Created by Lê Trọng Nghĩa

PRIME AND DIFFIE-HELLMAN KEY EXCHANGE

IC G14 PJRP-002

MODULAR EXPONENTIATION, CSPRNG (RANDOM NUMBER GENERATION), SAFE PRIME, PRIMALITY TEST

The report is part of the first group project of the Introduction to Cryptography course, a module provided by the University of Natural Sciences, VNU-HCM.

CQ2022/22, Introduction to Cryptography, University of Natural Sciences, VNU-HCM

INTRODUCTION TO CRYTOGRAPHY

A PROJECT REPORT

IC G14 PJRP-002

RESPONSIBILITY

GROUP 14

Member 01: 22120257

Đinh Lê Gia Như

Member 02: 22120226

Lê Trọng Nghĩa

Member 03: 22120192

Nguyễn Đăng Long

FACULTY OF INFORMATION TECHNOLOGY

UNIVERSITY OF SCIENCE, VNU-HCM CITY UNIVERSITY

HO CHI MINH CITY, VIET NAM COUNTRY

October, 2024

Language: English, Vietnamese

Tóm tắt

Tài liệu IC G14 PJRP-002 này là tài liệu báo cáo thuộc Đồ án Lab01 (Dự án nhỏ Lab01), khóa học Nhập môn mã hóa – mật mã, bộ môn Công nghệ trí thức, Trường Đại học Khoa học Tự Nhiên, Đại học Quốc Gia Thành phố Hồ Chí Minh, Việt Nam.

IC G14 PJRP-002 báo cáo các nội dung liên quan đến các vấn đề tính toán với lũy thừa module, lý thuyết số ngẫu nhiên trong mật mã học, lý thuyết số nguyên tố lớn an toàn (safe prime), và thuật toán kiểm tra số nguyên tố.

Các tài liệu, văn bản có liên quan đến Đồ án Lab01 bao gồm: Báo cáo IC G14 PJRP-001, Báo cáo IC G14 PJRP-002, Báo cáo IC G14 PJRP-003 được lưu trữ trong thư mục project_01_report và nộp bài tập đồ án trên trang môn học (Moodle) của Khoa Công nghệ thông tin, Trường Đại học Khoa học Tự nhiên, Đại học Quốc gia Thành phố Hồ Chí Minh, Việt Nam. Ngoài ra, mã nguồn của đồ án đã thực hiện được lưu trữ trong thư mục project_01_source có kèm theo với các tài liệu báo cáo này khi kết thúc đồ án.

Từ khóa có liên quan: crytography; safe prime; sophia germain prime; primality test; miller-rabin primality test; prime; OEIS; CSPRNG; RSA; Fermat algorithm; c++; cplusplus; random.

Lời cảm ơn

Người thực hiện bài báo cáo này, tôi là Lê Trọng Nghĩa, trân trọng ghi nhận đóng góp và đánh giá cao đến những đóng góp của các cá nhân Đinh Lê Gia Như, Lê Trọng Nghĩa, Nguyễn Đăng Long là thành viên của nhóm 14 thực hiện đồ án Lab01 có các nội dung liên quan đến các vấn đề được nêu có trong tài liệu này bao gồm các vấn đề về công nghệ thông tin, mật mã và bảo mật.

Tác giả bài báo cáo cũng xin cảm ơn các hướng dẫn, gợi ý về đồ án, và các giải đáp thắc mắc từ phía giảng viên phụ trách khóa học, người quản lý dự án có liên quan.

