**TRƯỜNG ĐẠI HỌC HÀNG HẢI VIỆT NAM  
KHOA CÔNG NGHỆ THÔNG TIN**

**-----\*\*\*-----**

****

**BÁO CÁO BÀI TẬP LỚN**

**HỌC PHẦN “AN TOÀN BẢO MẬT THÔNG TIN”**

***Đề tài:***

***< Mã hóa và giải mã mật mã Affine, Vigenere,Hill>***

***GVHD: ThS. Phạm Tuấn Đạt***

***Sinh viên thực hiện: <Phạm Bá Huy> – Mã sv: < 87726>***

***<Dương Thế Khang> – Mã sv: <86956 >***

***Hải Phòng, tháng 05 năm 2022***

**TRƯỜNG ĐẠI HỌC HÀNG HẢI**

**KHOA CÔNG NGHỆ THÔNG TIN**

**BỘ MÔN KHOA HỌC MÁY TÍNH**

**-----\*\*\*-----**

**BÀI TẬP LỚN**

**HỌC PHẦN: AN TOÀN BẢO MẬT THÔNG TIN**

**Mã đề tài: < 006>**

1. **Tên đề tài**

***< Mã hóa và giải mã mật mã Affine, Vigenere,Hill >***

1. **Mục đích**

< Tìm hiểu về mã hóa và giải mã>

1. **Công việc cần thực hiện**

* Viết chương trình
* Làm báo cáo bài tập lớn
* Bảo vệ bài tập lớn

1. **Yêu cầu**

* Kết quả làm bài tập lớn: Báo cáo bài tập lớn
* Báo cáo bài tập lớn phải được trình bày theo mẫu quy định (kèm theo), báo cáo có thể kết xuất thành tệp định dạng PDF và nộp qua email (không bắt buộc phải in ấn)
* Hạn nộp báo cáo bài tập lớn: Tuần 14/15

1. **Tài liệu tham khảo**
   * Nguồn code trên các diễn đàn tin học.

***Hải Phòng, tháng 05 năm 2022***

**NGƯỜI HƯỚNG DẪN**

MỤC LỤC

[**DANH MỤC CÁC HÌNH VẼ, BẢNG BIỂU iii**](#_1fob9te)

[**DANH MỤC CÁC TỪ VIẾT TẮT iv**](#_3znysh7)

[**GIỚI THIỆU 1**](#_2et92p0)

[**CHƯƠNG 1. CƠ SỞ LÝ THUYẾT 3**](#_tyjcwt)

[1.1.Mật mã Affine](#_3dy6vkm)

[1.2.Mật mã Vigenere](#_3dy6vkm)

[1.3.Mật mã Hill](#_2s8eyo1)

[**CHƯƠNG 2. BÀI TOÁN 14**](#_26in1rg)

[2.1. Mật mã Affine](#_lnxbz9)

[2.2. Mật mã Vigenere](#_35nkun2)

[2.3. Mật mã Hill](#_1ksv4uv)

[**CHƯƠNG 3. KẾT QUẢ THỬ NGHIỆM 20**](#_44sinio)

[3.1. Mật mã Affine](#_2jxsxqh)

[3.2. Mật mã Vigenere](#_z337ya)

[3.3. Mật mã Hill](#_z337ya)

**KÊT QUẢ** [22](#_z337ya)

[**TÀI LIỆU THAM KHẢO 23**](#_1y810tw)

# DANH MỤC CÁC HÌNH VẼ, BẢNG BIỂU

|  |  |
| --- | --- |
| **Hình vẽ** | **Trang** |
| Hình 1.1. Kênh liên lạc | 2 |
| Hình 1.2. Mật mã Affine | 7 |
| Hình 1.3. Mật mã Vigenère | 8 |
| Hình 1.4. Mật mã Hill | 15 |

# DANH MỤC CÁC TỪ VIẾT TẮT

|  |  |
| --- | --- |
| **Từ** | **Ý nghĩa** |
| … |  |

# GIỚI THIỆU

Đối tượng cơ bản của mật mã là tạo ra khả năng liên lạc trên một kênh không mật cho hai người sử dụng (tạm gọi là Alice và Bob) sao cho đối phương (Oscar) không thể hiểu được thông tin được truyền đi. Kênh này có thể là một đường dây điện thoại hoặc một mạng máy tính. Thông tin mà Alice muốn gửi cho Bob (bản rõ) có thể là một văn bản tiếng Anh, các dữ liệu bằng số hoặc bất cứ tài liệu nào có cấu trúc tuỳ ý. Alice sẽ mã hoá bản rõ bằng một kháo đã được xacs định trước và gửi bản mã kết quả trên kênh. Oscar có bản mã thu trộm được trên kênh song không thể xác định nội dung của bản rõ, nhưng Bob (người đã biết khoá mã) có thể giải mã và thu được bản rõ.

