ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC
******

PHƯƠNG PHÁP LƯỚI

ĐỒ ÁN I

Chuyên ngành: Toán Tin

Giảng viên hướng dẫn : TS. NGÔ QUỐC HOÀN Chữ kí của GVHD
Sinh viên thực hiện : NGUYỄN ĐÌNH ANH
Mã số sinh viên : 20206111
Lớp : Toán Tin 03 - K65

Hà Nội, 07 - 2023

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

1. Mục đích và nội dung của đồ án

2. Kết quả đạt được

3. Ý thức làm việc của sinh viên

Hà Nội, ngày , tháng , năm 2023

Giảng viên hướng dẫn

(Ký và ghi rõ họ tên)

TÓM TẮT NỘI DUNG ĐỒ ÁN

Đồ án này trình bày một số kiến thức cơ sở liên quan đến lý thuyết lưới trước khi trình bày và xây dựng thuật toán giảm lưới LLL. Tiếp đó, đồ án sẽ tổng hợp các vấn đề về 2 hệ mật mã phổ biến RSA Và NTRU. Cụ thể là, đồ án này sẽ tìm hiểu cách hoạt động của hai hệ mật mã trên, từ tạo khóa, mã hóa đến việc giải mã cũng như các áp dụng lý thuyết lưới trong các cuộc tấn công vào 2 hệ mật mã này. Cuối cùng, đồ án thực hiện triển khai xây dựng chương trình thuật toán LLL, và các cuộc tấn công vào hệ mật mã RSA và NTRU. Các kết quả thực nghiệm được cài đặt trên phần mềm SageMath.

Hà Nội, ngày , tháng , năm 2023

Sinh viên thực hiện đồ án

(Ký và ghi rõ họ tên)

LỜI CẢM ƠN

Để hoàn thành đồ án này, em xin chân thành gửi lời cảm ơn chân thành đến các thầy, các cô viện Toán ứng dụng và Tin học đã giảng dạy và giúp em tiếp thu những nền tảng quý giá phục vụ cho bài báo cáo này.

Đặc biệt, em xin được gửi lời cảm ơn sâu sắc tới TS. Ngô Quốc Hoàn đã tận tình hướng dẫn em hoàn thiện bài báo cáo này. Từ những ngày mới được giao đề tài cho đến lúc hoàn thành, em luôn được thầy nhắc nhở và theo dõi tiến trình hoàn thành Đồ Án 1. Thầy cũng đã sắp xếp thời gian bận rộn của mình để gặp gỡ trao đổi với sinh viên về những khó khăn, những thắc mắc trong quá trình tìm hiểu đề tài và truyền tải những kiến thức bổ ích đến em.

Em xin chân thành cảm ơn!

Nguyễn Đình Anh

Mục lục

Bảng kí hiệu

N	tập hợp số tự nhiên
Z	tập hợp các số nguyên
R	tập hợp số thực
$\|.\|$	Chuẩn Euclide
$x.y$	tích vô hướng của vector x và y
$det(.)$	tính định thức
$\otimes$	phép nhân trên vành đa thức

Lời mở đầu

Mật mã học có một lịch sử phát triển hàng ngàn năm. Từ những phương pháp đơn giản như thay thế chữ cái ở thời kỳ cổ đại, cho đến sự xuất hiện của máy chiếu mật(Enigma) trong Thế chiến thứ hai, mật mã học đã ngày càng tiến xa và phức tạp hơn. Với sự phát triển của máy tính và công nghệ tính toán, mật mã học liên tục tiến bộ để chống lại các cuộc tấn công ngày càng tinh vi và đáp ứng nhu cầu bảo mật thông tin trong thời đại số hóa hiện nay.

Sự ra đời thuật toán mật mã hóa công khai RSA được Ron Rivest, Adi Shamir và Len Adleman đề xuất vào năm 1977 tại Học viện Công nghệ Massachusetts (MIT) đánh dấu một sự tiến bộ vượt bậc của lĩnh vực mật mã học trong việc sử dụng khóa công cộng. RSA đang được sử dụng phổ biến trong thương mại điện tử và được cho là đảm bảo an toàn với điều kiện độ dài khóa đủ lớn. Ngoài ra hiện nay, các công nghệ mới như mật mã lưới và mật mã lượng tử đang trở thành xu hướng chính trong nghiên cứu mật mã hiện đại, do ưu điểm về tốc độ mã hóa và giải mã, cũng như việc lưu trữ khóa cần ít tài nguyên mà độ an toàn vẫn được đảm bảo. Tiêu biểu là hệ mật khóa công khai NTRU được đề xuất bởi J.Hoffstein, J.Pipher và J. Silverman năm 1998. Từ đó tới nay, hệ mật này được nghiên cứu phát triển cải tiến và đã thể hiện được những ưu điểm mà người sử dụng kì vọng về hệ mật sử dụng lưới.

Lý thuyết lưới là một lĩnh vực trong toán học có liên quan đến việc nghiên cứu các cấu trúc đại số và hình học của các mạng lưới được phát triển từ những năm 1940 và được sử dụng trong nhiều lĩnh vực như mật mã học, khoa học máy tính và kỹ thuật thông tin. Tuy nhiên, những ứng dụng của nó trong hệ mật mã mới được phát triển từ khi có thuật toán rút gọn cơ sở lưới do ba nhà toán học Arjen Lenstra, Hendrik Lenstra, László Lovász tìm ra (được viết tắt là LLL) năm 1982 [8]. Thuật toán LLL lập tức được xem là một trong những thuật toán quan trọng trong lý thuyết số khi đã chỉ ra một thuật toán trong thời gian đa thức cho việc phân tích các đa thức nguyên và giải các phương trình diophantine đồng thời. Năm 1996, Coppersmith đã xây dựng một thuật toán phá mã RSA trong một số các trường hợp dựa trên thuật toán LLL. Và chính vì vậy rất nhiều hệ mật mã với các điều kiện riêng trước đây đã bị phá mã. Điều đó cho thấy thuật toán LLL đã có một sức mạnh phá mã mạnh mẽ.

Đồ án này sẽ hướng tới phân tích, tìm hiểu những kiến thức tổng quan về lý thuyết lưới, các hệ

mật mã RSA, NTRU và những ứng dụng mạnh mẽ của thuật toán giảm cơ sở lưới LLL trong các cuộc tấn công vào hệ mật mã. Bố cục bài báo cáo gồm 3 chương chính:

- Chương 1 trình bày một số kiến thức chuẩn bị về lý thuyết lưới như định nghĩa lưới, định thức lưới, cơ sở lưới, vector ngắn nhất và một số nội dung về phép trực giao hóa Gram-Schmidt và cách xây dựng lưới con .
- Chương 2 trình bày phương pháp giảm lưới, tìm hiểu về nội dung của thuật toán rút gọn cơ sở lưới LLL và xây dựng chương trình chạy thực nghiệm trên phần mềm SageMath.
- Chương 3 trình bày một số nội dung về hệ mật mã RSA và NTRU và ứng dụng thuật toán LLL tấn công RSA. Bên cạnh đó, trình bày nội dung cuộc tấn công vào hệ mật mã RSA có dựa trên lý thuyết lưới. Xây dựng chương trình chạy thực nghiệm trên phần mềm SageMath.

Chương 1

Lý thuyết lưới

Trong chương này, bài báo cáo sẽ trình bày lại tổng quan một số kiên thức về lý thuyết lưới như định nghĩa lưới trong không gian thực, trực giao hóa Gram-Schmidt , xây dựng cơ sở lưới và trình bày một số nội dung về định thức của lưới và vector ngắn nhất . Để chuẩn bị cơ sở cho các chương về sau.Các nội dung của chương này về lý thuyết lưới hầu như được trích dẫn từ Luận văn Thạc sĩ của Khúc Xuân Thành [3] và cuốn sách Introduction to Cryptography with Coding Theory [4].

1.1 Định nghĩa

Tập hợp các vector cột $n \times 1$ gồm các phần tử thuộc R cùng các phép toán thông thường lập thành một không gian vector n chiều trên trường R, ký hiệu R^n. Cơ sở của không gian vector R^n là một hệ vector độc lập tuyến tính và sinh ra không gian vector đó.

Định nghĩa 1.1. *Tập hợp các vector* $\{x_1, x_2, ..., x_n\}$*trong* R^n *độc lập tuyến tính nếu:*

$$c_1x_1 + c_2x_2 + ... + c_nx_n = 0 \quad với \quad c_i \in R.$$

Dấu "="xảy ra khi và chỉ khi $c_1 = c_2 = ... = c_n$.

Định nghĩa 1.2. *Cho* $n \geq 1, \{x_1, x_2, ..., x_n\}$ *là một cơ sở* R^n. *Lưới n chiều với cơ sở* $x_1, x_2, ..., x_n$ *là tập hợp L tất cả các tổ hợp tuyến tính của các vector cơ sở đó với hệ số nguyên*

$$L = a_1x_1 + a_2x_2 + ... + a_nx_n \quad với\ a_i \in Z.$$

Các vector $x_1, x_2, ..., x_n$ *được gọi là cơ sở của lưới.*

1.2 Định thức

Định nghĩa 1.3. *Với $\{x_1, x_2, ..., x_n\}$ là cơ sở của lưới L ta định nghĩa*

$$detL = |det(x_1, x_2, ..., x_n)|.$$

Về hình học , nó là thể tích của các hình bình hành kéo dài bởi các vector cơ sở $x_1, x_2, ..., x_n$.

Bổ đề 1.4. *Cho $x_1, x_2, ..., x_n$ và $y_1, y_2, ..., y_n$ là 2 cơ sở của lưới $L \subset R^n$. Lấy X, Y lần lượt là các ma trận $n \times n$ nhận x_i (tương ứng y_i) là các hàng thứ i. khi đó $Y = CX$ với C là ma trận vuông $n \times n$ với các hệ số nguyên và $det(C) = \pm 1$.*

Chứng minh :

Với mọi y_i thuộc lưới với cơ sở $x_1, x_2, ..., x_n$ và mọi x_i thuộc lưới với cơ sở $y_1, y_2, ..., y_n$. Khi đó :

$$x_i = \sum_{j=1}^{n} b_{ij} y_j \ , \qquad y_i = \sum_{j=1}^{n} c_{ij} x_j.$$

Ta có thể viết gọn lại dưới dạng ma trận:

$$X = BY \ , \qquad Y = CX.$$

Vì vậy ta có $X = BCX$ và $Y = CBY$. Do $x_1, x_2, ..., x_n$ và $y_1, y_2, ..., y_n$ là cơ sở trong không gian R^n nên ma trận X, Y là ma trận khả nghịch.

