Nhập môn Phân tích độ phức tạp thuật toán

Báo cáo đồ án Seminar

Chủ đề: Bài toán Discrete Logarithms trên các hệ mã hóa công khai

I. Thành viên nhóm:

1. 18120019 – Nguyễn Hoàng Dũng
2. 18120052 – Lê Hạnh Linh
3. 18120134 – Nguyễn Hồ Thăng Long

II. Giới thiệu vấn đề:

Trên các hệ mã hóa công khai, những bài toán khó trong số học được chọn để làm cơ sở cho việc tạo ra các bộ khóa cá nhân và công khai.

Một người bất kì A có thể gửi một thông điệp bí mất đến cho người B, bằng cách sử dụng khóa công khai của người B và mã hóa văn bản bằng khóa công khai đó.

Khóa công khai và khóa cá nhân được lựa chọn trên cơ sở số học sao cho với bất kì giá trị nào của khóa công khai, việc tìm ra khóa cá nhân tương ứng là không khả thi trong thời gian ngắn (thời gian mà văn bản được mã hóa vẫn còn giá trị thông tin)

Ngoài việc sử dụng bài toán Phân tích Nguyên Tố làm bài toán khó (như trong hệ mã RSA), nhóm chúng em muốn tìm hiểu về **bài toán log rời rạc** và ứng dụng của nó trong việc tạo khóa công khai và khóa cá nhân trên các hệ mã hóa công khai.

III. Bài toán quan tâm:

Bài toán **log rời rạc** trong mã hóa có thể phát biểu đơn giản như sau. Cho phương trình đồng dư:

$$g^x = y \ (mod \ \mathrm{N})$$

Trong đó, ta sẽ cố định N là một số nguyên tố, cơ số g và số mũ x là các số ngẫu nhiên trong khoảng 2 đến N – 1, và y là kết quả đồng dư trên modulo N của phép lũy thừa g cho x. Ta sẽ xét ví dụ như sau:

$$1337^{3806} = 27 \ (\mathrm{mod}\ 13729)$$

Bài toán log rời rạc nói rằng, nếu ta tìm được các số thỏa mãn như trên, sau đó giấu kín giá trị x (số 3806) và công khai bộ số (1337, 27, 13729), khi đó việc tìm x để thỏa mãn phương trình sau, là một bài toán khó:

$$1337^x = 27 \ (\mathrm{mod}\ 13729)$$

Như vậy, áp dụng của bài toán log rời rạc vào mã hóa công khai sẽ như sau:

Alice	Bob
Parameters: p, g	
$A = \mathrm{random}()$ $a = g^A \pmod p$	$\mathrm{random}() = B$ $g^B \pmod p = b$
$a \longrightarrow$ $\longleftarrow b$	
$K = g^{BA} \pmod p = b^A \pmod p$	$a^B \pmod p = g^{AB} \pmod p = K$
$\longleftarrow E_K(data) \longrightarrow$	

Hình 1: Thuật toán Diffie - Hellman

Nhờ tính chất kết hợp, cơ số g sau khi được lũy thừa lần lượt với A và B sẽ tạo ra được khóa K chung bí mật. Điểm mấu chốt: từ a và b công khai, một người lạ bất kì không thể (khó) tìm ra được số mũ A hoặc B, để suy ra được khóa K.

IV. Hướng tiếp cận:

Như vậy, để tìm hiểu được độ khó của bài toán log rời rạc, chúng em sẽ tiếp cận bằng cách khảo sát một số ví dụ như sau:

VD1: xét $g = 2$

$2^x \equiv 5 \pmod{19}$

x	1	2	3	4	5	6	7	8	9
2^x	2	4	8	16	13	7	14	9	18

10	11	12	13	14	15	16	17	18
17	15	11	3	6	12	5	10	1

VD2: xét $g = 8$

$8^x \equiv 5 \pmod{19}$

x	1	2	3	4	5	6
8^x	8	7	18	11	12	1

Bằng 2 ví dụ minh họa, ta có thể thấy tùy vào cách chọn g, mà bài toán log rời rạc có thể có nghiệm, hoặc không có nghiệm. Tuy nhiên, sẽ tồn tại cách chọn g, sao cho với mọi y, bài toán log rời rạc luôn có nghiệm (VD1).