Nguồn tham khảo

Nguồn tài liệu tham khảo:

a. Tiêu chuẩn hóa: OEIS, OEIS: A005385, OEIS: A059327, OEIS: A059394, OEIS: A059395, OEIS: A059452, OEIS: A075705, OEIS: A075706, OEIS: A075707, OEIS: A227853, OEIS: A014233, cppreference

[01] OEIS https://oeis.org/
[02] OEIS: A005385 https://oeis.org/A005385
[03] OEIS: A059327 https://oeis.org/A059327
[04] OEIS: A059394 https://oeis.org/A059394
[05] OEIS: A059395 https://oeis.org/A059395
[06] OEIS: A059452 https://oeis.org/A059452
[07] OEIS: A075705 https://oeis.org/A075705
[08] OEIS: A075706 https://oeis.org/A075706
[09] OEIS: A075707 https://oeis.org/A075707
[10] OEIS: A227853 https://oeis.org/A227853
[11] OEIS: A014233 https://oeis.org/A014233
[12] cppreference: https://en.cppreference.com/w/cpp/standard_library
[13] cppreference: https://en.cppreference.com/w/cpp/numeric/random

b. Diễn đàn: GeeksforGeeks, Omni Calculator, Wikipedia, Linkedin, CryptoBook.Nakov, PlanetMath

[14] GeeksforGeeks: https://www.geeksforgeeks.org/modular-exponentiation-power-in-modular-arithmetic/
[15] GeeksforGeeks: https://www.geeksforgeeks.org/primality-test-set-3-miller-rabin/
[16] GeeksforGeeks: https://www.geeksforgeeks.org/introduction-to-primality-test-and-school-method/
[17] Omni Calculator: https://www.omnicalculator.com/math/power-modulo
[18] Wikipedia: https://en.wikipedia.org/wiki/Miller%E2%80%93Rabin_primality_test
[19] Wikipedia: https://en.wikipedia.org/wiki/Modular_exponentiation
[20] Linkedin: https://www.linkedin.com/pulse/cryptographically-secure-pseudo-random-number-csprng-ainapurapu-vfwbc/
[21] CryptoBook.Nakov: https://cryptobook.nakov.com/secure-random-generators/secure-random-generators-csprng
[22] PlanetMath: https://planetmath.org/SafePrime

c. Tài liệu tham khảo:

- *An Introduction to Mathematical Cryptography*
- *Handbook of Applied Cryptography* của Menezes, van Oorschot, và Vanstone
- *Cryptography and Network Security: Principles and Practice* của William Stallings

Table of Contents

1 Lũy thừa Mô-đun

1.1 Tổng quát

Lũy thừa mô-đun (hay phép lũy thừa mô-đun) là phép tính lũy thừa của một số $result$ lên một số mũ $exponent$ rồi lấy kết quả theo mô-đun m. Kết quả là phần dư của phép chia $base^{exponent}$ cho m. Tổng quát phép tính như sau:

$$result = (base^{exponent})\ mod\ m$$

Phép lũy thừa mô-đun được sử dụng nhiều trong lý thuyết số và mật mã học vì tính hiệu quả và bảo mật của nó. Thay vì tính trực tiếp $base^{exponent}$ rồi chia lấy dư, có một thuật toán tối ưu hơn là "lũy thừa bình phương và nhân" (exponentiation by squaring) để giảm số phép nhân cần thiết. Cách này đặc biệt hiệu quả khi số $exponent$ đủ lớn.

1.2 Thuật toán

Dưới đây là một ví dụ về mã giả thuật toán (pseudocode) cách hoạt động tính toán với lũy thừa mô-đun:

```
function modular_exponentiation(base, exponent, mod) is
    if mod = 1 then
        return 0
    result = 1
    base = base mod mod
    while exponent > 0 do
        if (exponent mod 2 = 1) then
            result = (result * base) mod mod
        exponent = exponent >> 1
        base = (base * base) mod mod
return result
```

☞ Nhận xét: Mã giả thuật toán trên được đánh giá có độ phức tạp O(log(n)).