Ta sẽ mô tả hình thức hoá nội dung bằng cách dung khái niệm toán học như sau:

***Định nghĩa 1.1***

*Một hệ mật là một bộ 5 (P,C,K,E,D) thoả mãn các điều kiện sau:*

1. *P là một tập hữu hạn các bản rõ có thể.*
2. *C là một tập hữu hạn các bản mã có thể.*
3. *K (không gian khoá) là tập hữu hạn các khoá có thể.*
4. *Đối với mỗi k∈ K có một quy tắc mã ek: P → C và một quy tắcv giải mã tương ứng dk ∈ D. Mỗi ek:* *P → C và dk: C → P là những hàm mà:*

*dk(ek (x)) = x với mọi bản rõ x ∈ P.*

Trong tính chất 4 là tính chất chủ yếu ở đây. Nội dung của nó là nếu một bản rõ x được mã hoá bằng *ek* và bản mã nhận được sau đó được giải mã bằng *dk* thì ta phải thu được bản rõ ban đầu x. Alice và Bob sẽ áp dụng thủ tục sau dùng hệ mật khoá riêng. Trước tiên họ chọn một khoá ngẫu nhiên K *∈ K* . Điều này được thực hiện khi họ ở cùng một chỗ và không bị Oscar theo dõi hoặc khi họ có một kênh mật trong trường hợp họ ở xa nhau. Sau đó giả sử Alice muốn gửi một thông baó cho Bob trên một kênh không mật và ta xem thông báo này là một chuỗi:

x = x1,x2 ,. . .,xn

với số nguyên n ≥ 1 nào đó. Ở đây mỗi ký hiệu của mỗi bản rõ xi *∈ P* , 1 ≤ i ≤ n. Mỗi xi sẽ được mã hoá bằng quy tắc mã ek với khoá K xác định trước đó. Bởi vậy Alice sẽ tính yi = ek(xi), 1 ≤ i ≤ n và chuỗi bản mã nhận được:

y = y1,y2 ,. . .,yn

sẽ được gửi trên kênh. Khi Bob nhận đươc y1,y2 ,. . .,yn anh ta sẽ giải mã bằng hàm giải mã dk và thu được bản rõ gốc x1,x2 ,. . .,xn. Hình 1.1 là một ví dụ về một kênh liên lạc

***Hình 1.1. Kênh liên lạc***

Oscar

Bộ giải mã

Bộ mã hoá

Bob

Alice

Kênh an toàn

Nguồn khoá

Rõ ràng là trong trường hợp này hàm mã hoá phải là hàm đơn ánh ( tức là ánh xạ 1-1), nếu không việc giải mã sẽ không thực hiện được một cách tường minh. Ví dụ

y = ek(x1) = ek(x2)

trong đó x1 ≠ x2 , thì Bob sẽ không có cách nào để biết liệu sẽ phải giải mã thành x1 hay x2 . Chú ý rằng nếu *P = C* thì mỗi hàm mã hoá ize="2">Bản quyền Công ty Phát ttập các bản mã và tập các bản rõ là đồng nhất thì mỗi một hàm mã sẽ là một sự sắp xếp lại (hay hoán vị ) các phần tử của tập này.

**CHƯƠNG 1. CƠ SỞ LÝ THUYẾT**

***1.1Mã Affine***

MDV là một trường hợp đặc biệt của MTT chỉ gồm 26 trong số 26! các hoán vị có thể của 26 phần tử. Một trường hợp đặc biệt khác của MTT là mã Affine được mô tả dưới đây. trong mã Affine, ta giới hạn chỉ xét các hàm mã có dạng:

e(x) = ax + b mod 26,

a,b ∈ Z26 . Các hàm này được gọi là các hàm Affine (chú ý rằng khi a = 1, ta có MDV).

Để việc giải mã có thể thực hiện được, yêu cầu cần thiết là hàm Affine phải là đơn ánh. Nói cách khác, với bất kỳ y ∈ Z26, ta muốn có đồng nhất thức sau:

ax + b ≡ y (mod 26)

phải có nghiệm x duy nhất. Đồng dư thức này tương đương với:

ax ≡ y-b (mod 26)

Vì y thay đổi trên Z26 nên y-b cũng thay đổi trên Z26 . Bởi vậy, ta chỉ cần nghiên cứu phương trình đồng dư:

ax ≡ y (mod 26) (y∈ Z26 ).

Ta biết rằng, phương tfình này có một nghiệm duy nhất đối với mỗi y khi và chỉ khi UCLN(a,26) = 1 (ở đây hàm UCLN là ước chung lớn nhất của các biến của nó). Trước tiên ta giả sử rằng, UCLN(a,26) = d >1. Khi đó, đồng dư thức ax ≡ 0 (mod 26) sẽ có ít nhất hai nghiệm phân biệt trong Z26 là x = 0 và x = 26/d. Trong trường hợp này, e(x) = ax + b mod 26 không phải là một hàm đơn ánh và bởi vậy nó không thể là hàm mã hoá hợp lệ.

Ví dụ, do UCLN(4,26) = 2 nên 4x +7 không là hàm mã hoá hợp lệ: x và x+13 sẽ mã hoá thành cùng một giá trị đối với bất kì x ∈ Z26 .

