**Hình thức kiểm tra-thi**

1.Điểm quá trình:40%

* 1 bài kiểm tra giữa kỳ:80%
* chấm bài tập về nhà
* điểm chuyên cần

2.Thi cuối kỳ:60% ⇒ Thi viết 60p ⇒ Online vấn đáp

3.Giáo trình: Khoa học mật mã và An ninh mạng máy tính

**Chương 1:Giới thiệu chung**

# ***1-Các mối đe dọa***

| **Mục tiêu** | **Mối đe doạ** |
| --- | --- |
| Sự bí mật của dữ liệu ⇒ | Lộ bí mật |
| Sự toàn vẹn của dữ liệu ⇒ | Sai lệch, mất mát dữ liệu |
| Sự sẵn sàng của hệ thống ⇒ | Từ chối dịch vụ |

Vd: Hệ thống quá tả ⇒ k thể phục vụ kịp(tốt⇒ quá tải,từ chối dịch vụ)

## 1.1.Sự bí mật của dữ liệu:

* Chỉ phụ vụ cho 1 số ng cụ thể
* Phân quyền ng dùng(k đủ thầm quyền sẽ k đc truy cập dữ liệu)

## 1.2 Sự toàn vẹn dữ liệu:

* k đủ thẩm quyền k đc sửa đổi dữ liệu
* sửa đổi dl: thay đổi, xóa, di chuyển và thêm dữ liệu

## 1.3 sự sẵn sàng của hệ thống:

* Ht luôn sẵn sàng hoạt động,k ai làm ngừng hệ thống ngoài admin
* DoS(Denial of Service- Từ chối dịch vụ)- là hình thức tấn công nhằn đe dọa sự sẵn sàng của hệ thống.

# ***2.Những kẻ xâm nhập***

* Ng dùng tò mò,k chuyên môn,thích đọc trộm.
* Các chuyên gia kỹ thuật muốn thử thách bản thân (sinh viên, lập trình viên hệ thống, người vận hành…)
* Những kẻ tấn công vì tiền như nhân viên hoặc lập trình viên của ngân hàng
* Gián điệp do thám thông tin thương mại hoặc quân sự
* *Virus máy tính*

# ***3.Mất dữ liệu do sự cố***

# Tai nạn ngẫu nhiên: hoả hoạn, lũ lụt, động đất, chiến tranh, bạo loạn…

* Phần cứng hoặc phần mềm bị lỗi
* Lỗi do con người: Entry dữ liệu không chính xác, lắp nhầm băng hoặc đĩa, chạy nhầm chương trình, mất đĩa hoặc băng…

⇒ ⇒ Cần có 1 giải pháp tổng thể(gồm chiến lược và kĩ thuật)để đảm bảo atht

⇒ Giải pháp: Sao lưu(Cơ chế Real)

**Chương 2:Mật mã đối xứng**

* KN: MM đối xứng là mm có cùng khóa khi mã hóa và giải mã

# ***1-Những vấn đề cơ bản của mật mã***

* Mật mã là công cụ cơ bản để bảo vệ sự bí mật và toàn vẹn của dl

# Đặc điểm mật mã

* Có thể biến đổi dữ liệu từ dạng ban đầu -> dạng khó đọc hơn⇒ bảo vệ sự bí mật của dl
* Một số kỹ thuật mật mã=> CM nguồn gốc và sự toàn vẹn của dl
* Ứng dụng: giao thức mạng=> bảo vệ dl khi lưu thông trên mạng

## ⇒ Muốn khôi phục được chuỗi ban đầu thì cần phải biết K.

*- Trong kĩ thuật mật mã, K được gọi là “Khoá” (Key) - dùng để mã hoá và giải mã.*