Suy ra $BC = CB = I$ và $det(B).det(C) = 1$. Thêm nữa B, C là các ma trận hệ số nguyên . Do đó:

$$\begin{cases} det(B) = det(C) = 1 \\ det(B) = det(C) = -1 \end{cases}$$

Nhận xét. *Định thức của một lưới không phụ thuộc vào cách chọn cơ sở.*

Chứng minh:

Giả sử lưới $L \subset R^n$ có 2 cơ sở $x_1, x_2, ..., x_n$ và $y_1, y_2, ..., y_n$. Theo Bổ đề 1.4 ta có :

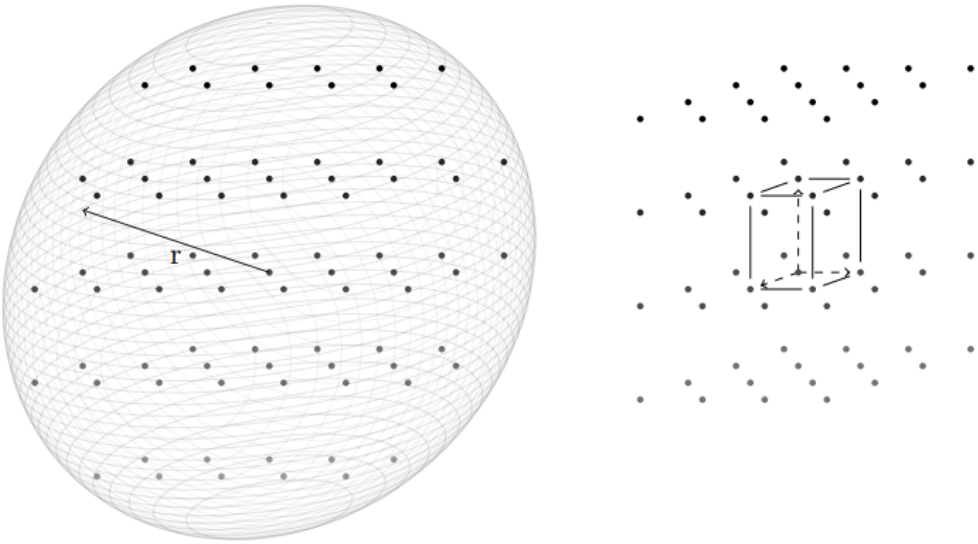
$$|det(Y)| = |det(CX)| = |det(C).det(X)| = |\pm 1.det(X)| = |det(X)|$$

Ví dụ: Đặt $\{x_1, x_2\}$ là cơ sở của lưới L. Với $x_1 = (1, 0), \quad x_2 = (0, 1)$. Khi đó

$$L = a_1 x_1 + a_2 x_2 \quad \text{với} \quad a_1, a_2 \in Z$$

Đặt

$$\begin{cases} y_1 = x_1 + 5x_2 = (1, 5) \\ y_2 = x_2 = (0, 1) \end{cases} \Rightarrow \{y_1, y_2\} \text{ là một cơ sở khác của lưới.}$$



Hình 1.1: Lưới $L \subset R^3$. Hình bên trái là biểu diễn $detL$ dưới quả bóng và các phần tử lưới L bên trong. Hình bên phải biểu diễn $detL$ dưới dạng thể tích hình bình hành kéo dài bởi các vector cơ sở.

1.3 Vector ngắn nhất

Độ dài vector $v = (v_1, v_2, ..., v_n)$ là:

$$\|v\| = (x_1^2 + x_2^2 + ... + x_n^2)^{\frac{1}{2}}.$$

Có nhiều vấn đề liên quan đến việc tìm một vector khác không ngắn nhất trong lưới. Bài toán tìm vector ngắn nhất rất khó để giải, khi số chiều của lưới lớn.

Ví dụ: Một vector ngắn nhất trong lưới tạo bởi cơ sở $(31, 59)$ và $(37, 70)$ là $(3, -1)$ (một vector ngắn nhất khác $(-3, 1)$).
Ta xác định $(3, -1)$ nằm trong lưới bằng cách viết lại:

$$(3, -1) = -19(31, 59) + 16(37, 70)$$

Thật vậy ta có $\{(3, -1), (1, 4)\}$ là một cơ sở của lưới. Ta thấy chúng gần như trực giao nhau. Còn 2 cơ sở ban đầu gần như song song và có tích vô hướng rất lớn. Ở phần tiếp theo, sẽ chỉ ra cách thay thế một cơ sở lưới bằng một cơ sở mới gần như trực giao nhau.

1.4 Trực giao hóa Gram-Schmidt

Phần này trình bày thuật toán Gram-Schmidt cổ điển cho việc chuyển một cơ sở bất kỳ trong R^n thành một cơ sở trực giao. Đây là kỹ thuật quan trọng trong thuật toán LLL.

Định nghĩa 2.1. *Cho $x_1, x_2, ..., x_n$ là một cơ sở trong R^n. Trực giao hóa Gram-Schmidt của $x_1, x_2, ..., x_n$ là một cơ sở $x_1^*, x_2^*, ..., x_n^*$ được định nghĩa như sau:*

i) $x_1 = x_1^*$,

ii) $x_i^* = x_i - \sum_{j=1}^{i-1} \mu_{ij} x_j^* \quad (2 \le i \le n), \quad \mu_{ij} = \frac{x_i.x_j^*}{x_j^*.x_j^*}$.

Chú ý: Các vector không được chuẩn hóa độ dài. Ngoài ra, nếu $x_1, x_2, ..., x_n$ là một cơ sở của lưới L thì sau khi trực giao hóa ta thu được các vector $x_1^*, x_2^*, ..., x_n^*$ có thể không nằm trong lưới L. Nếu ta đặt $\mu_{ii} = 1$, $\mu_{ij} = 0$ với $1 \le i < j \le n$. Ta có thể viết lại dưới dạng ma trận :

$$\underbrace{\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}}_{X} = \underbrace{\begin{pmatrix} 1 & 0 & \cdots & 0 \\ \mu_{21} & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \mu_{n1} & \cdots & \cdots & 1 \end{pmatrix}}_{M} \times \underbrace{\begin{pmatrix} x_1^* \\ x_2^* \\ \vdots \\ x_n^* \end{pmatrix}}_{X^*}$$

Định lý 2.2. *Cho $x_1, x_2, ..., x_n$ là một cơ sở của R^n và $x_1^*, x_2^*, ..., x_n^*$ là trực giao hóa Gram-Schmidt của nó. Khi đó*

i) $x_i^*.x_j^* = 0$ *với* $1 \le i < j \le n$,

ii) $\|x_k^*\| \le \|x_k\|$,

iii) $det(X^*) = det(X)$.

Chứng minh.

i) Điều này luôn đúng trong quá trình trực giao hóa Gram-Schmidt.

ii) Ta có $x_k = x_k^* + \sum_{j=1}^{k-1} \mu_{kj}.x_j^*$. Theo định lý Py-ta-go ta có :

$$\|x_k\|^2 = \|x_k^*\|^2 + \sum_{j=1}^{k-1} (\mu_{kj})^2.\|x_j^*\|^2.$$

Suy ra : $$\|x_k^*\| \le \|x_k\|.$$

iii) Ta có $X = MX^*$ (như chú ý trên)

Mà M là ma trận tam giác dưới với $\mu_{ii} = 1$.

Do đó: $det(M) = 1 \Rightarrow det(X^*) = det(X)$.

Bổ đề 2.3. *Cho $x_1, x_2, ..., x_n$ là một cơ sở của R^n và $x_1^*, x_2^*, ..., x_n^*$ là trực giao hóa Gram-Schmidt của nó. Ta có*

$$|det(X)| \leq \|x_1\|\|x_2\|...\|x_n\|.$$

Chứng minh.

Ta có $|det(X^*)|$ là thể tích hình hộp n chiều căng bởi các vector trực giao $x_1^*, x_2^*, ..., x_n^*$ nên suy ra:

$$|det(X^*)| = \|x_1^*\|\|x_2^*\|...\|x_n^*\|.$$

Theo Định lý 2.2, ta có $\|x_k^*\| \leq \|x_k\|$ nên

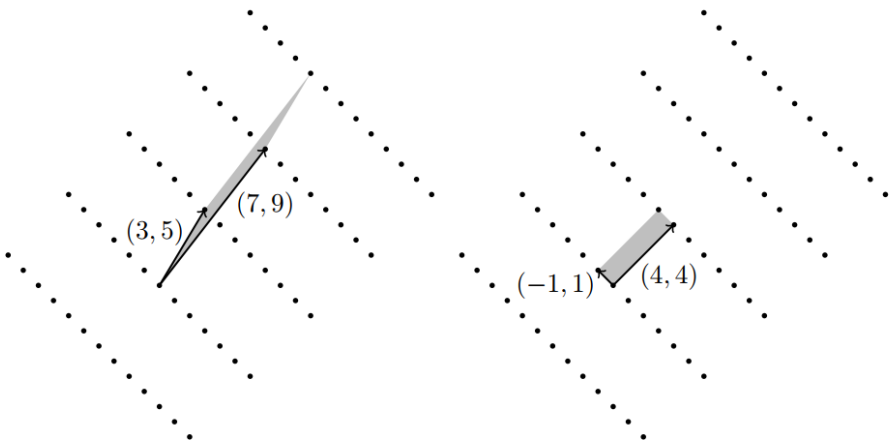
$$|det(X)| = |det(X^*)| \leq \|x_1\|\|x_2\|...\|x_n\|.$$

Chương 2

Thuật toán LLL

Trong mục này , bài báo cáo sẽ trình bày về thuật toán LLL cho việc rút gọn cơ sở lưới. Thuật toán này được lấy tên theo chữ cái đầu tiên của ba tác giả A.Ken.Lenstra, H.W.Lenstra Jr và L. Lovasz, những người đã giới thiệu thuật toán này đầu tiên năm 1982[8].