Ta giả thiết UCLN(a,26) = 1. Giả sử với x1 và x2 nào đó thảo mãn:

ax1 ≡ ax2 (mod 26)

Khi đó

a(x1- x2) ≡ 0(mod 26)

bởi vậy

26 | a(x1- x2)

Bây giờ ta sẽ sử dụng một tính chất của phép chia sau: Nếu USLN(a,b)=1 và a ⏐bc thì a ⏐c. Vì 26 ⏐ a(x1- x2) và USLN(a,26) = 1 nên ta có:

26⏐(x1- x2)

tức là

x1 ≡ x2 (mod 26)

Tới đây ta chứng tỏ rằng, nếu UCLN(a,26) = 1 thì một đồng dư thức dạng ax ≡ y (mod 26) chỉ có (nhiều nhất) một nghiệm trong Z26 . Do đó , nếu ta cho x thay đổi trên Z26  thì ax mod 26 sẽ nhận được 26 giá trị khác nhau theo modulo 26 và đồng dư thức ax ≡ y (mod 26) chỉ có một nghiệm y duy nhất.

Không có gì đặc biệt đối vơí số 26 trong khẳng định này. Bởi vậy, bằng cách tương tự ta có thể chứng minh được kết quả sau:

***Định lí 1.1***

*Đồng dư thức ax ≡ b mod m chỉ có một nghiệm duy nhất x ∈ Zm với mọi b ∈ Zm khi và chỉ khi UCLN(a,m) = 1.*

Vì 26 = 2 ×13 nên các giá trị a ∈ Z26 thoả mãn UCLN(a,26) = 1 là a = 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23 và 25. Tham số b có thể là một phần tử bất kỳ trong Z26 . Như vậy, mã Affine có 12 × 26 = 312 khoá có thể ( dĩ nhiên con số này quá nhỉ để bảo đảm an toàn).

Bây giờ ta sẽ xét bài toán chung với modulo m. Ta cần một định nghĩa khác trong lý thuyết số.

***Định nghĩa 1.2***

*Giả sử a ≥ 1 và m ≥ 2 là các số nguyên. UCLN(a,m) = 1 thì ta nói rằng a và m là nguyên tố cùng nhau. Số các số nguyên trong Zm nguyên tố cùng nhau với m thường được ký hiệu là φ(m) ( hàm này được gọi là hàm Euler).*

Một kết quả quan trọng trong lý thuyết số cho ta giá trị của φ(m) theo các thừa số trong phép phân tích theo luỹ thừa các số nguyên tố của m. ( Một số nguyên p >1 là số nguyên tố nếu nó không có ước dương nào khác ngoài 1 và p. Mọi số nguyên m >1 có thể phân tích được thành tích của các luỹ thừa các số nguyên tố theo cách duy nhất. Ví dụ 60 = 2 3 × 3 × 5 và 98 = 2 × 7 2 ).

Ta sẽ ghi lại công thức cho φ(m) trong định lí sau:

***Định lý 1.2.*** ( thiếu )

Giả sử m = ∏ *pi*

Trong đó các số nguyên tố p*i* khác nhau và *ei* >0 ,1

Định lý này cho thấy rằng, số khoá trong mã Affine trên Zm bằng mφ(m), trong đó φ(m) được cho theo công thức trên. ( Số các phép chọn của b là m và số các phép chọn của a là φ(m) với hàm mã hoá là e(x) = ax + b). Ví dụ, khi m = 60, φ(60) = 2 × 2 × 4 = 16 và số các khoá trong mã Affine là 960.

Bây giờ ta sẽ xét xem các phép toán giải mã trong mật mã Affine với modulo m = 26. Giả sử UCLN(a,26) = 1. Để giải mã cần giải phương trình đồng dư y ≡ax+b (mod 26) theo x. Từ thảo luận trên thấy rằng, phương trình này có một nghiệm duy nhất trong Z26 . Tuy nhiên ta vẫn chưa biết một phương pháp hữu hiệu để tìm nghiệm. Điều cần thiết ở đây là có một thuật toán hữu hiệu để làm việc đó. Rất mayb là một số kết quả tiếp sau về số học modulo sẽ cung cấp một thuật toán giải mã hữu hiệu cần tìm.

***Định nghĩa 1.3***

*Giả sử a ∈ Zm . Phần tử nghịch đảo (theo phép nhân) của a là phần tử a-1 ∈ Zm sao cho aa-1 ≡ a-1a ≡ 1 (mod m).*

Bằng các lý luận tương tự như trên, có thể chứng tỏ rằng a có nghịch đảo theo modulo m khi và chỉ khi UCLN(a,m) =1, và nếu nghịch đảo này tồn tại thì nó phải là duy nhất. Ta cũng thấy rằng, nếu b = a-1 thì a = b-1 . Nếu p là số nguyên tố thì mọi phần tử khác không của ZP đều có nghịch đảo. Một vành trong đó mọi phần tử đều có nghịch đảo được gọi là một trường.

Trong phần sau sẽ mô tả một thuật toán hữu hiệu để tính các nghịch đảo của Zm với m tuỳ ý. Tuy nhiên, trong Z26 , chỉ bằng phương pháp thử và sai cũng có thể tìm được các nghịch đảo của các phần tử nguyên tố cùng nhau với 26: 1-1 = 1, 3-1 = 9, 5-1 = 21, 7-1 = 15, 11-1 = 19, 17-1 =23, 25-1 = 25. (Có thể dễ dàng kiểm chứng lại điều này, ví dụ: 7 × 5 = 105 ≡ 1 mod 26, bởi vậy 7-1 = 15).