*- Chuỗi ban đầu được gọi là “bản rõ” (Plain text)*

*- Chuỗi sau khi mã hoá được gọi là “bản mã” (Cipher text)*

*- Cách thức biến bản rõ thành bản mã được gọi là “thuật toán mã hoá”.*

*- Cách thức biến bản mã thành bản rõ được gọi là “thuật toán giải mã”.*

## ⇒ Quá trình mã hóa và giải mã

* Đầu vào :plaintext(P) đc đưa vào qt mã hóa(E:Hàm mã hóa) kết hợp khóa Key(encryption khóa dùng cho qt mã hóa)
* Đầu ra: hàm E là Bản mã (C:Cipher text) truyền từ nơi gửi đến đích(bên nhận)
* Lưu thông trên đg truyền là Bản mã C=E(P,Ke: ;à 2 tham số đầu vào)
* Khi nhận đc thì giải mã Hàm D kết hợp khóa giải mã Kd(khóa dùng trong qt giải mã) thu đc đầu ra là P=D(C,Kd) ban đầu

*Trong đó:*

*P* là bản rõ

*KE* là khoá mã hoá *(Encryption Key)*

*C* là bản mã

*E* là thuật toán mã hoá (hay hàm mã hoá)

*C* = *E*(*P*, *KE*) là công thức định nghĩa sự mã hoá. *Bản mã được tạo ra nhờ áp dụng thuật toán mã hoá E, tác động vào bản rõ P, với khoá mã hoá KE làm tham số.*

*KD* là khoá giải mã *(Decryption Key)*

*D* là thuật toán giải mã (hay hàm giải mã)

*P* = *D*(*C*, *KD*) là công thức định nghĩa sự giải mã. *Thuật toán D sẽ tác động lên bản mã C với KD là tham số, nó sẽ biến bản mã C trở về dạng ban đầu là bản rõ P*.

* Nếu *KD = KE* (tức là khoá giải mã và khoá mã hoá giống nhau), ta gọi đây là *“mật mã đối xứng”.* Khoá này phải được giữ bí mật.
* Ngược lại, nếu *KD ≠ KE*, đây là *“mật mã bất đối xứng”.*

# ***2-Các kỹ thuật mã hóa cổ điển***

## **2.1 Kỹ thuật thay thế là**

Thuật toán mã hoá sẽ thay thế mỗi kí tự trong bản rõ bằng một kí tự khác

CAESAR

### **2.1.1 Mật mã CAESAR(cũng là mm monoalphabetic)(1-1)**

KN:

### Mật mã CAESAR Mỗi kí tự trong bảng chữ cái được thay thế bởi một kí tự khác cùng bảng, cách sau nó ba vị trí.

### Brute-force là hình thức tấn công bằng cách thử tất cả các khả năng của khoá để tìm ra khoá đúng

\*Quy tắc thay thế(làm tay)

Plaintext:

**a b c d e f g h i j k l m n o p q r s t u v w x y z**

Ciphertext:

**D E F G H I J K L M N O P Q R S T U V W X Y Z A B C**

| Ưu điểm | Nhược điểm |
| --- | --- |
| -đơn giản,dễ thực hiện | -Độ an toàn k cao  -Dễ bị bẻ khóa bởi tấn công Brute-force do số lượng khóa quá ít(chỉ có 25 khóa) |

* mọi phép toán mã hoá và giải mã đều được thực hiện trong một tập hợp 26 số nguyên từ 0 đến 25 (gọi là “Vành 26” hay Z26).
* Công thức mã hoá:
* *C* = *E*(*P, 3*) = (*P* + 3) mod 26
* Công thức giải mã:
* *P* = *D*(*C, 3*) = (*C* – *3*) mod 26

VD:

***Mã hoá:***

Với *P(bản rõ)* = ‘A’ = 0 thì *C(bản ngã)* = (0 + 3) mod 26

*C* = 3 mod 26 = 3 = ‘D’

Với *P* = ‘X’(vành 26) = 23 thì *C* = (23+3) mod 26

*C* = 26 mod 26 = 0 = ‘A’