1.3 Ứng dụng

1.3.1 Ứng dụng trong mật mã học

- **Hệ thống mã hóa công khai (Public-Key Cryptography)**: Lũy thừa mô-đun là nền tảng của các hệ thống mã hóa công khai như RSA và thuật toán trao đổi khóa Diffie-Hellman. Trong RSA, các cặp khóa được tạo ra từ các phép tính lũy thừa mô-đun với số mũ lớn và mô-đun là tích của hai số nguyên tố lớn. Khi một thông điệp được mã hóa bằng khóa công khai, chỉ có khóa bí mật tương ứng mới có thể giải mã được thông điệp đó, giúp bảo vệ tính bảo mật cho thông tin truyền tải.
- **Trao đổi khóa Diffie-Hellman**: Trong phương pháp trao đổi khóa Diffie-Hellman, hai bên có thể chia sẻ một khóa bí mật mà không cần phải trao đổi trực tiếp khóa này. Phép tính lũy thừa mô-đun được dùng để tạo ra một giá trị chung từ hai số ngẫu nhiên của mỗi bên, đồng thời đảm bảo rằng khóa chia sẻ là duy nhất và chỉ có hai bên biết.
- **Chữ ký số (Digital Signatures)**: Kỹ thuật lũy thừa mô-đun cũng được ứng dụng để tạo ra chữ ký số, giúp xác thực tính chính xác và nguồn gốc của thông điệp. Các hệ thống chữ ký số, như thuật toán DSA (Digital Signature Algorithm), sử dụng lũy thừa mô-đun để tạo chữ ký độc nhất cho một thông điệp. Người nhận có thể kiểm tra chữ ký này bằng khóa công khai của người gửi mà không cần biết nội dung của khóa bí mật.

1.3.2 Ứng dụng ngoài mật mã học

- **Lý thuyết số**: Được sử dụng để giải quyết các bài toán chia lấy dư và các phương pháp phân tích số học.
- **Định danh và xác thực**: Được sử dụng để tạo các mã thông báo ngẫu nhiên trong hệ thống bảo mật web, qua đó giúp các trang web xác minh danh tính người dùng một cách an toàn.

2 Số ngẫu nhiên trong mật mã học

2.1 Tính ngẫu nhiên trong mật mã học

Số ngẫu nhiên là một thành phần quan trọng trong nhiều hệ thống mật mã, được sử dụng để tạo khóa, số nonce, và các giá trị không thể đoán trước khác. Tính ngẫu nhiên đảm bảo rằng các khóa và mã thông báo không dễ bị dự đoán hoặc sao chép.

2.2 Bộ sinh số ngẫu nhiên mật mã (CSPRNG)

Trong mật mã học, các số ngẫu nhiên không chỉ cần phân phối đều mà còn phải không thể đoán trước. Các CSPRNG (Cryptographically Secure Random Number Generators) có khả năng sinh ra các số ngẫu nhiên an toàn về mặt mật mã. Để đáp ứng yêu cầu về độ an toàn, CSPRNG phải đảm bảo các yếu tố sau:

- **Không thể dự đoán**: Không thể dự đoán số tiếp theo trong chuỗi nếu không biết trạng thái bên trong của bộ sinh.
- **Chống tấn công**: Đảm bảo rằng kẻ tấn công không thể tái tạo hoặc dự đoán chuỗi số ngẫu nhiên dựa trên các giá trị đã biết.

2.3 Hàm sinh số ngẫu nhiên an toàn trên máy tính

Để đảm bảo tính ngẫu nhiên an toàn trong các ứng dụng mật mã học, chúng ta sẽ triển khai một hàm sinh số ngẫu nhiên trong *C++* với các đặc điểm đảm bảo độ bảo mật cao.

Các bước thực hiện:

1. **Khởi tạo nguồn entropy**: Sử dụng nguồn entropy từ hệ thống (chẳng hạn từ thời gian thực hoặc các thao tác ngẫu nhiên từ phần cứng). Đây là bước cần thiết để có một nguồn ngẫu nhiên không thể dự đoán trước.