Xét phương trình đồng dư y ≡ ax+b (mod 26). Phương trình này tương đương với

ax ≡ y-b ( mod 26)

Vì UCLN(a,26) =1 nên a có nghịch đảo theo modulo 26. Nhân cả hai vế của đồng dư thức với a-1 ta có:

a-1(ax) ≡ a-1(y-b) (mod 26)

Áp dụng tính kết hợp của phép nhân modulo:

a-1(ax) ≡ (a-1a)x ≡ 1x ≡ x.

Kết quả là x ≡ a-1(y-b) (mod 26). Đây là một công thức tường minh cho x. Như vậy hàm giải mã là:

d(y) = a-1(y-b) mod 26

Hình 1.4 cho mô tả đầy đủ về mã Affine. Sau đây là một ví dụ nhỏ

***Hình 1.2 Mật mãA ffine***

Cho *P = C* = Z26 và giả sử

*P* = { (a,b) ∈ Z26 × Z26 : UCLN(a,26) =1 }

Với K = (a,b) ∈*K*  , ta định nghĩa:

eK(x) = ax +b mod 26

và

dK(y) = a-1(y-b) mod 26,

x,y ∈ Z26

*Ví dụ 1.1*

Giả sử K = (7,3). Như đã nêu ở trên, 7-1 mod 26 = 15. Hàm mã hoá là

eK(x) = 7x+3

Và hàm giải mã tương ứng là:

dK(x) = 15(y-3) = 15y -19

Ở đây, tất cả các phép toán đều thực hiện trên Z26. Ta sẽ kiểm tra liệu dK(eK(x)) = x với mọi x ∈ Z26 không?. Dùng các tính toán trên Z26 , ta có

dK(eK(x)) =dK(7x+3)

=15(7x+3)-19

= x +45 -19

= x.

Để minh hoạ, ta hãy mã hoá bản rõ "*hot*". Trước tiên biến đổi các chữ h, o, t thành các thặng du theo modulo 26. Ta được các số tương ứng là 7, 14 và 19. Bây giờ sẽ mã hoá:

7 × 7 +3 mod 26 = 52 mod 26 = 0

7 × 14 + 3 mod 26 = 101 mod 26 =23

7 × 19 +3 mod 26 = 136 mod 26 = 6

Bởi vậy 3 ký hiệu của bản mã là 0, 23 và 6 tương ứng với xâu ký tự AXG. Việc giải mã sẽ do bạn đọc thực hiện như một bài tập.

***1.2Mã Vigenère***

Trong cả hai hệ MDV và MTT (một khi khoá đã được chọn) mỗi ký tự sẽ được ánh xạ vào một ký tự duy nhất. Vì lý do đó, các hệ mật còn được gọi hệ thay thế đơn biểu. Bây giờ ta sẽ trình bày ( trong hùnh 1.5) một hệ mật không phải là bộ chữ đơn, đó là hệ mã Vigenère nổi tiếng. Mật mã này lấy tên của Blaise de Vigenère sống vào thế kỷ XVI.

Sử dụng phép tương ứng A ⇔ 0, B ⇔ 1, . . . , Z ⇔ 25 mô tả ở trên, ta có thể gắn cho mỗi khoa K với một chuỗi kí tự có độ dài m được gọi là từ khoá. Mật mã Vigenère sẽ mã hoá đồng thời m kí tự: Mỗi phần tử của bản rõ tương đương với m ký tự.

Xét một ví dụ nhỏ

*Ví dụ 1.2*

Giả sử m =6 và từ khoá là CIPHER. Từ khoá này tương ứng với dãy số K = (2,8,15,4,17). Giả sử bản rõ là xâu:

*thiscryptosystemisnotsecure*

***Hình 1.3 Mật mã Vigenère***

Cho m là một số nguyên dương cố định nào đó. Định nghĩa *P = C = K* = (Z26)m . Với khoá K = (k1, k2, . . . ,km) ta xác định :

eK(x1, x2, . . . ,xm) = (x1+k1, x2+k2, . . . , xm+km)

và

dK(y1, y2, . . . ,ym) = (y1-k1, y2-k2, . . . , ym-km)

trong đó tất cả các phép toán được thực hiện trong Z26

Ta sẽ biến đổi các phần tử của bản rõ thành các thặng dư theo modulo 26, viết chúng thành các nhóm 6 rồi cộng với từ khoá theo modulo 26 như sau:

19 7 8 18 2 17 24 15 19 14 18 24

2 8 15 7 4 17 2 8 15 7 4 17

21 15 23 25 6 8 0 23 8 21 22 15

18 19 4 12 8 18 13 14 19 18 4 2

2 8 15 7 4 17 2 8 15 7 4 17

20 1 19 19 12 9 15 22 8 15 8 19

20 17 4

2 8 15

22 25 19

Bởi vậy, dãy ký tự tương ứng của xâu bản mã sẽ là:

V P X Z G I A X I V W P U B T T M J P W I Z I T W Z T

Để giải mã ta có thể dùng cùng từ khoá nhưng thay cho cộng, ta trừ cho nó theo modulo 26.

Ta thấy rằng các từ khoá có thể với số độ dài m trong mật mã Vigenère là 26m, bởi vậy, thậm chí với các giá trị m khá nhỏ, phương pháp tìm kiếm vét cạn cũng yêu cầu thời gian khá lớn. Ví dụ, nếu m = 5 thì không gian khoá cũng có kích thước lớn hơn 1,1 × 107 . Lượng khoá này đã đủ lớn để ngaen ngừa việc tìm khoá bằng tay( chứ không phải dùng máy tính).