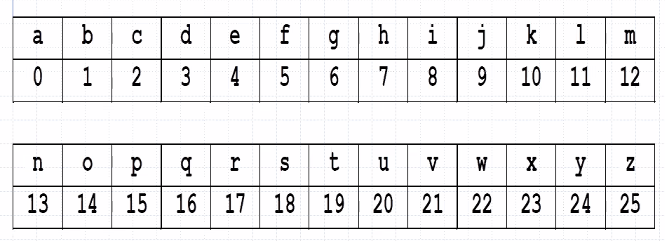
***Giải mã:***

Với *C* = ‘A’ = 0 thì *P* = (0 – 3) mod 26

*P* = -3 mod 26 = -3(k thuộc vành 26)

Chú ý: Đối với a thuộc Z26 thì **a + 26 = a**

*P* = -3 +26 = 23 = ‘X’



VD:

Plaintext:

meet me after the toga party

Ciphertext:

PHHW PH DIWHU WKH WRJD SDUWB

AFFINE

### **2.1.2 Mật mã Affine(cũng là mm monoalphabetic)**

KN: Kí tự *P* ban đầu được thay thế bởi kí tự *C* theo công thức:

*C* = *E*(*P, {a,b}*) = (*aP* + *b*) mod 26 ;(*aP* + *b*)==P=(c-b)/a

Công thức giải mã:

*P* = *D*(*C, {a,b}*) = *a-1*(*C* – *b*) mod 26⇒ (a+1)(C-b)%26

Trong đó khoá *K* chính là cặp 2 số nguyên *{a,b} thuộc Z26*

*Nếu a =1 thì mật mã Affine sẽ trở thành mật mã Caesar tổng quát*

*Điều kiện:*

Để có thể giải mã được thì a phải nguyên tố với 26, tức là ước số chung lớn nhất của a và 26 bằng 1: ƯSCLN(a, 26) = 1

Vì chỉ khi ƯSCLN(a, 26) = 1 thì mới tồn tại a-1Î Z26 để tính P

a-1 là một số thuộc Z26 thoả mãn:

a.a-1 = a-1.a = 1 (trong Z26)

| Ưu điểm | Nhược điểm |
| --- | --- |
| -Độ phước tạp lớn hơn mật mã ceasar tổng quát  -Số lương khóa nhiều hơn | -Độ an toàn chưa cao  -Dễ bị phá bởi tấn công Brute-force do slg khóa chưa nhiều(chỉ có 312 khóa) |

**Bài tập 5:**

Xác định các giá trị có thể có của a trong Z26, và tính a-1 tương ứng

Các giá trị của a:

1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25

Các a-1 tương ứng:

1, 9, 21, 15, 3, 19, 7, 23, 11, 5, 17, 25

Bẻ khoá mật mã Affine

Có tất cả bao nhiêu cặp số nguyên *{a,b}* trong Z26 có thể dùng làm khoá của mật mã Affine?

Có 12 giá trị của a

Có 26 giá trị của b

Tổng cộng có 12 x 26 = 312 cặp số {a,b}

MONOAPHABETIC

### **2.1.3 Mật mã Monoalphabetic(tổng quát)(1-1)**

| KN | Ưu điểm | Nhược điểm | Bẻ khóa |
| --- | --- | --- | --- |
| -Mỗi kí tự trong bảng chữ cái được thay thế bởi một kí tự bất kì khác cùng bảng  -Trong ví dụ sau, các kí tự *A* được thay bằng *Q*, *B* được thay bằng *W*, *C* được thay bằng *E*…  Bản rõ: ABCDEFGHIJKLMNOPQRSTUVWXYZ  Bản mã: QWERTYUIOPASDFGHJKLZXCVBNM | -Có số lượng khóa rất lớn,khó bẻ khóa bằng phương pháp Brute-force | -Vẫn có thể bẻ khóa mật mã này dựa trên các thống kê về các đặc điểm tự nhiên ngôn ngữ  -không làm thay đổi số lần xuất hiện trong bản mã(đoán số lượng xuất hiện kí tự được) | Số lượng hoán vị của chuỗi dài 26 chữ cái là: 26! ≈ 4.1026 (trên 4.1026 khoá!)  Giả sử một máy tính cá nhân tốc độ 4 GHz (thực hiện 4 tỷ phép tính/1 giây) có thể kiểm tra được 1 khoá trong 1 phép tính.  Để kiểm tra hết 4.1026 khoá sẽ cần tới: |

