông tin gửi đi và giải mã thông tin đến**: RSA sử dụng cặp khóa để mã hóa và giải mã dữ liệu. Khóa công khai được sử dụng để mã hóa thông tin trước khi nó được gửi đi, và chỉ khóa riêng tư tương ứng được sử dụng để giải mã nó khi nó đến đích.

Với việc sử dụng RSA, thông tin truyền tải qua mạng giữa máy tính cá nhân của bạn và máy chủ GitHub được bảo vệ một cách an toàn.

HTTPs

RSA cũng được sử dụng trong giao thức HTTPS để bảo vệ quá trình truyền tải thông tin giữa trình duyệt web của người dùng và máy chủ web. Trong ngữ cảnh của HTTPS, RSA thường được sử dụng trong quá trình xác thực và truyền tải khóa để thiết lập kết nối an toàn. Dưới đây là một số bước quan trọng:

1. **Xác thực và thiết lập kết nối an toàn (SSL/TLS Handshake)**:
   * Khi người dùng truy cập một trang web sử dụng HTTPS, máy chủ sẽ gửi một chứng chỉ số chính xác chứa khóa công khai của nó đến trình duyệt của người dùng.
   * Chứng chỉ số chính xác này thường được ký bằng thuật toán RSA, nơi khóa riêng tư của máy chủ được sử dụng để ký chứng chỉ và khóa công khai tương ứng được sử dụng để xác nhận chữ ký số.
2. **Mã hóa phiên truyền tải (Session Encryption)**:
   * Sau khi xác thực, một "phiên" an toàn được thiết lập giữa trình duyệt và máy chủ. Trong quá trình này, một khóa phiên được tạo ra để mã hóa dữ liệu trong suốt quá trình truyền tải.
   * Thông thường, khóa phiên này cũng được truyền tải bằng cách sử dụng RSA. Máy chủ mã hóa khóa phiên bằng khóa công khai của trình duyệt và sau đó trình duyệt sử dụng khóa riêng tư của mình để giải mã nó.
3. **Mã hóa dữ liệu truyền tải**:
   * Khi khóa phiên đã được thiết lập, dữ liệu truyền tải giữa trình duyệt và máy chủ được mã hóa bằng khóa phiên này. Mã hóa này giúp đảm bảo rằng dữ liệu không thể bị đánh cắp hoặc hiểu đúng cách nếu nó bị đánh cắp trên đường truyền.

RSA đóng một vai trò quan trọng trong việc đảm bảo tính toàn vẹn và bảo mật của kết nối HTTPS, tăng cường sự tin cậy giữa người dùng và trang web.

DEMO

***Bước 1: Tạo Cặp Khóa Riêng và Công Khai***

# Tạo khóa riêng (private key)

***openssl genpkey -algorithm RSA -out private\_key.pem***

# Tạo khóa công khai (public key) từ khóa riêng

***openssl rsa -pubout -in private\_key.pem -out public\_key.pem\_key.pem***

Bước 2: Mã Hóa Thông Điệp

# Tạo một tệp tin chứa thông điệp cần mã hóa

***echo "Hello, this is a secret message." > plaintext.txt***

# Mã hóa thông điệp sử dụng khóa công khai

***openssl rsautl -encrypt -inkey public\_key.pem -pubin -in plaintext.txt -out encrypted.bin***

Bước 2.5:

***cat encrypted.bin***

Bước 3: Giải Mã Thông Điệp

# Giải mã thông điệp sử dụng khóa riêng

***openssl rsautl -decrypt -inkey private\_key.pem -in encrypted.bin -out decrypted.txt***

Bước 4: Kiểm Tra Kết Quả

***cat decrypted.txt***