Trong hệ mật Vigenère có từ khoá độ dài m,mỗi ký tự có thể được ánh xạ vào trong m ký tự có thể có (giả sử rằng từ khoá chứa m ký tự phân biệt). Một hệ mật như vậy được gọi là hệ mật thay thế đa biểu (polyalphabetic). Nói chung, việc thám mã hệ thay thế đa biểu sẽ khó khăn hơn so việc thám mã hệ đơn biểu.

***1.3 Mật mã Hill***

Trong phần này sẽ mô tả một hệ mật thay thế đa biểu khác được gọi là mật mã Hill. Mật mã này do Lester S.Hill đưa ra năm 1929. Giả sử m là một số nguyên dương, đặt *P = C =*  (Z26)m . Ý tưởng ở đây là lấy m tổ hợp tuyến tính của m ký tự trong một phần tử của bản rõ để tạo ra m ký tự ở một phần tử của bản mã.

Ví dụ nếu m = 2 ta có thể viết một phần tử của bản rõ là x = (x1,x2) và một phần tử của bản mã là y = (y1,y2). Ở đây, y1cũng như y2 đều là một tổ hợp tuyến tính của x1và x2. Chẳng hạn, có thể lấy

y1 = 11x1+ 3x2

y2 = 8x1+ 7x2

Tất nhiên có thể viết gọn hơn theo ký hiệu ma trận như sau

(y1 y2) = (x1 x2)

11 8

3 7

Nói chung, có thể lấy một ma trận K kích thước m × m làm khoá. Nếu một phần tử ở hàng i và cột j của K là ki,,j thì có thể viết K = (ki,,j), với x = (x1, x2, . . . ,xm) ∈ *P* và K ∈*K* , ta tính y = eK(x) = (y1, y2, . . . ,ym) như sau:

k1,1 k1,2 ... k1,m

k2,1 k2,2 ... k2,m

... ... ... . .

km,1 km,2 ... km,m

(y1,. . .,ym) (x1, . . . ,xm)

Nói một cách khác y = xK.

Chúng ta nói rằng bản mã nhận được từ bản rõ nhờ phép biến đổi tuyến tính. Ta sẽ xét xem phải thực hiện giải mã như thế nào, tức là làm thế nào để tính x từ y. Bạn đọc đã làm quen với đại số tuyến tính sẽ thấy rằng phải dùng ma trận nghịch đảo K-1 để giả mã. Bản mã được giải mã bằng công thức y K-1 .

Sau đây là một số định nghĩa về những khái niệm cần thiết lấy từ đại số tuyến tính. Nếu A = (xi,j) là một ma trận cấp l × m và B = (b1,k ) là một ma trận cấp m × n thì tích ma trận AB = (c1,k ) được định nghĩa theo công thức:

m

c1,k  = Σ ai,j bj,k

j=1

Với 1 ≤ i ≤ l và 1 ≤ k ≤ l. Tức là các phần tử ở hàng i và cột thứ k của AB được tạo ra bằng cách lấy hàng thứ i của A và cột thứ k của B, sau đó nhân tương ứng các phần tử với nhau và cộng lại. Cần để ý rằng AB là một ma trận cấp l × n.

Theo định nghĩa này, phép nhân ma trận là kết hợp (tức (AB)C = A(BC)) nhưng noiâ chung là không giao hoán ( không phải lúc nào AB = BA, thậm chí đố với ma trận vuông A và B).

Ma trận đơn vị m × m (ký hiệu là Im ) là ma trận cấp m × m có các số 1 nằm ở đường chéo chính và các số 0 ở vị trí còn lại. Như vậy ma trận đơn vị 2 × 2 là:

1. 0

0 1

Im được gọi là ma trận đơn vị vì AIm = A với mọi ma trận cấp l × m và ImB =B với mọi ma trận cấp m × n. Ma trận nghịch đảo của ma trận A cấp m × m ( nếu tồn tại) là ma trận A-1 sao cho AA-1 = A-1A = Im . Không phải mọi ma trận đều có nghịch đảo, nhưng nếu tồn tại thì nó duy nhất.

I2 =

Với các định nghĩa trên, có thể dễ dàng xây dựng công thức giải mã đã nêu: Vì y = xK, ta có thể nhân cả hai vế của đẳng thức với K-1 và nhận được:

yK-1 = (xK)K-1 = x(KK-1) = xIm = x

( Chú ý sử dụng tính chất kết hợp)

Có thể thấy rằng, ma trận mã hoá ở trên có nghịch đảo trong Z26:

1. 8

3 7

-1

=

1. 18

23 11

vì

1. 8

3 7

1. 18

23 11

=

11×7+8×23 11×18+8×11

3×7+7×23 3×18+7×11

(Hãy nhớ rằng mọi phép toán số học đều được thực hiện theo modulo 26).

=

261 286

182 131

=

1. 0

0 1

Sau đây là một ví dụ minh hoạ cho việc mã hoá và iải mã trong hệ mật mã Hill.