* Mật mã Monoalphabetic dễ bẻ vì không che giấu được tần xuất suất hiện của các kí tự trong văn bản

⇒ Để khắc phục có thể áp dụng mã hoá đa kí tự, hoặc sử dụng nhiều bảng mã thay thế (Polyalphabetic)…

### **2.1.4 Mã hóa đa kí tự(N-N)**

PLAYFAIR

#### 2.1.4.1.Mật mã Playfair

| khái niệm | -Mật mã Playfair là một hệ mã hóa nhiều chữ, giảm bớt tương quan giữa văn bản mã hóa và nguyên bản bằng cách mã hóa đồng thời nhiều [chữ cái](https://vi.wikipedia.org/wiki/Ch%E1%BB%AF_c%C3%A1i) của nguyên bản. |  |
| --- | --- | --- |
| Cơ chế hoạt động | -Sử dụng một ma trận chữ cái 5x5 trên cơ sở một từ khóa: điền các chữ cái của từ khóa (bỏ các chữ trùng), điền những vị trí còn lại của ma trận với các chữ cái khác của bảng chữ cái; I, J có thể ở trên cùng một ô của ma trận. |  |
| Cách giải mã bằng tay | Ví dụ ma trận với từ khóa MONARCHY  M O N A R  C H Y B D  E F G I/J K  L P Q S T  U V W X Z  • Mã hóa 2 chữ cái một lúc  – Nếu 2 chữ giống nhau, tách ra bởi 1 chữ điền thêm thường là X hoặc Q  Ví dụ: EE sẽ dược thay bởi EX  – Nếu 2 chữ nằm cùng hàng, thay bởi các chữ bên phải  Ví dụ: EF sẽ thay bằng FG  – Nếu 2 chữ nằm cùng cột, thay bởi các chữ bên dưới  Ví dụ: OF thay bằng HP  – Các trường hợp khác(chéo), mỗi chữ cái được thay bởi chữ cái khác cùng hàng, trên cột chữ cái cùng cặp  Ví dụ: ET sẽ thay bằng KL | VD:plaintext:DAIHOCTHUYLOI  B1:tách  DA IH OC TH UY LO IX  BR FB MH PD WC PM SA  VD2:  plaintext:VIETNAMCOLEN  khóa: MINH  M I N H A  B C D E F  G K L O P  Q R S T U  V W X Y Z  VI ET NA MC OL EN  WM OY HM IB PO DH  cipher text:WMOYHMIBPODH |
| Ưu điểm | - Ko gian khóa lớn(25!) ít hơn mm mono(312 khoá)=> khó bẻ khóa đc bằng pp brute-force  - Có khả năng che giấu 1 phần thông tin về tần số xuất hiện các chữ cái,nhờ thực hiện mã hóa từng cạp hai kí tự |  |
| Nhược điểm |  |  |

HILL

#### 2.1.4.2 .Mật mã Hill

| Khái niệm | Mật mã Hill sẽ thay thế từng nhóm *m* kí tự trong plaintext bởi *m* kí tự ciphertext  *m* kí tự ciphertext được xác định bởi hệ *m* phương trình tuyến tính sau: |  |
| --- | --- | --- |
|  | **Mã hóa: C** = **KP** mod 26  *Trong đó:*  C là ma trận cột biểu diễn ciphertext  P là ma trận cột biểu diễn plaintext  K là ma trận khoá  **Giải mã** *:***P** = **K-1C** mod 26  K-1 là ma trận nghịch đảo của ma trận khoá K  tức là K .K-1 =  K-1.K = I (I là ma trận đơn vị) | *C*1 = (*k*11*P*1 + k12*P*2 +…+ *k*1m*P*m) mod 26  *C*2 = (*k*21*P*1 + k22*P*2 + …+ *k*2m*P*m) mod 26  *C*3 = (*k*31*P*1 + k32*P*2 + …+ *k*3m*P*m) mod 26 …  *C*m = (*k*m1*P*1 + km2*P*2 + …+ *k*mm*P*m) mod 26 |
| Nhận xét | Độ an toàn sẽ càng lớn khi sử dụng ma trân K càng lớn |  |
| ưu điểm | -Có khả năng che dấu hoàn toàn tần suất xuất hiện các kí tự đơn  -Rất mạnh khi chống lại tấn công chỉ biết ciphertext  -đc sử dụng nhiều hơn |  |
| Nhc điểm | -Khối lượng tính toán lớn  -nhưng nó lại dễ dàng bị bẻ gãy với một tấn công biết plaintext, do có thể dễ dàng xác định ma trận K từ các cặp P-C đã biết. |  |