Phép lũy thừa chính là một hình ảnh của việc ta áp dụng phép nhân nhiều lần liên tiếp lên chính số g cho đến khi nhận được kết quả đồng dư là 1. Khi đó chu kỳ đồng dư của phép lũy thừa đã tạo ra các kết quả phân biệt.

Thứ tự xuất hiện của các kết quả này là không có quy luật. Do đó để tìm được số mũ x tương ứng với kết quả cho trước nào đó, ta chỉ có thể kiểm tra hết các trường hợp. Chu kỳ đồng dư càng lớn, nghĩa là không gian tìm kiếm x cho bài toán log rời rạc càng lớn.

Theo định lý Fermat nhỏ, với số nguyên tố P cho trước, ta có với mọi g:

$$g^{P-1} \equiv 1 (\mathrm{mod}\, P)$$

Tuy nhiên, ta cần chọn g để đảm bảo P – 1 là giá trị nhỏ nhất thỏa mãn đẳng thức trên. Khi đó với số nguyên tố P lớn (khoảng 1024 bit), và cho trước kết quả y nằm trên chu kỳ đồng dư, việc tìm x là không khả thi trong thời gian ngắn. Những giá trị như vậy của g được gọi là **phần tử sinh**.

Có thể nhận thấy rằng, nếu P – 1 chưa phải là giá trị nhỏ để thỏa mãn đẳng thức, vậy thì giá trị nhỏ nhất đó phải là một ước của P – 1, do chu kỳ phải được lặp lại số nguyên lần để có thể kết thúc tiếp tại lũy thừa P – 1.

Như vậy để kiểm tra một số g bất kì có là phần tử sinh hay không. Ta sẽ xét kết quả đồng dư khi lũy thừa g với lần lượt các ước của P – 1.

V. Mô tả thuật toán:

1. Thuật toán tạo số nguyên tố an toàn:

Việc tạo ra một số giả nguyên tố q (1024 bit) với độ chính xác cao có thể được thực hiện bằng thuật toán Miller – Rabin. Nhóm em sẽ sử dụng kết quả từ thuật toán Miller – Rabin để tìm số nguyên tố P có dạng 2q × i + 1.

Những số nguyên tố P có dạng 2q + 1 được gọi là những số nguyên tố an toàn (Safe Prime). Tuy nhiên để tìm được một số nguyên tố có điều kiện nghiêm ngặt như vậy là không dễ. Do đó nếu 2q + 1 không thỏa mãn kiểm tra Miller – Rabin, chúng em sẽ kiểm tra tiếp với các bội của 2q.

Mục đích của việc tìm kiếm số nguyên tố P như vậy là để P – 1 có thừa số nguyên tố lớn là q, có thể giúp làm tăng độ khó của bài toán log rời rạc bằng cách gây khó dễ cho các thuật toán tìm log rời rạc như Pohlig – Hellman.

Algorithm 1 Generate Safe Prime

Input: number of bits n
Output: prime p with large factor q

```
1: function GetSafePrime
2:     q ← GetPrime(n)
3:     i ← 1
4:     while True do
5:         p ← 2q * i + 1
6:         i ← i + 1
7:         if IsPrime(p) then break
8: return p, q
```

2. Thuật toán chọn phần tử sinh:

Việc chọn phần tử sinh sẽ được thực hiên bằng cách kiểm tra số g bất kì trong khoảng (2, P – 1), sử dụng tính chất về độ dài chu kỳ là ước của P – 1. Tuy nhiên, ta sẽ chỉ cần kiểm tra các lũy thừa của g với ước của (P – 1) / q.