*Ví dụ 1.4*

=

1. 8

3 7

Từ các tính toán trên ta có:

Giả sử khoá K

K-1 =

1. 18

23 11

Giả sử cần mã hoá bản rõ "July". Ta có hai phần tử của bản rõ để mã hoá: (9,20) (ứng với Ju) và (11,24) (ứng với ly). Ta tính như sau:

(9,20)

1. 8

3 7

= (99+60, 72+140) = (3,4)

và

(11,24)

1. 8

3 7

= (121+72, 88+168) = (11,22)

Bởi vậy bản mã của July là DELW. Để giải mã Bob sẽ tính

(3,4)

1. 18

23 11

= (9,20)

(11,22)

1. 18

23 11

= (11,24)

và

Như vậy Bob đã nhận được bản đúng.

Cho tới lúc này ta đã chỉ ra rằng có thể thực hiện phép giải mã nếu K có một nghịch đảo. Trên thực tế, để phép giải mã là có thể thực hiện được, điều kiện cần là K phải có nghịch đảo. ( Điều này dễ dàng rút ra từ đại số tuyến tính sơ cấp, tuy nhiên sẽ không chứng minh ở đây). Bởi vậy, chúng ta chỉ quan tâm tới các ma trận K khả nghich.

Tính khả nghịch của một ma trận vuông phụ thuộc vào giá trị định thức của nó. Để tránh sự tổng quát hoá không cần thiết, ta chỉ giới hạn trong trường hợp 2×2.

***Định nghĩa 1.4***

*Định thức của ma trận A = (a,i j ) cấp 2× 2 là giá trị*

*det A = a1,1 a2,2 - a1,2 a2,1*

*Nhận xét:* Định thức của một ma trận vuông cấp mm có thể được tính theo các phép toán hằng sơ cấp: hãy xem một giáo trình bất kỳ về đại số tuyến tính.

Hai tính chất quan trọng của định thức là det Im = 1 và quy tắc nhân det(AB) = det A × det B.

Một ma trận thức K là có nghịch đảo khi và chỉ khi định thức của nó khác 0. Tuy nhiên, điều quan trọng cần nhớ là ta đang làm việc trên Z26 . Kết quả tương ứng là ma trận K có nghịch đảo theo modulo 26 khi và chỉ khi UCLN(det K,26) = 1.

Sau đây sẽ chứng minh ngắn gọn kết quả này.

Trước tiên, giả sử rằng UCLN(det K,26) = 1. Khi đó det K có nghịch đảo trong Z26 . Với 1 ≤ i ≤ m, 1 ≤ j ≤ m, định nghĩa Ki j ma trận thu được từ K bằng cách loại bỏ hàng thứ i và cột thứ j. Và định nghĩa ma trận K\* có phần tử (i,j) của nó nhận giá trị(-1) det Kj i (K\* được gọi là ma trận bù đại số của K). Khi đó có thể chứng tỏ rằng:

K-1 = (det K)-1K\* .

Bởi vậy K là khả nghịch.

Ngược lại K có nghịch đảo K-1 . Theo quy tắc nhân của định thức

1 = det I = det (KK-1) = det K det K-1

Bởi vậy det K có nghịch đảo trong Z26 .

Nhận xét: Công thức đối với ở trên không phải là một công thức tính toán có hiệu quả trừ các trường hợp m nhỏ ( chẳng hạn m = 2, 3). Vớim lớn, phương pháp thích hợp để tính các ma trận nghịch đảo phải dựa vào các phép toán hằng sơ cấp.

Trong trường hợp 2×2, ta có công thức sau:

***Định lý 1.3***

*Giả sử A = (ai j) là một ma trận cấp 2 × 2 trên Z26 sao cho det A = a1,1a2,2 - a1,2 a2,1 có nghịch đảo. Khi đó*

A-1 = (det A)-1

a2,2 -a1,2

-a2,1 a1,1

Trở lại ví dụ đã xét ở trên . Trước hết ta có:

det

1. 8

3 7

= 11 × 7 - 8 ×3 mod 2

= 77 - 24 mod 26 = 53 mod 26

= 1

Vì 1-1 mod 26 = 1 nên ma trận nghịch đảo là

1. 8

3 7

-1

=

1. 18

23 11

Đây chính là ma trận đã có ở trên.

Bây giờ ta sẽ mô tả chính xác mật mã Hill trên Z26 (hình 1.6)