POLYAPHABETIC

### **2.1.5 Mật mã Polyaphabetic(1-nhiều)**

**So sánh:Monoa và Polya**

| Mật mã Monoalphabetic(1-1) | Mật mã Polyalphabetic(1-N) |
| --- | --- |
| Mật mã Monoalphabetic chỉ sử dụng một bảng mã (mỗi kí tự plain text được thay thế bởi một kí tự cố định), nên không giấu được tần suất xuất hiện các kí tự | Còn mật mã Polyalphabetic lại sử dụng nhiều bảng mã khác nhau (mỗi kí tự plain text có thể được thay thế bởi nhiều kí tự khác nhau, dựa trên các khoá thay thế khác nhau) |

VIGENERE

#### 2.1.5.1 .Mật mã ***Vigenère giống CAESER***

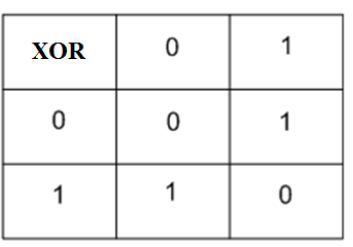
| Khái niệm | -Thay thế từng nhóm m kí tự trong plaintext bởi m ksi tự ciphertext  -m kí tự ciphertext đc xác định  *C*1 = (*P*1 + *k*1) mod 26  *C*2 = (*P*2 + *k*2) mod 26  *C*3 = (*P*3 + *k*3) mod 26…  *C*m = (*P*m + *k*m) mod 26  Trong đó  *Ci* là các kí tự ciphertext, *Pi* là các kí tự plaintext,  *ki* là các giá trị của khoá  Giải mã:  *P*1 = (*C*1 - *k*1) mod 26  *P*2 = (*C*2 - *k*2) mod 26  *P*3 = (*C*3 - *k*3) mod 26 …  *P*m = (*C*m - *k*m) mod 26 | VD:  Giả sử lấy *m=7* và khoá *K* là PHAMNGA, plaintext là:  NICETOOMEETYOU  Hãy xác định ciphertext.  - Do m=7, ta sẽ tách plaintext thành từng nhóm 7 kí tự:  NICETOO/MEETYOU  Viết theo dạng số là:  13 8 2 4 19 14 14 / 12 4 4 19 24 14 20  Từ khoá CIPHER tương ứng với:  *K* = (15,7,0,12,13,6,0)  Cộng từng nhóm 7 kí tự của plaintext với *K* ta có:  13 8 2 4 19 14 14 / 12 4 4 19 24 14 20  15 7 0 12 13 6 0 / 15 7 0 12 13 6 0  28/26 15 2 18 31 20 14 / 27 11 4 31 37 20 20  2 5 1 5 11  C P C S F U O / B L E F L U U  Ciphertext tương ứng là: CPCSFUOBLEFLUU |
| --- | --- | --- |
| Nhận xét | -Như vậy, nếu *k1= k2 = … = km* thì mật mã Vigenère sẽ trở thành mật mã Caesar tổng quát.  -Khi *k1≠ k2 ≠ … ≠ km*: một kí tự plaintext có thể được thay thế bởi nhiều kí tự khác nhau (ứng với các *k* khác nhau), nhờ vậy có thể che giấu được tần suất xuất hiện các kí tự. |  |
| Nhược | -Mật mã Vigenère vẫn không giấu được hoàn toàn tần suất xuất hiện các kí tự, và vẫn có thể bị phân tích  Giải pháp khắc phục là sử dụng hệ thống biểu diễn thông tin không mang tính thống kê của ngôn ngữ - đó là hệ nhị phân |  |