Ý tưởng việc rút gọn cơ sở là thay đổi cơ sở B của lưới L thành cơ sở mới B' ngắn hơn sao cho $|detL|$ không đổi. Rút gọn cơ sở có thể được sử dụng để giải bài toán vector ngắn nhất theo nghĩa là vector cơ sở ngắn nhất (b_1 trong thuật toán giảm lưới gần như là vector ngắn nhất).



Hình 2.1: Lưới $L \subset R^2$. Mô tả kết quả giảm lưới trong R^2.

2.1 Lưới 2 chiều

Việc giảm cơ sở lưới 2 chiều tương đối dễ hiểu và nó đóng vai trò then chốt trong thuật toán giảm cơ sở LLL.

Đặt $\{x_1, x_2\}$ làm cơ sở của lưới 2 chiều và mục tiêu của chúng ta sẽ thay thế cơ sở này bằng cơ sở giảm. Nếu $\|x_1\| > \|x_2\|$ ta sẽ hoán vị x_1 với x_2. Sau đó , ta thay vector x_2 bằng vector x_2^* vuông góc với x_1 qua phép trực giao hóa Gram-Schmidt :

$$x_2^* = x_2 - \frac{x_1.x_2}{x_1.x_1}.x_1.$$

Đặt t là số nguyên gần nhất với $\frac{x_1.x_2}{x_1.x_1}$. Sau đó, ta thay thế cơ sở $\{x_1, x_2\}$ bằng cơ sở

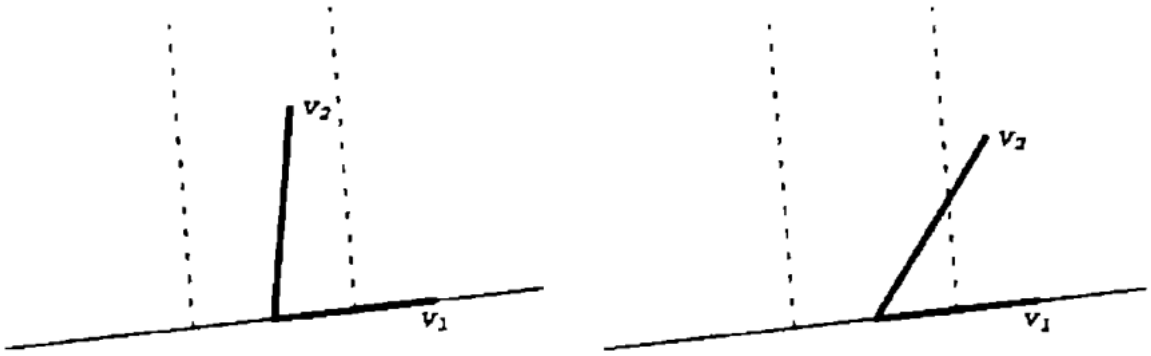
$$\{x_1, x_2 - tx_1\}.$$

Sau đó, ta lặp lại quy trình này đến khi không thể giảm cơ sở nữa.

Định nghĩa 2.4. *Một cơ sở $\{x_1, x_2\}$ được gọi là cơ sở giảm khi và chỉ khi*

i) $\|x_1\| \leq \|x_2\|$,

ii) $|\mu| = \left|\frac{x_1.x_2}{x_1.x_1}\right| \leq \frac{1}{2}$.



Hình 2.2: Hình ảnh cơ sở giảm và chưa giảm

Ví dụ Cho $\{x_1, x_2\}$ là cơ sở của lưới L với $x_1 = (31, 59), x_2 = (37, 70)$. Thực hiện giảm cơ sở lưới trên.

Ta thấy $\|x_1\| \leq \|x_2\|$. Ta có $\frac{x_1.x_2}{x_1.x_1} = \frac{5277}{4442} \approx 1,18$

+ Chọn $t_1 = 1$ suy ra cơ sở mới:

$$\begin{cases} x_1' = x_1 = (31, 59) \\ x_2' = x_2 - t_1 x_1 = (6, 11) \end{cases}$$

+ Ta thấy $\|x_1\| \geq \|x_2\|$, hoán vị 2 vector ta có:

$$\begin{cases} x_1' = (6, 11) \\ x_2' = (31, 59) \end{cases}.$$

Ta có: $\dfrac{x_1^{'}.x_2^{'}}{x_1^{'}.x_1^{'}} \approx 5,3$

+ Chọn $t_2 = 5$ ta có cơ sở mới:

$$\begin{cases} x_1^{''} = x_1^{'} = (6,11) \\ x_2^{''} = x_2^{'} - t_2 x_1^{'} = (1,4) \end{cases}$$

+ Hoán vị vị trí $x_1^{''}$ và $x_2^{''}$ ta thu được cơ sở

$$\begin{cases} x_1^{''} = (1,4) \\ x_2^{''} = (6,11) \end{cases}$$

Ta có $\dfrac{x_1^{''}.x_2^{''}}{x_1^{''}.x_1^{''}} = \dfrac{50}{17} \approx 2,9.$ Chọn $t_3 = 3$ ta có:

$$\begin{cases} x_1^{(3)} = (1,4) \\ x_2^{(3)} = x_2^{''} - 3.x_1^{''} = (3,-1) \end{cases}$$

+ Hoán vị 2 vector ta có

$$\begin{cases} x_1^{(3)} = (3,-1) \\ x_2^{(3)} = (1,4) \end{cases}$$

Ta thấy $\dfrac{x_1^{(3)}.x_2^{(3)}}{x_1^{(3)}.x_1^{(3)}} = -\dfrac{1}{10}$. Suy ra $\{x_1^{(3)}, x_2^{(3)}\}$ là cơ sở giảm.

Định lý 2.5. *Nếu x_1, x_2 là cơ sở rút gọn của lưới L thì x_1 là vector khác không ngắn nhất của lưới đó.*

Chứng minh.

Đặt $ax_1 + bx_2$ là vector khác không bất kì trong lưới L (với $a, b \in Z$).

Ta có: $$\|a_1x_1 + a_2x_2\|^2 = a^2\|x_1\|^2 + b^2\|x_2\|^2 + 2abx_1x_2$$

Vì x_1, x_2 là cơ sở giảm nên:

$$-\frac{1}{2}x_1x_1 < x_1x_2 < \frac{1}{2}x_1x_1.$$

Suy ra: $2abx_1x_2 \geq -|ab|\|x_1\|^2$. Do đó

$$\begin{aligned} \|a_1x_1 + a_2x_2\|^2 &\geq a^2\|x_1\|^2 - |ab|\|x_1\|^2 + b^2\|x_2\|^2 \\ &\geq a^2\|x_1\|^2 - |ab|\|x_1\|^2 + b^2\|x_1\|^2 \quad (\text{do } \|x_2\| \geq \|x_1\|) \\ &= (a^2 - |ab| + b^2)\|x_1\|^2. \end{aligned}$$

Ta thấy $a^2 - |ab| + b^2 \in N$ và bằng 0 khi và chỉ khi $a = b = 0$.

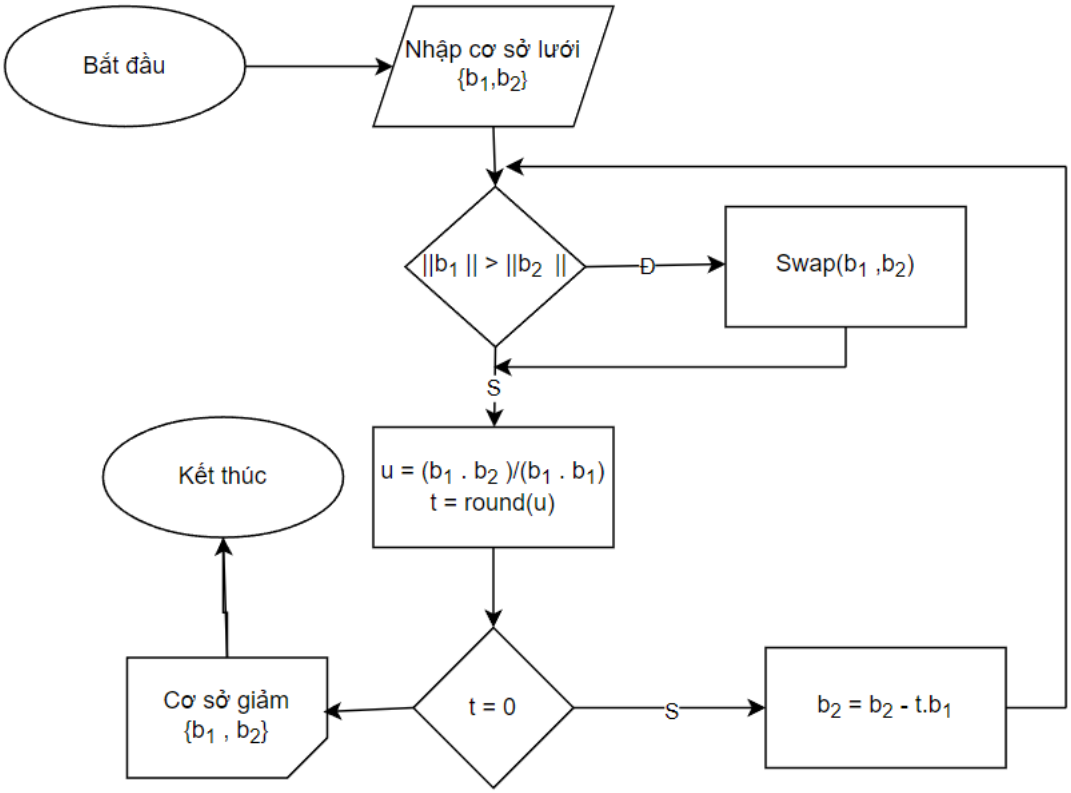
Nhưng $ax_1 + bx_2 \neq 0$ $\Rightarrow$ Trường hợp $a = b = 0$ (loại).

$$\Rightarrow a^2 - |ab| + b^2 \geq 1$$
$$\Rightarrow \|a_1x_1 + a_2x_2\| \geq \|x_1\|^2.$$

Do đó x_1 là vector khác không ngắn nhất của lưới.