***Hình 1.4 Mật mã HILL***

Cho m là một số nguyên dương có định. Cho *P = C =* (Z26 )m và cho

*K*  = { các ma trận khả nghịch cấp m × m trên Z26}

Với một khoá K ∈*K* ta xác định

eK(x) = xK

và dK(y) = yK -1

Tất cả các phép toán được thực hiện trong Z26

**CHƯƠNG 2. BÀI TOÁN**

***2.1.Mã Affine***

***#include <stdio.h>***

***#include <string.h>***

***int main(int argc, char \*argv[]) {***

***char banRo[127], banMa[127];***

***int i;***

***fflush(stdin);***

***printf("Nhap ban ro: "); gets(banRo);***

***int a= 3, b= 6;***

***for(i=0; i<strlen(banRo); i++)***

***banMa[i]=( ( (a \* banRo[i]) + b) % 256);***

***banMa[strlen(banRo)] = 0;***

***printf("Ban ma: %s\n", banMa);***

***int a\_inv = 0;***

***int flag = 0;***

***for (int j = 0; j < 256; j++)***

***{***

***flag = (a \* j) % 256;***

***if (flag == 1)***

***{***

***a\_inv = j;***

***}***

***}***

***for(i=0; i<strlen(banMa); i++)***

***banRo[i]=( ( a\_inv \* (banMa[i] - b) ) % 256 );***

***banRo[strlen(banMa)] = 0;***

***printf("Giai ma, ban ro: %s\n", banRo);***

***return 0;***

***}***

***2.2Mã Vigenere***

***#include <stdio.h>***

***#include <string.h>***

***int main(int argc, char \*argv[]) {***

***char banRo[127], banMa[127];***

***char key[10];***

***fflush(stdin);***

***printf("Nhap ban ro: "); gets(banRo);***

***printf("Nhap khoa: "); gets(key);***

***for(int i=0,key\_L=strlen(key); i<strlen(banRo); i++)***

***banMa[i] = ((banRo[i] + key[i%key\_L]) % 256);***

***banMa[strlen(banRo)] = 0;***

***printf("Ban ma: %s\n", banMa);***

***for(int i=0,key\_L=strlen(key); i<strlen(banMa); i++)***

***banRo[i] = ((banMa[i] - key[i%key\_L]) % 256);***

***banRo[strlen(banMa)] = 0;***

***printf("Giai ma, ban ro: %s\n", banRo);***

***return 0;***

***}***

***2.3Mã Hill***

***#include<iostream>***

***#include<vector>***

***using namespace std;***

***int modInverse(int a, int m){***

***a=a%m;***

***for(int x=-m;x<m;x++)***

***if((a\*x)%m==1)***

***return x;***

***}***

***void getCofactor(vector<vector<int> > &a, vector<vector<int> > &temp, int p, int q, int n){***

***int i=0,j=0;***

***for(int row=0;row<n;row++){***

***for(int col=0;col<n;col++){***

***if(row!=p&&col!=q){***

***temp[i][j++] = a[row][col];***

***if (j==n-1){***

***j=0;***

***i++;***

***}***

***}***

***}***

***}***

***}***

***int determinant(vector<vector<int> > &a, int n, int N){***

***int D = 0;***

***if(n==1)***

***return a[0][0];***

***vector<vector<int> > temp(N, vector<int>(N));***

***int sign = 1;***

***for(int f=0;f<n;f++){***

***getCofactor(a, temp, 0, f, n);***

***D += sign \* a[0][f] \* determinant(temp, n - 1, N);***

***sign = -sign;***

***}***

***return D;***

***}***

***void adjoint(vector<vector<int> > &a,vector<vector<int> > &adj,int N){***

***if(N == 1){***

***adj[0][0] = 1;***

***return;***

***}***

***int sign = 1;***

***vector<vector<int> > temp(N, vector<int>(N));***

***for(int i=0;i<N;i++){***

***for(int j=0;j<N;j++){***

***getCofactor(a, temp, i, j, N);***

***sign = ((i+j)%2==0)? 