2. **Tạo hàm sinh số ngẫu nhiên an toàn bằng thư viện tiêu chuẩn**:
    - Sử dụng thư viện `<random>` trong C++ với các đối tượng như `std::random_device` và `std::mt19937_64`.
    - `std::random_device` là thiết bị tạo số ngẫu nhiên không thể đoán trước, có thể dùng để seed cho `std::mt19937_64`, một bộ sinh số ngẫu nhiên hiệu quả.
3. **Viết mã triển khai hàm**:

```
cpp
Copy code
#include <iostream>
#include <random>
#include <chrono>

// Hàm sinh số ngẫu nhiên an toàn
unsigned long long generateSecureRandomNumber() {
    std::random_device rd;
    std::mt19937_64 gen(rd());
    std::uniform_int_distribution<unsigned long long> dist(0, ULLONG_MAX);
    return dist(gen);
}

int main() {
    unsigned long long randomNumber = generateSecureRandomNumber();
    std::cout << "Số ngẫu nhiên an toàn: " << randomNumber << std::endl;
    return 0;
}
```

Giải thích mã lệnh:

- `std::random_device rd;`: Thiết bị sinh số ngẫu nhiên dựa trên nguồn entropy từ phần cứng.
- `std::mt19937_64 gen(rd());`: Bộ sinh số ngẫu nhiên được khởi tạo với nguồn entropy từ `random_device`.
- `std::uniform_int_distribution<unsigned long long> dist(0, ULLONG_MAX);`: Thiết lập khoảng giá trị ngẫu nhiên.

2.4 Ứng dụng

2.4.1 Ứng dụng trong mật mã học

- **Tạo khóa mã hóa an toàn**: Các khóa mã hóa được tạo ra từ CSPRNG có tính chất ngẫu nhiên cao và không dễ dàng đoán được. Chúng được sử dụng trong cả mã hóa đối xứng (AES) và mã hóa công khai (RSA, ECC). Vì các khóa ngẫu nhiên này rất khó bị phá, chúng đảm bảo tính bảo mật và tính toàn vẹn của các hệ thống mã hóa thông tin.
- **Tạo số nonce trong các giao thức bảo mật**: Nonce là số ngẫu nhiên dùng một lần và được sử dụng rộng rãi trong các giao thức bảo mật như SSL/TLS và IPsec để ngăn chặn tấn công lặp lại (replay attacks). Số nonce này thường được sinh ra từ CSPRNG, đảm bảo tính không trùng lặp và không thể dự đoán.
- **Xác thực hai yếu tố (Two-Factor Authentication - 2FA)**: CSPRNG cũng đóng vai trò quan trọng trong việc sinh các mã xác thực một lần (OTP) được sử dụng trong hệ thống 2FA. Mỗi mã OTP được tạo ra là duy nhất và chỉ có giá trị trong một thời gian ngắn, giúp tăng cường bảo mật cho các tài khoản người dùng.

2.4.2 Ứng dụng ngoài mật mã học

- **Tạo mã OTP trong ngân hàng và tài chính**: Các tổ chức tài chính sử dụng OTP sinh từ CSPRNG để đảm bảo rằng các giao dịch trực tuyến được thực hiện bởi người dùng hợp pháp.
- **Trò chơi và mô phỏng ngẫu nhiên**: Các trò chơi máy tính sử dụng CSPRNG để tạo các yếu tố ngẫu nhiên như bài xáo, số xúc xắc và kết quả của các sự kiện để tạo ra các trò chơi không thể đoán trước.

3 Số nguyên tố an toàn

3.1 Tổng quan

Với một số nguyên tố q , nếu $p = \frac{q-1}{2}$, trong đó p là một số nguyên tố khác, thì q là một số nguyên tố an toàn. Từ đó suy ra $q - 1 = 2p$ (và do đó là một số bán nguyên tố) và p là một số nguyên tố Sophie Germain.

Mọi số nguyên tố an toàn đều khớp với một số nguyên tố Sophie Germain theo cách này, và một số số nguyên tố an toàn cũng là số nguyên tố Sophie Germain, khớp với một số nguyên tố an toàn khác. Một tập hợp các số nguyên tố được liên kết theo cách này được gọi là chuỗi Cunningham loại một. Ví dụ, 2, 5, 11, 23, 47. Với tiêu chuẩn của Pocklington, người ta có thể chứng minh tính nguyên tố của một số nguyên tố an toàn nếu trước tiên người ta chứng minh được tính nguyên tố của số nguyên tố Sophie Germain khớp với nó.