Thuật toán: *Giảm lưới 2 chiều.*

- Bước 1 : Nhập vào cơ sở lưới 2 chiều $b_1, b_2 \in R^2$.
- Bước 2 : Nếu $\|b_1\| > \|b_2\|$, ta hoán đổi vị trí b_1 và b_2.
- Bước 3 : Đặt t bằng phần nguyên $\left|\frac{b_1.b_2}{b_1.b_1}\right|$. Nếu $t = 0$, dừng lại và kết luận là cơ sở giảm. Ngược lại, ta tính lại $b_2 = b_2 - t.b_1$ và quay lại bước 2



Hình 2.3: Thuật toán giảm lưới 2 chiều.

2.2 Thuật toán LLL

Việc giảm lưới ở kích thước lớn hơn 2 chiều khó khăn hơn nhiều. Trong nhiều bài toán, cần có một vector ngắn và không nhất thiết vector đó ngắn nhất. Thuật toán LLL sử dụng cách tiếp cận này và tìm kiếm các vector gần như ngắn nhất(trong thời gian đa thức). Phương pháp này được dùng cho nhiều ứng dụng khác nhau.

Định nghĩa 2.6. *Tham số rút gọn α là một số thực sao cho $\frac{1}{4} < \alpha < 1$. Giá trị chính tắc của α là $\alpha = \frac{3}{4}$[6].*

Định nghĩa 2.7. *Cơ sở $x_1, x_2, ..., x_n$ được gọi là một cơ sở α-rút gọn nếu nó thỏa mãn:*

i) $|\mu_{ij}| \leq \frac{1}{2}$ *với* $1 \leq j < i \leq n$,

ii) $\|x_i^*\|^2 \geq (\alpha - \mu_{i,i-1}^2)\|x_{i-1}^*\|^2$ *với* $2 \leq i \leq n$.

Ta định nghĩa tham số bổ trợ β như sau: $\beta = \frac{4}{4\alpha - 1}$.

Do $\frac{1}{4} < \alpha < 1 \quad \Rightarrow \quad \beta > \frac{4}{3}$. Nếu $\alpha = \frac{3}{4}$ thì $\beta = 2$.

Mệnh đề 2.8. *Nếu $x_1, x_2, ..., x_n$ là một cơ sở α-rút gọn của lưới L trong R^n và $x_1^*, x_2^*, ..., x_n^*$ là cơ sở trực giao hóa Gram-Schmidt của nó thì*

i) $\|x_j\|^2 \leq \beta^{i-1}\|x_i^*\|^2$ *với* $1 \leq j \leq i \leq n$,

ii) $\|x_1\|.\|x_2\|...\|x_n\| \leq \beta^{\frac{n(n-1)}{4}} det(L)$,

iii) $\|x_1\| \leq \beta^{\frac{(n-1)}{4}} det(L)^{\frac{1}{n}}$.

Chứng minh.

i) Theo Định nghĩa 2.7 ta có :

$$\|x_i^*\| \geq (\alpha - \mu_{i,i-1}^2)\|x_{i-1}^*\|^2 \geq (\alpha - \frac{1}{4}).\|x_{i-1}^*\|^2 = \frac{1}{\beta}\|x_{i-1}^*\|^2.$$

Khi đó $\|x_{i-1}^*\|^2 \leq \beta.\|x_i^*\|^2$,và

$$\|x_j^*\|^2 \leq \beta^{i-j}\|x_i^*\|^2 \qquad (1 \leq j \leq i \leq n). \qquad (1)$$

Mặt khác, ta có :

$$x_i = x_i^* + \sum_{j=1}^{i-1} \mu_{ij} x_j^*,$$

và vì $x_1^*, x_2^*, ..., x_n^*$ là trực giao nên:

$$\|x_i\|^2 = \|x_i^*\|^2 + \sum_{j=1}^{i-1}(\mu_{ij})^2\|x_j^*\|^2 \leq \|x_i^*\|^2 + \sum_{j=1}^{i-1}\frac{1}{4}\beta^{i-j}\|x_i^*\|^2$$
$$\leq (1 + \frac{1}{4}\sum_{j=1}^{i-1}\beta^{i-j})\|x_i^*\|^2 = (1 + \frac{1}{4}\frac{\beta^i - \beta}{\beta - 1})\|x_i^*\|^2.$$

Bằng quy nạp, ta có thể dễ chứng minh

$$1 + \frac{1}{4}\frac{\beta^i - \beta}{\beta - 1} \leq \beta^{i-1}.$$

Vì vậy,

$$\|x_i\|^2 \leq \beta^{i-1}\|x_i^*\|^2. \qquad (2)$$

Từ (1), (2) ta có:

$$\|x_j\|^2 \leq \beta^{j-1}\|x_j\|^2 \leq \beta^{i-1}\|x_i\|^2.$$

ii) Theo Bổ đề 2.3. ta có :

$$det(L) = \|x_1^*\|.\|x_2^*\|...\|x_n^*\| \leq \|x_1\|.\|x_2\|...\|x_n\|.$$

Từ (2) ta có:

$$\|x_1\|.\|x_2\|...\|x_n\| \leq \beta^{\frac{n(n-1)}{4}}\|x_1^*\|.\|x_2^*\|...\|x_n^*\| = \beta^{\frac{n(n-1)}{4}}.det(L)$$

iii) Xét $j = 1$ trong i), ta có

$$\|x_1\|^2 \leq \beta^{i-1}\|x_i^*\|^2$$

với $1 \leq i \leq n$.

Khi đó:

$$\|x_1\|^{2n} \leq \beta^{0+1+2+...+(n-1)}\|x_1^*\|.\|x_2^*\|...\|x_n^*\| \leq \beta^{\frac{n(n-1)}{2}}.det(L)^2.$$

Suy ra:

$$\|x_1\| \leq \beta^{\frac{n-1}{4}}.det(L)^{\frac{1}{n}}.$$

Bổ đề 2.9. *Cho $x_1, x_2, ..., x_n$ là một cơ sở của lưới L và $x_1^*, x_2^*, ..., x_n^*$ tương ứng là phép trực giao hóa Gram-Schmidt của nó. Với $y \in L$ khác không,*

$$\|y\| \geq min\{\|x_1^*\|, \|x_2^*\|, ..., \|x_n^*\|\}.$$

Chứng minh.

Cho y là vector khác không bất kỳ thuộc lưới L ta có :

$$y = a_1x_1 + a_2x_2 + ... + a_nx_n.$$

Lấy k là chỉ số lớn nhất sao cho $a_k \neq 0$. Khi đó:

$$y = \sum_{i=1}^{k} a_i . \sum_{j=1}^{i} \mu_{ij} x_j^* = \sum_{i=1}^{k} \sum_{j=1}^{i} a_i . \mu_{ij} . x_j^*$$
$$= \sum_{i=1}^{k} \left(\sum_{j=1}^{i} a_i . \mu_{ij} \right) x_j^* = a_k . x_k^* + \sum_{j=1}^{k} v_j x_j^* .$$

Khi đó:

$$\|y\| \geq a_k^2 . \|x_k^*\|^2 + \sum_{j=1}^{k} v_j^2 \|x_j^*\|^2 .$$

Mặt khác, do a_k nguyên và khác 0 nên $a_k^2 \geq 1$:

$$\Rightarrow \quad \|y\| \geq \|x_k^*\|^2 + \sum_{j=1}^{k} v_j^2 \|x_j^*\|^2 .$$

Vì vậy:

$$\|y\| \geq \|x_k^*\| \geq min\{\|x_1^*\|, \|x_2^*\|, ..., \|x_n^*\|\}.$$

Ta có điều phải chứng minh.

Định lý 2.10. *Nếu $x_1, x_2, ..., x_n$ là một cơ sở α-rút gọn của lưới L trong R^n và $y \in L$ là vector trong lưới khác không bất kỳ thì:*

$$\|x_1\| \leq \beta^{\frac{n-1}{2}} \|y\|.$$

Đặc biệt, vector đầu tiên trong cơ sở rút gọn không dài hơn $\beta^{\frac{n-1}{2}}$ lần vector ngắn nhất khác không của lưới L.

Chứng minh.

Ta có $\|x_i^*\| \geq \frac{1}{\beta} \|x_{i-1}^*\|$ với $1 \leq i \leq n$. Vì $x_1^* = x_1$ nên

$$\|x_1\|^2 = \|x_1^*\| \leq \beta^{n-1} \|x_n^*\|.$$

Do đó, với $1 \leq i \leq n$ ta có :

$$\|x_i^*\|^2 \geq \beta^{-(i-1)} \|x_1\|^2.$$

Theo Bổ đề 2.9, với vector khác 0 bất kì $y \in L$ ta có:

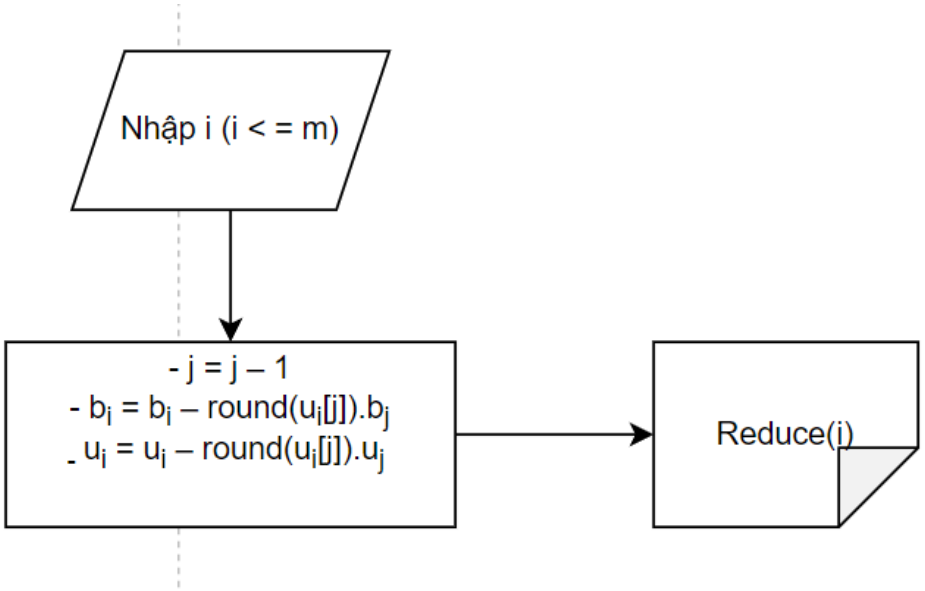
$$\|y\| \geq \min\{\|x_1^*\|, \|x_2^*\|, ..., \|x_n^*\|\} \geq \beta^{\frac{-(n-1)}{2}} \|x_1^*\|.$$