VD: Xét $\boldsymbol{p} = 103 \Rightarrow \boldsymbol{p} - 1 = 2 \times 3 \times 17$

Có: $\boldsymbol{p} - 1$ có q = 17 là thừa số nguyên tố lớn

Algorithm 2 Get Generator

Input: prime p with large factor q
Output: a number g with large order

```
function GetGenerator
    k ← (p − 1)/q
    while True do
        g ← RANDOM(2, p − 2)
        if g^k ≠ 1 (mod p) then break
return g
```

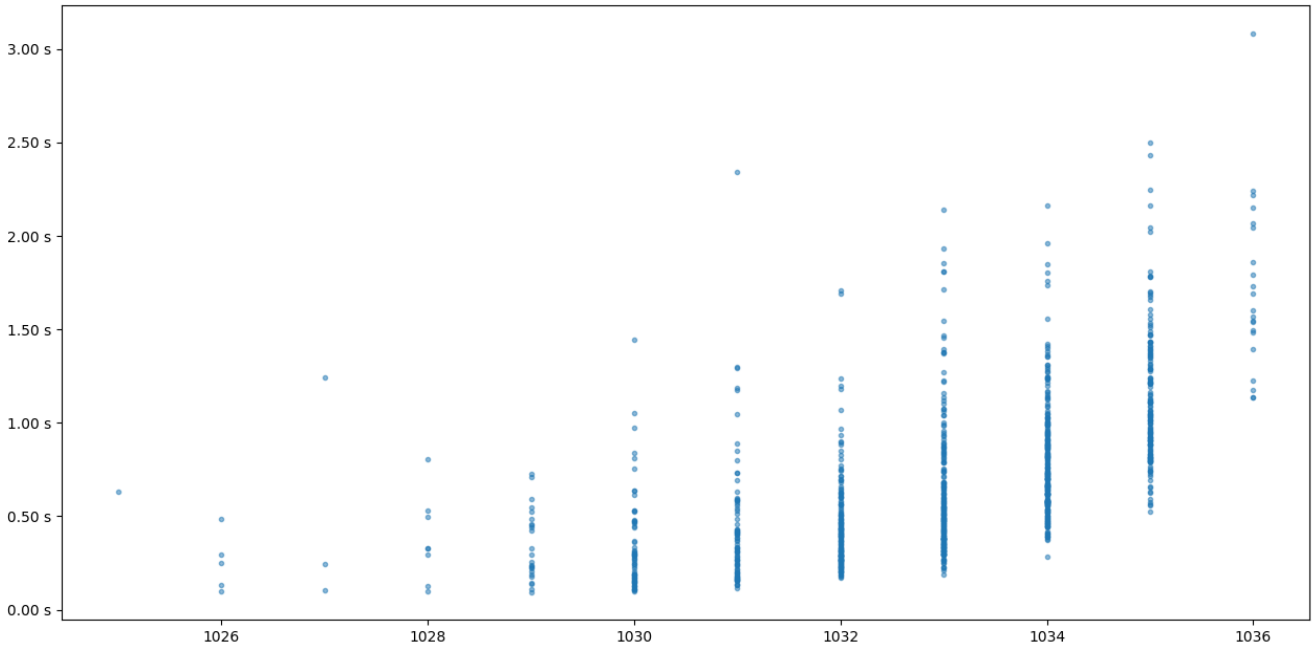
TH: $g^6 = 1 \ (mod\ p)$
⇒ không chọn g

TH: $g^6 \neq 1 \ (mod\ p)$
⇒ $g^2 \neq 1,\ g^3 \neq 1$
⇒ $g^q = 1$ or $g^{2q} = 1$ or $g^{3q} = 1$
or $g^{6q} = g^{p-1} = 1$
⇒ chọn g