Một số số nguyên tố an toàn đầu tiên là 5, 7, 11, 23, 47, 59, 83, 107, 167, 179, 227, … Những số này được liệt kê trong A005385 của OEIS của Sloane. Tính đến năm 2010, số nguyên tố an toàn lớn nhất được biết đến là $183027 * 2^{265441} - 1$

Các số nguyên tố an toàn đã được sử dụng trong nhiều phương pháp mật mã khác nhau, nhưng tính an toàn khi sử dụng chúng không chỉ phụ thuộc vào các tính chất toán học này mà còn phụ thuộc vào việc chúng có đủ lớn để các bội số của chúng không thể được phân tích trong một khoảng thời gian hợp lý bằng máy tính hiện đại hay không. [2]

3.2 Tiêu chuẩn hóa và cộng đồng

Một số trang web thư viện lưu trữ, cơ sở dữ liệu, bài nghiên cứu có liên quan đến số nguyên tố an toàn (Safe Prime), ngoài ra cũng như số nguyên tố Sophia Germain (Sophia Germain Prime) phải kể đến như Wolfram MathWorld, Wolfram Alpha, NIST DLMF, OEIS.

Trong dự án Lab01 này, dữ liệu về các giá trị số nguyên tố phục vụ cho việc thử nghiệm chương trình triển khai trao đổi khóa Diffie-Hellman được lấy từ OEIS bao gồm các cơ sở OEIS được nêu ở phần nguồn tài liệu tham khảo có nêu trong bản báo cáo này.

3.3 Ứng dụng

3.3.1 Ứng dụng trong mật mã học

- **Thuật toán trao đổi khóa Diffie-Hellman**: Số nguyên tố an toàn giúp cải thiện tính bảo mật cho quá trình trao đổi khóa Diffie-Hellman, vì việc bẻ khóa một số nguyên tố an toàn cần thời gian tính toán cao hơn nhiều so với các số nguyên tố thông thường. Trong Diffie-Hellman, số nguyên tố an toàn giúp đảm bảo rằng chỉ có hai bên tham gia trao đổi khóa mới có thể tính toán ra được khóa chung.
- **Chứng minh mật mã zero-knowledge (Zero-Knowledge Proofs)**: Số nguyên tố an toàn còn được dùng trong các hệ thống chứng minh không tiết lộ thông tin. Zero-knowledge proofs là một phương pháp giúp một bên chứng minh với bên kia rằng mình biết một thông tin cụ thể mà không tiết lộ thông tin đó. Điều này rất quan trọng trong các giao dịch tài chính hoặc hệ thống bảo mật nơi mà tính riêng tư cần được bảo vệ tuyệt đối.

3.3.2 Ứng dụng ngoài mật mã học

- **Phân tích dữ liệu lớn**: Số nguyên tố an toàn còn được sử dụng trong các phương pháp xử lý và phân tích dữ liệu lớn, nơi tính năng ngẫu nhiên và an toàn của số nguyên tố giúp tối ưu hoá việc lưu trữ và truy xuất dữ liệu.

4 Thuật toán kiểm tra số nguyên tố

4.1 Tổng quan

Trong mật mã học, kiểm tra tính nguyên tố của một số là điều kiện cơ bản để tạo ra các khóa mã hóa an toàn. Các thuật toán mã hóa khóa công khai như RSA yêu cầu các số nguyên tố lớn để tạo ra các cặp khóa bảo mật. Tính chất cơ bản của số nguyên tố giúp đảm bảo rằng chỉ có các khóa bí mật đã biết mới có thể giải mã dữ liệu

4.2 Một số thuật toán thường gặp

Một số thuật toán kiểm tra số nguyên tố phổ biến bao gồm:

- **Phương pháp thử chia (Trial Division)**: Đây là phương pháp đơn giản nhất, trong đó ta kiểm tra tính chia hết của số n cho các số từ 2 đến n. Nếu không có số nào chia hết n, thì n là số nguyên tố. Tuy nhiên, phương pháp này không hiệu quả với các số rất lớn.
    - **Độ phức tạp**: Phương pháp này có độ phức tạp là $O(\sqrt{n})$ do chỉ cần kiểm tra tính chia hết đến n . Với các số lớn, n cũng tăng nhanh khiến thời gian kiểm tra tăng đáng kể. Vì vậy, phương pháp này thường chỉ phù hợp để kiểm tra các số nhỏ.