1: -1;***

***adj[j][i] = (sign)\*(determinant(temp, N-1 , N));***

***}***

***}***

***}***

***bool inverse(vector<vector<int> > &a, vector<vector<int> > &inv, int N){***

***int det = determinant(a, N, N);***

***if(det == 0){***

***return false;***

***}***

***int invDet = modInverse(det,256);***

***vector<vector<int> > adj(N, vector<int>(N));***

***adjoint(a, adj, N);***

***for(int i=0;i<N;i++)***

***for(int j=0;j<N;j++)***

***inv[i][j] = (adj[i][j]\*invDet)%256;***

***return true;***

***}***

***int main(){***

***int x,y,i,j,k,n=2;***

***cout<<"Nhap ma tran co kich thuoc 2x2\n";***

***int b[n][n];***

***for(i=0;i<n;i++){***

***for(j=0;j<n;j++){***

***cin>>b[i][j];***

***}***

***}***

***cout<<"Nhap ban ro\n";***

***string banRo;***

***cin>>banRo;***

***int tp = (n-banRo.size()%n)%n;***

***for(i=0;i<tp;i++){***

***banRo+='x';***

***}***

***k=0;***

***string banMa="";***

***while(k<banRo.size()){***

***for(i=0;i<n;i++){***

***int sums = 0;***

***int temps = k;***

***for(j=0;j<n;j++){***

***sums += (b[i][j]%256\*(banRo[temps++]-'a')%256)%256;***

***sums = sums%256;***

***}***

***banMa+=(sums+'a');***

***}***

***k+=n;***

***}***

***cout<<"Ban ma: "<<banMa<<'\n';***

***cout<<"Nhap khoa giai ma\n";***

***vector<vector<int> > a(n, vector<int>(n));***

***vector<vector<int> > adj(n, vector<int>(n));***

***vector<vector<int> > inv(n, vector<int>(n));***

***for(i=0;i<n;i++){***

***for(j=0;j<n;j++){***

***cin>>a[i][j];***

***}***

***}***

***if(inverse(a,inv,n)){***

***}***

***k=0;***

***string Kq;***

***while(k<banMa.size()){***

***for(i=0;i<n;i++){***

***int sum = 0;***

***int temp = k;***

***for(j=0;j<n;j++){***

***sum += ((inv[i][j] + 256)%256\*(banMa[temp++]-'a')%256)%256;***

***sum = sum%256;***

***}***

***Kq+=(sum+'a');***

***}***

***k+=n;***

***}***

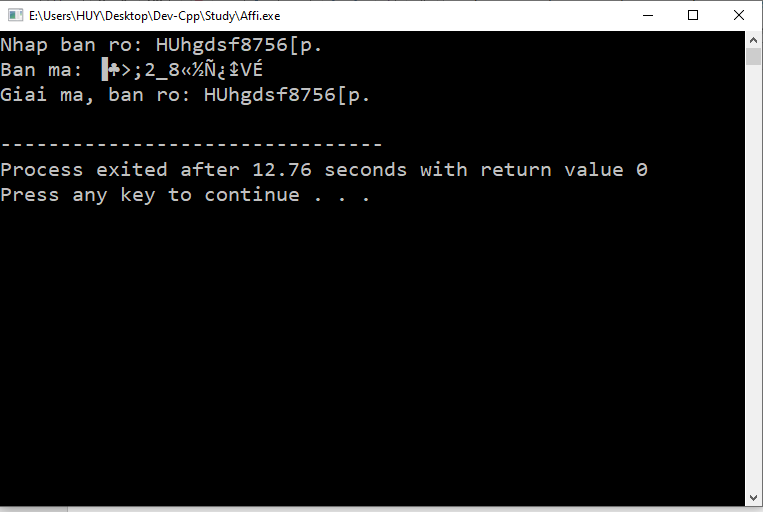
***cout<<"Giai ma, ban ro: "<<Kq;***

***return 0;***

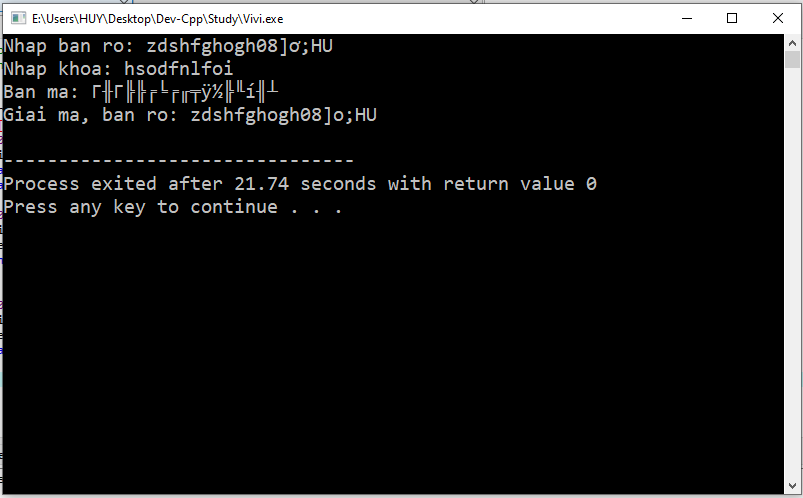
***}***

**CHƯƠNG 3. KẾT QUẢ THỬ NGHIỆM**

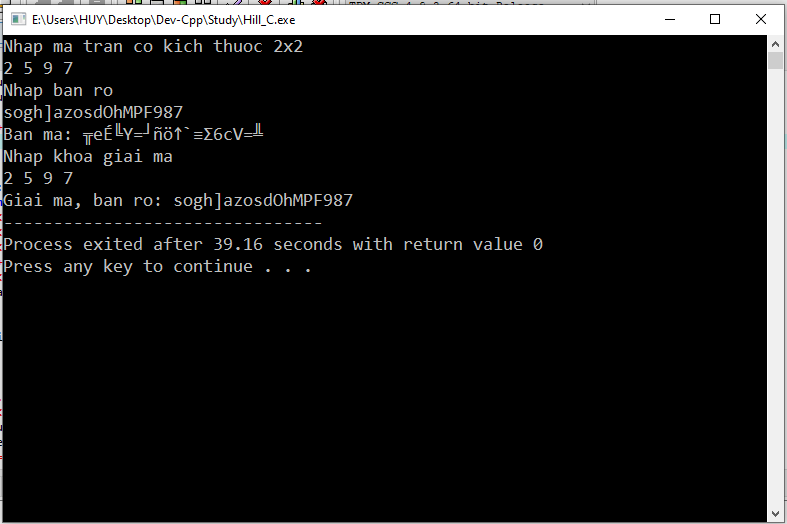
***3.1Mã Affine***



***3.2Mã Vigenere***



***3.3Mã Hill***



**KÊT QUẢ**

Còn một số hạn chế:

Gặp phải một số lỗi chưa khắc phục được như:

Mật mã Vigenere, còn sai khi gặp một vài kí tự đặc biệt như : ‘^’,...

Mật mã Hill, tại bản rõ khi nhập dấu “cách” hay “tab”,... sẽ không báo lỗi, hay khi khóa sai,giải mã sẽ tự động trả về các kí tự ‘a’ tương ứng mỗi kí tự trong bản rõ,...

Nhìn chung, bài tập đã hoàn thành được yêu cầu mã hóa và giải mã các mật mã bảng mã ASCII mod đồng dư 256.

**TÀI LIỆU THAM KHẢO**

Giáo trình An toàn bảo mật thông tin ĐHHHVN

Một số nguồn trên google