VI. Đánh giá độ phức tạp:

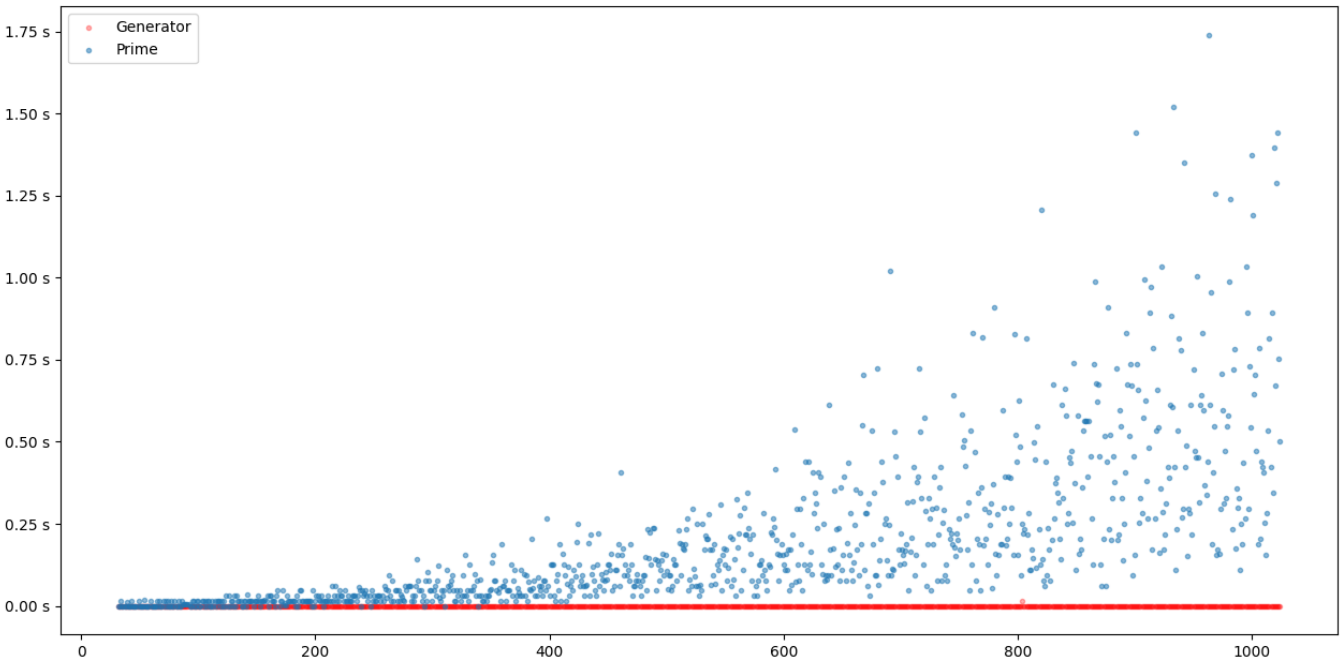
Việc đánh giá chính xác độ phức tạp cho 2 thuật toán trên cần nhiều kiến thức, định lý về số nguyên tố như: phân bố số nguyên tố, khoảng cách giữa 2 số nguyên tố nên chúng em sẽ chỉ tập trung thực hiện các thí nghiệm thống kê để có thể đưa ra những nhận xét tổng quan nhất.

1. Thống kê thực nghiệm cho thuật toán tạo số nguyên tố an toàn:



Thống kê cho thấy trên 1000 lần sinh q nguyên tố có 1024 bit, chỉ có 1 lần duy nhất là 2q + 1 cũng là số nguyên tố. Những trường hợp còn lại có i dao động trong khoảng từ 6 đến 10 bit thì 2q × i + 1 sẽ là số nguyên tố. Như vậy P – 1 có thừa số nguyên tố rất lớn là q (1024 bit)

2. Thống kê thực nghiệm cho thuật toán tìm phần tử sinh:



Thống kê cho biết thời gian trung bình khi sinh số nguyên tố an toàn P có độ lớn từ 32 bit đến 1024 bit sẽ nằm trong khoảng dưới 1s. Mặt khác, thống kê cho thấy việc tìm phần tử sinh g tương ứng có thời gian không đáng kể so với việc tìm số nguyên tố P.

VII. Tài liệu tham khảo:

1. Cryptography and Network Security - William Stallings
2. Understanding Cryptography - Christof Paar · Jan Pelzl
3. An Introduction to Mathematical Cryptography - Jeffrey Hoffstein · Jill Pipher · Joseph H. Silverman