- **Thuật toán Fermat (Fermat's Little Theorem)**: Theo định lý nhỏ của Fermat, nếu n là số nguyên tố, thì với bất kỳ số nguyên nào a sao cho $1 < a < n - 1$, $a^{n-1} \bmod n = 1$. Nếu tồn tại a mà $a^{n-1} \bmod n \neq 1$, thì n không phải là số nguyên tố. Tuy nhiên, phương pháp này có thể gặp sai lầm với một số số giả nguyên tố (các số Carmichael), nên thường được sử dụng kết hợp với các phương pháp khác.
    - **Độ phức tạp**: Để tính $a^{n-1} \bmod n$, thuật toán sử dụng phương pháp lũy thừa nhanh với độ phức tạp $O(\log(n))$. Đây là một cách kiểm tra nhanh, nhưng do có thể gặp phải các số Carmichael nên độ tin cậy của thuật toán không cao bằng các phương pháp tiên tiến hơn như Miller-Rabin.

- **Thuật toán Miller-Rabin**: Đây là phương pháp kiểm tra xác suất, có thể xác định liệu một số có phải là nguyên tố với độ chính xác cao. Phương pháp Miller-Rabin sử dụng biến đổi số học để phân tích cấu trúc của n, dựa trên việc phân tích số mũ. Với độ tin cậy cao, thuật toán này là lựa chọn phổ biến cho các ứng dụng mật mã, giúp tiết kiệm thời gian so với việc thử chia từng số.
    - **Độ phức tạp**: Thuật toán Miller-Rabin có độ phức tạp là $O(k.\log^3(n))$, trong đó k là số lần kiểm tra với các giá trị khác nhau của a. Độ phức tạp này đến từ việc phải tính nhiều phép lũy thừa mô-đun để xác minh các điều kiện của thuật toán. Việc chọn k càng lớn sẽ tăng độ chính xác, tuy nhiên sẽ tăng thêm thời gian tính toán. Thuật toán Miller-Rabin rất hiệu quả khi áp dụng cho các số lớn và ít gặp phải các số giả nguyên tố như trong thuật toán Fermat.

4.3 Ứng dụng

4.3.1 Ứng dụng trong mật mã học

- **RSA và các hệ thống mã hóa công khai**: RSA yêu cầu hai số nguyên tố lớn để tạo ra cặp khóa công khai và bí mật. Các thuật toán kiểm tra số nguyên tố giúp xác định các số nguyên tố lớn này. Với mỗi số nguyên tố lớn được chọn, hệ thống sẽ đảm bảo rằng chỉ người có khóa bí mật mới có thể giải mã thông điệp, giúp thông tin được bảo mật.
- **Blockchain và xác nhận giao dịch**: Các hệ thống blockchain như Bitcoin cần các phép tính bảo mật để xác nhận các giao dịch. Trong quá trình này, các số nguyên tố lớn đóng vai trò quan trọng trong việc mã hóa dữ liệu và xác minh tính toàn vẹn của các khối (blocks) trong chuỗi khối (blockchain).
- **Thuật toán chữ ký số**: Các thuật toán chữ ký số như DSA yêu cầu các số nguyên tố lớn để tạo ra các chữ ký số an toàn. Việc kiểm tra số nguyên tố là điều cần thiết để đảm bảo rằng các chữ ký số không bị giả mạo và có tính an toàn cao.

4.3.2 Ứng dụng ngoài mật mã học

- **Xử lý số liệu khoa học và thống kê**: Các thuật toán kiểm tra số nguyên tố còn được áp dụng trong các nghiên cứu toán học và thống kê, giúp tìm ra các mẫu số liệu đặc biệt và kiểm tra tính chính xác của dữ liệu.