Vì vậy, ta có điều phải chứng minh

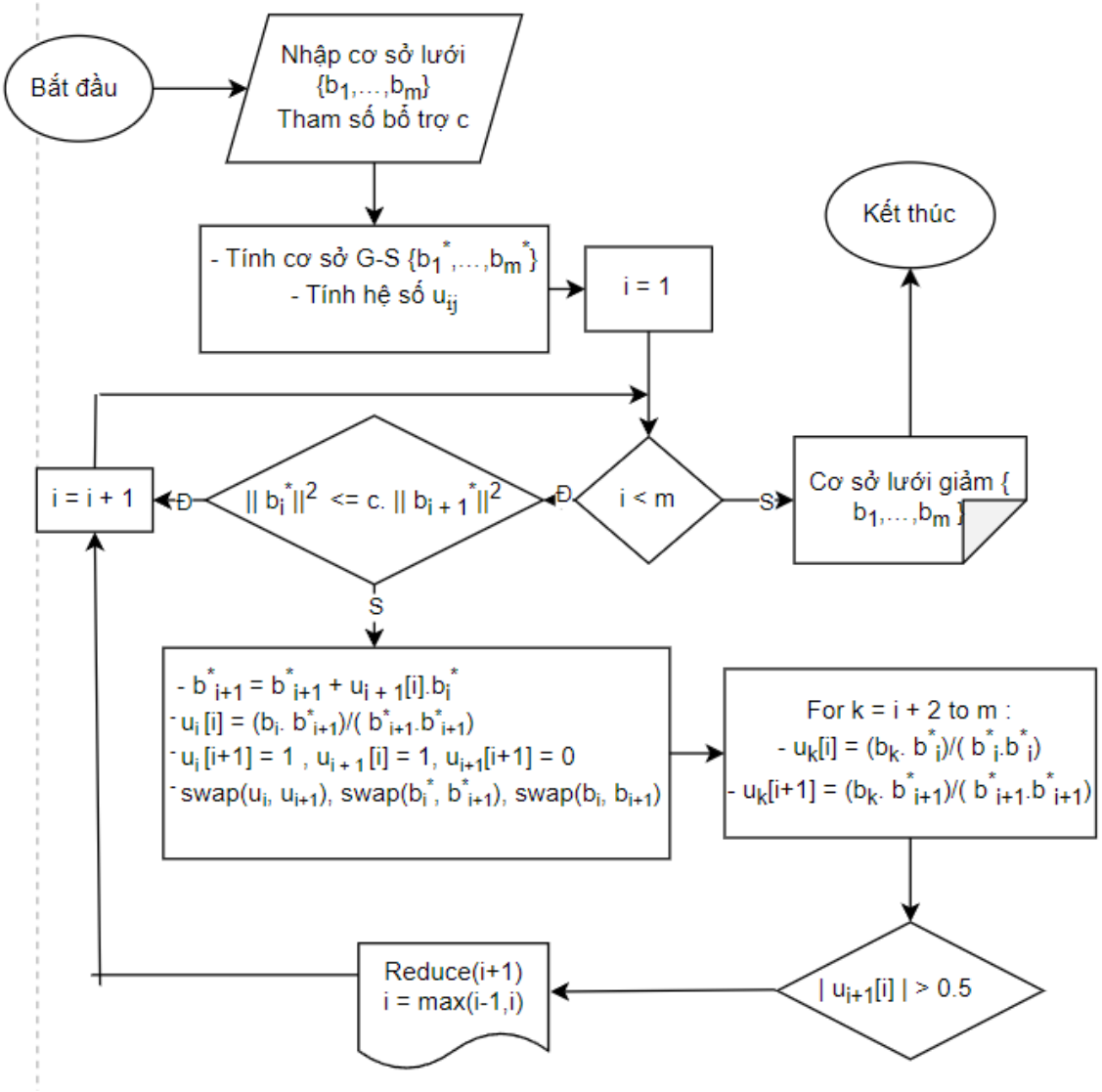
Thuật toán LLL.

- Bước 1 : Nhập vào lưới L với cơ sở lưới $b_1, b_2, ..., b_n$ và tham số bổ trợ c với $c \geq \frac{4}{3}$.
- Bước 2 : Tính cơ sở trực giao hóa Gram-Schmidt $b_1^*, b_2^*, ..., b_n^*$ và các hệ số trực giao u_{ij}.
- Bước 3 : Thực hiện vòng lặp, nếu $\|b_i^*\|^2 \leq c.\|b_{i+1}^*\|^2$ tăng i lên, ngược lại thực hiện hoán vị các vector b_i và b_{i+1}, sau đó cập nhật lại cơ sở trực giao Gram-Schmidt và các hệ số của nó.
  Kiểm tra xem hệ số $\|u_{ij}\| > \frac{1}{2}$ thực hiện thủ tục rút gọn.
  Kết quả cuối cùng thu được cơ sở lưới giảm L.
- Thủ tục rút gọn: Ta sẽ rút gọn vector b_i bằng cách trừ đi một bội nguyên $\|u_{ij}\|$ của b_j. Vì $\|u_{ij}\|$ là số nguyên gần nhất với hệ số Gram-Schmidt u_{ij} nên đây là phép rút gọn tốt nhất ta có thể thực hiện, với điều kiện ràng buộc là b_i vẫn nằm trong lưới ban đầu. Và cập nhật các hệ số cơ sở Gram-Schmidt.

Sau đây là lưu đồ mô tả cách hoạt động của thuật toán LLL.



Hình 2.4: Hàm reduce.



Hình 2.5: Thuật toán LLL.

2.3 Thực hiện chương trình

Để triển khai và kiểm tra các thuật toán, bài báo cáo này sẽ sử dụng phần mềm SageMath . SageMath là một chương trình tính toán mã nguồn mở, cấp phép bởi GPL. Nó là sự kết hợp sức mạnh của nhiều gói source vào Python (NumPy, SciPy, matplotlib, Sympy,Maxima, GAP, FLINT, R and many more). Nhiệm vụ của Sage là tạo ra một chương trình miễn phí có thể thay thế Magma, Maple, Mathematica và Matlab. Trang web https://www.sagemath.org/.

Chương trình thuật toán LLL:

```
    #Ham giam luoi
def reduce(i, B, U):

    print("gia tri ma tran B: \n")
    j = i - 1
    while j >= 0:
        B.set_column(i, B.column(i) - round(U[j, i]) * B.column(j))
        U.set_column(i, U.column(i) - round(U[j, i]) * U.column(j))
        j = j - 1
    print(B)

#ham chinh gia tri ma tran bang thuat toan LLL
def LLL(B, c=2):
    #Lay so hang va cot ma tran
    n = B.nrows()
    m = B.ncols()
    #khoi tao ma tran
    U = matrix(RR, m, m)
    O = matrix(RR, n, m)

    # Truc giao hoa G-S
    for i in range(0, m):
        U[i, i] = 1
        O.set_column(i, B.column(i))
        for j in range(0, i):
            U[j, i] = (B.column(i) * O.column(j)) / (O.column(j) * O.column(j))
            O.set_column(i, O.column(i) - U[j, i] * O.column(j))
            reduce(i, B, U) # Goi ham recduce de bien doi ma tran
        i += 1

    # Dieu kien giam
    i = 0
    while i < m - 1:
        if O.column(i) * O.column(i) <= c * O.column(i + 1) * O.column(i + 1):
            i += 1
        else:
            O.set_column(i + 1, O.column(i + 1) + U[i, i + 1] * O.column(i))
            U[i, i] = (B.column(i) * O.column(i + 1)) / (O.column(i + 1) * O.column(i + 1))
```

```
            U[i + 1, i] = 1
            U[i, i + 1] = 1
            U[i + 1, i + 1] = 0
            O.set_column(i, O.column(i) - U[i, i] * O.column(i + 1))
            U.swap_columns(i, i + 1)
            O.swap_columns(i, i + 1)
            B.swap_columns(i, i + 1)
            for k in range(i + 2, m):
                U[i, k] = (B.column(k) * O.column(i)) / (O.column(i) * O.column(i))
                U[i + 1, k] = (B.column(k) * O.column(i + 1)) / (O.column(i + 1) * O.column(i
                    + 1))
            if abs(U[i, i + 1]) > 0.5:
                reduce(i + 1, B, U)
                i = max(i - 1, 0)
    return B

# Nhap du lieu tu ban phim

n = int(input("Nhap so chieu cua luoi n: "))
A = Matrix(ZZ, n, n)
for j in range(n):
    row = input(f"Nhap hang {j+1} cua ma tran : ")
    A[j] = list(map(int, row.split()))
print(A)
print("Ket qua giam luoi:\n")
res = LLL(A)
```

Một số kết quả thu được.

1. **Lưới 2 chiều:** Cho cơ sở $x_1 = (2, 5), x_2 = (5, 8)$:

```
Nhap so chieu cua luoi n: 2
Nhap hang 1 cua ma tran : 2 5
Nhap hang 2 cua ma tran : 5 8
[2 5]
[5 8]
Ket qua giam luoi:

gia tri ma tran B:

[ 2  1]
[ 5 -2]
gia tri ma tran B:

[ 1  4]
[-2  1]
```

Hình 2.6: Chương trình thuật toán LLL với lưới 2 chiều.

2. **Lưới 3 chiều:** Cho cơ sở $x_1 = (1, 1, 1), x_2 = (-1, 0, 2), x_3 = (3, 5, 6)$:

```
Nhap so chieu cua luoi n: 3
Nhap hang 1 cua ma tran : 1 -1 3
Nhap hang 2 cua ma tran : 1 0 5
Nhap hang 3 cua ma tran : 1 2 6
[ 1 -1  3]
[ 1  0  5]
[ 1  2  6]
Ket qua giam luoi:

gia tri ma tran B:

[ 0  1 -1]
[ 1  0  0]
[ 0  1  2]
```

Hình 2.7: Chương trình thuật toán LLL với lưới 3 chiều.

Chương 3

Ứng dụng lý thuyết lưới trong hệ mật mã

3.1 Mật mã RSA

Trong phần này, bài báo cáo sẽ trình bày phương pháp biến đổi từ giải phương trình đồng dư phức tạp thành việc giải phương trình đa thức một biến. Và sử dụng thuật toán LLL, để tấn công hệ thống mật mã RSA trong các điều kiện nhất định. Bên cạnh đó, giới thiệu qua phương pháp Coppersmith trong việc tấn công hệ mật mã RSA.

3.1.1 Hệ mật mã RSA

RSA là một thuật toán mã hóa công khai bất đối xứng nổi tiếng được dùng để trao đổi thông tin bí mật qua Internet. Thuật toán được Ron Rivest, Adi Shamir và Len Adleman mô tả lần đầu tiên vào năm 1977 tại Học viện Công nghệ Massachusetts (MIT)[5]. Đây là thuật toán đầu tiên phù hợp với việc tạo ra chữ ký điện tử đồng thời với việc mã hóa. Nó đánh dấu một sự tiến bộ vượt bậc của lĩnh vực mật mã học trong việc sử dụng khóa công cộng. RSA đang được sử dụng phổ biến trong thương mại điện tử và được cho là đảm bảo an toàn với điều kiện độ dài khóa đủ lớn.
Đây là cách nó hoạt động:

* Tạo khóa

  + Chọn 2 số nguyên tố p và q

  + Biểu thị n là tích của p và q

  + Tính hàm số Euler $\phi(n) = (p-1)(q-1)$

  + Chọn số tự nhiên e sao cho $1 < e < \phi(n)$ và là số nguyên tố cùng nhau với $\phi(n)$

  + Tính d là modulo nghịch đảo $\phi(n)$ của e : $ed \equiv 1 \ (mod \ \phi(n))$.

* Mã hóa : $c = m^e \ mod \ n$. Trong đó m là tin nhắn và c là tin nhắn được mã hóa.

* Giải mã: Tin nhắn $m = c^d \ (mod \ n)$. Trong đó m là tin nhắn và c là tin nhắn được mã hóa.

Định nghĩa 3.1. *Bài toán RSA với số mũ e nhỏ, m có giá trị lớn. cho* m^e *, ta có B với* $|m - B| \leq n^{\frac{1}{e}}$. *Tìm* $m \in Z_n$.

3.1.2 Tấn công hệ mật mã RSA dựa trên lý thuyết lưới

Alice muốn gửi Bob một tin nhắn có dạng

Câu trả lời là $**$

Trong bài toán này, thông báo có dạng $m = B + x$, trong đó B cố định và $|x| \leq Y$ với Y nguyên. Sau đây là một cuộc tấn công với số mã hóa nhỏ.

Giả sử Bob có khóa công khai $(n, e) = (n, 3)$. Khi đó bài toán là $c \equiv (B + x)^3 \pmod{n}$.

Ta giả sử rằng Eve biết B, Y, n . Vì vậy cô ấy chỉ cần tìm x. Ta có thể tạo thành đa thức:

$$\begin{aligned} f(T) &= (B + T)^3 - c = T^3 + 3BT^2 + 3B^2T + B^3 - c \\ &\equiv T^3 + a_2T^2 + a_1T + a_0 (mod \ n). \end{aligned}$$

Eve đang tìm $|x| \leq Y$ sao cho $f(x) \equiv 0 \pmod{n}$. Nói cách khác, cô ấy đang tìm nghiệm nhỏ của đồng dư đa thức $f(T) \equiv 0 \pmod{n}$.

Eve áp dụng thuật toán LLL cho lưới được tạo bởi các vector:

$$v_1 = (1, 0, 0, 0), v_2 = (0, Yn, 0, 0), v_3 = (0, 0, Y^2n, 0), v_4 = (a_0, a_1Y, a_2Y^2, y^3).$$

Điều này tạo nên cơ sở mới b_1, b_2, b_3, b_4 . Nhưng ở đây, ta chỉ quan tâm đến b_1. Theo Mệnh đề 2.8 ta có

$$\begin{aligned} \|b_1\| &\leq 2^{\frac{3}{4}}.det(v_1, ..., v_4)^{\frac{1}{4}} \qquad \text{với} \quad \alpha = \frac{3}{4} \quad \text{hay} \quad \beta = 2 \\ &= 2^{\frac{3}{4}}.(n^3Y^6)^{\frac{1}{4}} = 2^{\frac{3}{4}} n^{\frac{3}{4}} Y^{\frac{3}{2}}. \end{aligned}$$

Ta có thể viết lại:

$$b_1 = c_1v_1 + ... + c_4v_4 = (e_0, Ye_1, Y^2e_2, Y^3e_3).$$

với c_i nguyên và với

$$\begin{aligned} e_0 &= C_1n + c_4a_0 \\ e_1 &= C_2n + c_4a_1 \\ e_2 &= C_3n + c_4a_2 \\ e_3 &= c_4. \end{aligned}$$

Ta dễ dàng thấy rằng $e_i \equiv c_4a_i \pmod{n}, \quad 0 \leq i \leq 2.$

Ta có thể thiết lập được đa thức

$$g(T) = e_3T^3 + e_2T^2 + e_1T + e_0.$$

Khi đó , vì số nguyên x thỏa mãn $f(x) \equiv 0 \pmod{n}$ và vì các hệ số của $c_4 f(T)$ và $g(T)$ đồng dạng với $mod(n)$ nên,

$$0 \equiv c_4 f(x) \equiv g(x) \pmod{n}.$$

Bây giờ ta giả sử rằng

$$Y < 2^{-7/6} n^{1/6}$$

Khi đó

$$\begin{aligned} |g(x)| &\leq |e_0| + |e_1 x| + |e_2 x^2| + |e_3 x^3| \\ &\leq |e_0| + |e_1| Y + |e_2| Y^2 + |e_3| Y^3 \\ &= (1,1,1,1).(|e_0|, |e_1 Y|, |e_2 Y^2|, |e_3 Y^3|) \\ &\leq \|(1,1,1,1)\| \|(|e_0|, |e_1 Y|, |e_2 Y^2|, |e_3 Y^3|)\| \\ &= 2\|b_1\|, \end{aligned}$$

Trong đó, bất đẳng thức cuối sử dụng bất đẳng thức Cauchy-Schwarz cho các tích vô hướng ($u.v \leq \|u\|\|v\|$) . Suy ra:

$$\|b_1\| \leq 2^{\frac{3}{4}} n^{\frac{3}{4}} Y^{\frac{3}{2}} < 2^{\frac{3}{4}} n^{\frac{3}{4}} (2^{\frac{-7}{6}} n^{\frac{1}{6}})^{\frac{3}{2}} = 2^{-1} n.$$

Do đó ta có được,

$$|g(x)| < n.$$

Vì $g(x) \equiv 0 \pmod{n}$, nên ta có $g(x) = 0$. Từ đó , ta có thể tìm được tối đa 3 nghiệm của $g(x)$. Sau đó ta sẽ thử xem nó có đưa ra bản mã chính xác hay không. Do đó Eve có thể tìm thấy x.

Lưu ý rằng phương pháp trên thay thế cho vấn đề tìm giải pháp cho giải phương trình đồng dư $f(T) \equiv 0 \pmod{n}$ bằng việc giải phương trình đa thức $g(T) = 0$. Việc giải phương trình đồng dư thường yêu cầu phân tích n , nhưng việc giải phương trình đa thức có thể được thực hiện bằng các phương pháp như phương pháp Newton. Theo cách tương tự, ta có thể tìm nghiệm nhỏ (nếu chúng tồn tại) cho một đa thức đồng dư bậc d, sử dụng một lưới có kích thước d + 1. Tất nhiên, d phải nhỏ để thuật toán LLL chạy trong một khoảng thời gian hợp lý. Cải tiến cho phương pháp này , Coppersmith đã đưa ra ,một thuật toán sử dụng các lưới có nhiều chiều hơn để tìm nghiệm nhỏ cho phương trình đa thức $f(T) \equiv 0 \pmod{n}$(có hệ số bậc cao nhất bằng 1) có bậc d. Nếu $|x| \leq n^{\frac{1}{d}}$, thì thuật toán chạy theo thời gian đa thức trong $log(n)$ và d.

Tổng quát: Giả sử hệ thống RSA có số mũ e thấp và modulus N được công khai , và nó được sử dụng để gửi thông báo có dạng $(M + x)^e \pmod{N}$, trong đó M đã biết và $0 < x < Y$.
Để phá vỡ mã hóa RSA sẽ yêu cầu giải phương trình đồng dư $c \equiv (M + x)^e \pmod{N}$. Từ đây ta có phương trình đồng dư:

$$x^n + a_{n-1} x^{n-1} + ... + a_2 x^2 + a_1 x + a_0 \equiv 0 \pmod{N}.$$

Trong đó $0 \leq x \leq Y$ với Y đã biết. Ta có được lưới bởi cơ sở như sau:

$$\vec{v_1} = (N, 0, 0, ..., 0, 0)$$
$$\vec{v_2} = (0, YN, 0, ..., 0, 0)$$
$$\vdots$$
$$\vec{v_n} = (0, 0, 0, ..., Y^{n-1}N, 0)$$
$$v_{n+1} = (a_0, a_1N, \vec{...}, a_{n-1}N^{n-1}, N^n)$$

Dùng thuật toán LLL để thực hiện giảm lưới. Ta thu được cơ sở mới sau khi giảm lưới $b_0, b_1, b_2, ..., b_{n+1}$. Ta sử dụng vector b_0 như vector ngắn nhất của lưới và chuyển về dạng đa thức:

$$b_0 + \frac{b_1}{Y}x + ... + \frac{b_{n-1}}{m^{n-1}}x^{n-1} + \frac{b_n}{m^n}x^n.$$

Sau đó thực hiện giải phương trình đa thức tìm nghiệm nguyên bài toán[7].

3.1.3 Thực hiện chương trình

Thực hiện chạy chương trình trên phần mềm SageMath .

```
# Ham tan cong RSA
def RSA(pol,N,X):
    d = pol.degree()

    # Chuyen doi ve da thuc nguyen va dat bien x
    polZ = pol.change_ring(ZZ)
    x = polZ.parent().gen()

    # Xay dung luoi B
    BB = Matrix(ZZ, d + 1,d+1)

    for i in range(d + 1) :

        for j in range(d):
            if i == j :
                BB[i,j] = N*X**i
            else :
                BB[i,j] = 0
        BB[i, d] = polZ[i] * X**i
    print("Ma tran luoi B : \n")
    print(BB)
    # Su dung thuat toan LLL
```

```
    BB = LLL(BB,2)
    print("\nma tran giam luoi\n")
    print(BB)
    print("\nvecto B1\n")
    print(BB.column(0))
    # Xay dung da thuc theo vector B1
    new_pol = 0
    for i in range(d+1):
        new_pol += x**i * BB[i, 0] / X**i
    print("\nDa thuc giai nghiem: ")
    print(new_pol)
    # Tim nghiem va kiem tra xem phai nghiem nguyen khong
    potential_roots = new_pol.roots()

    roots = []
    for root in potential_roots:
        if root[0].is_integer():
            result = polZ(ZZ(root[0]))
            roots.append(ZZ(root[0]))

    return roots
#bai toan 1
e = 3
C = 64784502
N = 115348777

ZmodN = Zmod(N);

# Gia tri dau vao
P.<x> = PolynomialRing(ZmodN)
pol = (5180 + x)^e - C
# pol = (2008050001141923051800091900000 + x)^e - C

d = pol.degree()
X = 9
# X = 100

roots = RSA(pol,N,X)
print("\nNghiem can tim:", str(roots))
```

Thực hiện 1 ví dụ : Một hệ mật RSA có khóa công khai $(N, e) = (115348777, 3)$, gửi đi một thông báo có dạng $M = B+x$, trong đó $B = 5180$ và $0 \leq x \leq 9$. Ta cần giải bài toán $c \equiv (B+x)^3 \pmod N$ với $c = 64784502$.

Kết quả thực nghiệm bằng chương trình:

```
Ma tran luoi B :
[ 115348777          0          0   47119990]
[         0 1038138993          0  724474800]
[         0          0 9343250937    1258740]
[         0          0          0        729]

ma tran giam luoi
[ -8023688   9804792   1976288 -49378169]
[   -61119 -22224951  -8613324 -21797118]
[  5411043  12587400 -21659562 -25289901]
[  5414283      7290  32454351 -37892691]

vecto B1

(-8023688, -61119, 5411043, 5414283)

Da thuc giai nghiem:
7427*x^3 + 66803*x^2 - 6791*x - 8023688

Nghiem can tim: [8]
```

Hình 3.1: Chương trình tấn công hệ mật mã RSA.

3.2 Mật mã NTRU

Năm 1998[1], NTRU là hệ mật khóa công khai được đề xuất bởi J.Hoffstein, J.Pipher và J. Silverman. Từ đó tới nay, hệ mật này được nghiên cứu phát triển cải tiến và đã thể hiện được những ưu điểm mà người sử dụng kì vọng về hệ mật sử dụng lưới. Độ an toàn của NTRU vẫn được bảo đảm, trong đó đặc biệt theo đánh giá của NIST, hệ mật NTRU là một trong những hệ mật có khả năng kháng lại tấn công dựa trên tính toán lượng tử tốt nhất. Khi có máy tính lượng tử đủ mạnh, thuật toán Shor sẽ được dùng để thám hệ mật RSA. Hệ mật NTRU có độ an toàn dựa trên lưới, nên hiện chưa có thuật toán nào dùng cho máy tính lượng tử thám được NTRU. Năm 2009, NTRU đã được đưa vào chuẩn mật mã khóa công khai IEEE P1363.1. Năm 2020, NTRU tiếp tục được cải tiến và lọt vào vòng 3 cuộc thi mật mã hậu lượng tử của NIST[1].

Thuật toán NTRU được ứng dụng trong hệ thống kiểm soát và thu thập dữ liệu (Supervisory Control and Data Acquisition System) . Lý do nó được ứng dụng vì có tốc độ làm việc nhanh hơn so với thuật toán RSA và ECC.

3.2.1 Hệ mật mã NTRU

Tập hợp tham số NTRU là $(N, p, q, \zeta_f, \zeta_g, \zeta_\phi, \zeta_m)$ với N là số nguyên tố đủ lớn, p, q là các số nguyên dương nguyên tố cùng nhau (q lớn hơn đáng kể so với p); $\zeta_f, \zeta_g, \zeta_\phi, \zeta_m$ là tập hợp các đa thức bậc $N-1$ với hệ số nguyên. Các phép toán của thuật toán mật mã NTRU được thực hiện trên vành đa thức $Z[X]/(X^N-1)$ gồm các đa thức có hệ số nguyên và bậc nhỏ hơn hoặc bằng $N-1$.
Trong đó, một phần tử $f \in Z[X]/(X^N-1)$ có thể được viết dưới dạng một đa thức hoặc một vector :

$$f = \sum_{i=0}^{N-1} f_i X_i = [f_0, f_1, ..., f_{N-1}].$$

Chúng ta ký hiệu $\otimes$ là phép nhân trên vành đa thức này và được xác định như sau:

$$h = f \otimes g = c_{N-1}X^{N-1} + ... + c_0.$$

Trong đó:

$$\begin{aligned} c_k &= \sum_{i=0}^{k} f_i g_{k-i} + \sum_{i=k+1}^{N-1} f_i g_{N+k-i} \\ &= \sum_{i+j \equiv k \quad mod(N)} f_i g_i. \end{aligned}$$

-**Ví dụ**: Cho $N = 3, f = X^2 + 7X + 9, g = 3X^2 + 2X + 5$. Từ đây ta tính các hệ số của X trong $f \otimes g$.
Ta có : $c_1 = a_0b_1 + a_1b_0 + a_2b_2 = 9.2 + 7.5 + 1.3 = 56$. Tương tự như thế , ta tìm ra:

$$f \otimes g = 46X^2 + 56X + 68.$$

NTRU làm việc với các đa thức có hệ số nhỏ, vì vậy sẽ thuận tiện khi ta ký hiệu cho chúng. Đặt $\zeta(i, k)$ trong đó:

- Đa thức có bậc $< N$,
- Với j hệ số bằng +1 , k hệ số bằng -1 và còn lại bằng 0.

1. Quá trình tạo khóa

Để tạo một khóa cho NTRU, Bob chọn ngẫu nhiên 2 đa thức $f, g \in \zeta_g$. Đa thức f phải có nghịch đảo theo modulo p và modulo q.

Đối với lựa chọn tham số thích hợp, điều này sẽ đúng với hầu hết các lựa chọn của f. Ta ký hiệu các nghịch đảo này là

$$F_p \otimes f \equiv 1 \pmod{p}, F_q \otimes f \equiv 1 \pmod{q}.$$

Tiếp theo Bob tính toán :

$$h \equiv F_q \otimes g \pmod{q}.$$

Khi đó khóa công khai của Bob là (N, p, q, h). Khóa bí mật là đa thức f. Vì $g \equiv f \otimes h \pmod{q}$, nên đa thức g chỉ cần dùng trong quá trình tạo khóa, nhưng cũng cần giữ bí mật vì kết hợp nó với khóa công khai có thể suy ra khóa bí mật.

2. Quá trình mã hóa

Giả sử Alice (người mã hóa) muốn gửi một thông báo cho Bob (người giải mã). Ban đầu, cô ấy chọn một thông báo m từ tập bản rõ ζ_m có bậc nhỏ hơn N với hệ số $\in \left[\frac{-(p-1)}{2}; \frac{p+1}{2}\right]$ (ví dụ với $p = 3$, đa thức m có hệ số $-1, 0, 1$). Tiếp theo, cô ấy chọn đa thức $\phi \in \zeta_\phi$ và sử dụng khóa công khai của Bob để tính toán

$$e \equiv p\phi \otimes h + m \pmod{q}$$

Đây chính là bản mã mà Alice gửi cho Bob.

3. Quá trình giải mã

Bob nhận được thông báo e từ Alice và muốn giải mã nó bằng khóa bí mật f. Để làm được điều này một cách hiệu quả. Bob nên tính trước đa thức F_p. Trước tiên, Bob tính

$$a \equiv f \otimes e \pmod{p},$$

Trong đó, các hệ số của $a \in [-q/2, q/2]$. a là một đa thức hệ số nguyên, Bob khôi phục thông báo dưới dạng

$$m \equiv F_p \otimes \pmod{p}.$$

Tại sao quá trình giải mã hoạt động?

Đa thức a mà Bob đang tính thỏa mãn:

$$\begin{aligned} a &\equiv f \otimes e \equiv f \otimes (p\phi \otimes h + m) \pmod{q} \\ &\equiv f \otimes p\phi \otimes F_q \otimes g + f \otimes m \pmod{q} \\ &\equiv p\phi \otimes g + f \otimes m \pmod{q}. \end{aligned}$$

Xét đa thức $p\phi \otimes g + f \otimes m$. Đối với các lựa chọn tham số phù hợp , chúng ta có thể đảm bảo rằng

(hầu như luôn luôn) tất cả các hệ số của nó nằm trong đoạn $-q/2$ và $q/2$. Vì vậy, nó sẽ không thay đổi khi Bob rút gọn theo mod q. Trong trường hợp này, ta có đẳng thức:

$$a = p\phi \otimes g + f \otimes m.$$

Sau đó :

$$F_p \otimes a = pF_p \otimes \phi \otimes g + F_p \otimes f \otimes m \equiv 0 + 1 \otimes m \equiv m \pmod{p}$$

Vì vậy, việc giải mã hoạt động[2].

- **Ví dụ:** Sau đây là một ví dụ nhỏ về thuật toán NTRU với khóa công khai $(N, p, q) = (7, 3, 41)$. Bob chọn 2 đa thức :

$$f = X^6 - X^4 + X^3 + X^2 - 1 \in \zeta(3, 2),$$

$$g = X^6 + X^4 - X^2 - X \in \zeta(2, 2).$$

Sau đó, anh ấy tính nghịch đảo của đa thức f ta được:

$$F_q = 8X^6 + 26X^5 + 31X^4 + 21X^3 + 40X^2 + 2X + 27.$$

$$F_p = X^6 + 2X^5 + X^3 + X^2 + X + 1.$$

Bob lưu trữ (f, F_p) làm khóa riêng của mình và tính toán:

$$h \equiv F_q \otimes g \equiv 20X^6 + 40X^5 + 2X^4 + 38X^3 + 8X^2 + 26X + 30 \pmod{41}.$$

Alice muốn gửi một thông báo cho Bob là $m = -X^5 + X^3 + X^2 - X + 1$. Chọn ngẫu nhiên đa thức $\phi = X^6 - X^5 + X - 1$. Khi đó bản mã mà Alice muốn gửi cho Bob :

$$e \equiv p\phi \otimes h + m \equiv 31X^6 + 19X^5 + 4X^4 + 2X^3 + 40X^2 + 3X + 25 \pmod{41}.$$

Bob nhận được thông báo e từ Alice và bắt đầu giải mã. Bob tính:

$$\begin{aligned} f \otimes e &\equiv X^6 + 10X^5 + 33X^4 + 40X^3 + 40X^2 + X + 40 \pmod{41} \\ &\equiv X^6 + 10X^5 - 8X^4 - X^3 - X^2 + X - 1 \pmod{41}. \end{aligned}$$

Cuối cùng, Bob tính :

$$\begin{aligned} F_p \otimes a &\equiv 2X^5 + X^3 + X^2 + 2X + 1 \pmod{3} \\ &\equiv -X^5 + X^3 + X^2 - X + 1 \pmod{3}. \end{aligned}$$

Do đó , Bob nhận được thông báo $m = -X^5 + X^3 + X^2 - X + 1$.

Code chạy thực nghiệm:

```
   #Ham giam he so da thuc
def reduce(t,p):
    if (ZZ(t)%p)>(p//2):
        return ((ZZ(t)%p)-p)
    else:
        return ZZ(t)%p

print("Chon khoa cong khai:\n")
#khoa cong khai ban dau
N=7
p=3
q=41
print("(N,p,q)=",(N,p,q))

#Chon da thuc
Zx.<X> = ZZ[]
f=X^6-X^4+X^3+X^2-1
g=X^6+X^4-X^2-X
print("f=",f)
print("g=",g)
Pp.<b>=PolynomialRing(GF(p))
Pq.<c>=PolynomialRing(GF(q))

#Tim nghich dao da thuc
f_p=Pp(f).inverse_mod(b^N-1)
f_q=Pq(f).inverse_mod(c^N-1)
print("f_p=",f_p(b=X))
print("f_q=",f_q(c=X))

#Tinh da thuc h
h=(f_q*Pq(g))%(c^N-1)
print("\nkhoa cong khai h: ",h(c=X))

#chon da thuc phi (thay bang r)
r=X^6-X^5+X-1
print("da thuc phi r=",r)
```

```
#thong bao muon gui
m=-X^5+X^3+X^2-X+1
print("thong bao m:",m)

#Qua trinh giai ma
e=(p*Pq(r)*h)%(c^N-1)+Pq(m)%(c^N-1)
print("\n Ma hoa thong bao: e= ", e(c=X))
A=(Pq(f)*e)%(c^N-1)
print("tinh da thuc a : ",A(c=X))
A1=[reduce(k,q) for k in A.list()]
print(Zx(A1))
print("\n giai ma thong bao m: ", Zx([reduce(k,p) for k in
    Pp((Zx(A1)*Zx(f_p))%(X^N-1)).list()]))
```

-**Ví dụ chạy chương trình**: Với khóa công khai $(N, p, q) = (7, 3, 41)$. Chọn 2 đa thức :$f = X^6 - X^4 + X^3 + X^2 - 1 \in \zeta(3, 2),, g = X^6 + X^4 - X^2 - X \in \zeta(2, 2)$.. thông báo muốn gửi là $m = -X^5 + X^3 + X^2 - X + 1$. Chọn đa thức $\phi = X^6 - X^5 + X - 1$. Và sau đây là kết quả chạy chương trình:

```
Chon khoa cong khai:

(N,p,q)= (7, 3, 41)
f= X^6 - X^4 + X^3 + X^2 - 1
g= X^6 + X^4 - X^2 - X
f_p= X^6 + 2*X^5 + X^3 + X^2 + X + 1
f_q= 8*X^6 + 26*X^5 + 31*X^4 + 21*X^3 + 40*X^2 + 2*X + 37

khoa cong khai h:  20*X^6 + 40*X^5 + 2*X^4 + 38*X^3 + 8*X^2 + 26*X + 30
da thuc phi r= X^6 - X^5 + X - 1
thong bao m: -X^5 + X^3 + X^2 - X + 1

 Ma hoa thong bao: e=  31*X^6 + 19*X^5 + 4*X^4 + 2*X^3 + 40*X^2 + 3*X + 25
tinh da thuc a :  X^6 + 10*X^5 + 33*X^4 + 40*X^3 + 40*X^2 + X + 40
X^6 + 10*X^5 - 8*X^4 - X^3 - X^2 + X - 1

 giai ma thong bao m:  -X^5 + X^3 + X^2 - X + 1
```

Hình 3.2: Kết quả tấn công hệ mật mã NTRU

3.2.2 Tấn công hệ mật mã NTRU dựa trên lý thuyết lưới

Đặt $h = h_{N-1}X^{N-1} + ... + h_0$. Ta lập được ma trận $N \times N$ sau :

$$H = \begin{pmatrix} h_0 & h_1 & \cdots & h_{N-1} \\ h_{N-1} & h_0 & \cdots & h_{N-2} \\ \vdots & \vdots & \ddots & \vdots \\ h_1 & h_2 & \cdots & h_0 \end{pmatrix}.$$

Nếu ta biểu diễn $f = f_{N-1}X^{N-1} + ... + f_0$ và $g = g_{N-1}X^{N-1} + ... + g_0$ theo các vector hàng $\overline{f} = (f_0, f_1, ..., f_{N-1})$ và $\overline{g} = (g_0, g_1, ..., g_{N-1})$ thì ta thấy rằng:

$$\overline{f}H \equiv \overline{g} \pmod{q}.$$

Với I là ma trận đơn vị $N \times N$. Ta sẽ hình thành ma trận $2N \times 2N$:

$$M = \begin{pmatrix} I & H \\ 0 & qI \end{pmatrix}.$$

Gọi L là lưới được tạo bởi các hàng của M. Vì $g \equiv f \otimes h \mod q$. từ đó ta có được $g = f \otimes h + qy$, với hệ số đa thức y được biểu diễn dưới dạng vactor hàng $\overline{y}$ có N chiều. Do đó $(\overline{f}, \overline{y})$ là vector hàng $2N$ chiều. Ta có thể viết lại được như sau:

$$(\overline{f}, \overline{y})M = (\overline{f}, \overline{g}),$$

nên $(\overline{f}, \overline{g}) \in L$. Vì 2 đa thức f, g có hệ số nhỏ nên $(\overline{f}, \overline{g})$ là một vector ngắn trong lưới L. Do đó, thông tin bí mật cho khóa có thể được biểu diễn dưới dạng một vector ngắn trong lưới. Kẻ tấn công có thể áp dụng thuật toán giảm lưới để thu được các vector ngắn và có thể thu được vector $(\overline{f}, \overline{g})$. Khi kẻ tấn công tìm thấy f, g, thì hệ thống sẽ bị hỏng.

Để ngăn chặn các cuộc tấn công vào lưới , chúng ta cần làm cho lưới có kích thước đủ lớn để thuật toán giảm lưới không hiệu quả. Điều này ta có thể dễ dàng thực hiện bằng cách cho N đủ lớn. Tuy nhiên, nếu N quá lớn, thì thuật toán mã hóa và giải mã sẽ trở nên chậm. Do đó, các giá trị của N sẽ được chọn để vừa đạt tính bảo mật trong khi vẫn giữ cho các thuật toán mật mã hiệu quả.

Chúng ta có thể cải tiến kỹ thuật tìm vector ngắn nhất nhanh hơn khi giảm hằng số lưới nhỏ đi. Với lưới L có ma trận cơ sở là:

$$\begin{pmatrix} \lambda I & H \\ 0 & qI \end{pmatrix}.$$

với λ là một số thực phù hợp. Điều này làm cho kết quả vector $(\lambda\overline{f}, \overline{g})$ tương đối ngắn hơn và dễ tìm hơn. Những thông số trong NTRU, đặc biệt là kích thước của f và g sẽ được chọn để hạn chế ảnh

hưởng của các cuộc tấn công vào lưới này[4].

Tổng kết

Trên đây là toàn văn báo cáo Đồ án 1 về chủ đề **Phương pháp lưới**. Trong đồ án này, em đã trình bày một số kiến thức cơ bản về lý thuyết lưới, tìm hiểu về thuật toán LLL và nhận thấy một phần tầm quan trọng của nó tróng lý thuyết số và mật mã.Trong bài báo cáo có giới thiệu về một số hệ mật mã như RSA và NTRU ,thực hiện các cuộc tấn công vào chúng và thực hiện chạy chương trình thực nghiệm trên phần mềm SageMath.

Báo cáo Đồ án vẫn còn rất nhiều thiếu sót, vậy nên em rất mong được thầy và các bạn đọc cùng góp ý, nhận xét để báo cáo trở nên hoàn thiện hơn.

Em xin chân thành cảm ơn!

Tài liệu tham khảo

[1] Phạm Quốc Hoàng, Phạm Thị Hiền,"Phân tích độ an toàn của thuật toán mật mã NTRU",Tạp chí An toàn thông tin, Số 5 (063), 2021.

[2] Jeffrey Hofstein, Jill Pipher, Joseph H. Silverman, *NTRU: A Ring-Based Public Key Cryptosystem*, https://www.ntru.org/f/hps98.pdf.

[3] Khúc Xuân Thành, *Luận văn thạc sĩ khoa học - Lý thuyết lưới và ứng dụng trong mật mã*, 2017.

[4] Lawrence C. Washington and Wade Trappe, *Introduction to Cryptography with Coding Theory*,Pearson Education International, 2002.

[5] Hệ mật mã RSA, web *vi.wikipedia.org*

[6] Lenstra–Lenstra–Lovász lattice basis reduction algorithm, web *en.wikipedia.org*

[7] Thuật toán LLL và RSA, web *www.youtube.com*

[8] Toán rời rạc và Khoa học máy tính đã đi vào trung tâm của toán học, web *vietnamnet.vn*

Chỉ mục