VERMAN

#### 2.1.5.2 .Mật mã Verman

| Khái niệm | -Plaintext đc biểu diễn dưới dạng chuỗi bit nhị phân  -Khoá *K* cũng được biểu diễn dưới dạng một chuỗi bít nhị phân (càng dài càng tốt, càng ngẫu nhiên càng tốt)  -Ciphertext được sinh ra bởi phép XOR giữa plaintext với khóa *K* | Công thức mã hoá:    Công thức giải mã:    ***Trong đó:***  *pi* = bit thứ *i* của plaintext  *ki* = bit thứ *i* của mật khóa *K*  là phép XOR |
| --- | --- | --- |
| Nhận xét | -K dài và ngẫu nhiên tốt đòi hỏi nhiều công sưc=>che giấu tốt  -K ngắn lặp lại K=>dễ lộ phần thống kê tần số  -mật mã hệ nhị phân(mm cổ đại) là tiền đề sự ra đời mật mã hiện đại |  |
| Ưu điểm |  |  |
| Nhược điểm | -K ngắn(8 bit) quá khi mã hóa k che dấu được tần xuất xuất hiện kí tự(dễ bẻ khóa bằng tay) |  |

#### 



VD: Giả sử plaintext là “HA NOI” (các kí tự trong chuỗi có thể được biểu diễn dưới dạng nhị phân theo bảng mã chuẩn ASCII), khóa K là một dãy nhị phân 8 bít như sau:

K = 10010011

Hãy mã hóa chuỗi ban đầu bằng phương pháp Vernman.

Giải mã chuỗi thu được rồi so sánh với chuỗi ban đầu

HA NOI 01001000 01000001 00100000 01001110 01001111 01001001

KHOA K=10010011 10010011 10010011 10010011 10010011 10010011

C= 11011011 11010010 10110011 11011101 11011100 11011010

C= 11011011 11010010 10110011 11011101 11011100 11011010

KHOA K= 10010011 10010011 10010011 10010011 10010011 10010011

HANOI 01001000 01000001 00100000 01001110 01001111 01001001

Giả sử plaintext là “VIET NAM” (các kí tự trong chuỗi có thể được biểu diễn dưới dạng nhị phân theo bảng mã chuẩn ASCII), khóa K là một dãy nhị phân 8 bít như sau:

K = 01001010

01010110 01001001 01000101 01010100 00100000 01001110 01000001 01001101

01001010 01001010 01001010 01001010 01001010 01001010 01001010 01001010

00011100 00000011 00001111 00011110 01101010 00000100 00001011 00000111

## **2.2 Kỹ thuật chuyển dịch - hoán vị**

Ngoài Kỹ thuật thay thế, các mật mã cổ điển còn sử dụng Kỹ thuật chuyển dịch - hoán vị

Các kí tự của plaintext sẽ được hoán đổi vị trí cho nhau để tạo thành ciphertext

RAIN-FENCE & HOÁN VỊ

### **2.2.1 Kỹ thuật rain-fence**

| Khái niệm | -Plaintext được viết dịch xuống tuần tự theo các đường chéo rồi đọc trình tự theo các hàng.  **ricrac VD1G**iả sử plaintext là *"meet me after the toga party"*, cách viết như sau:  **m e m a t r h t g p r y**  **e t e f e t e o a a t**  **Ciphertext thu được là:**  **MEMATRHTGPRYETEFETEOAAT** | VD2:**hoán đổi hàng cột ma trận**Viết plaintext trong một hình chữ nhật theo từng hàng, rồi đọc thông điệp theo từng cột, nhưng hoán đổi trật tự cột.Trật tự của các cột sẽ trở thànhkhóa cho thuật toán  Giả sử plaintext là *"attack postponed until two am"*, cách viết như sau:  Key: *4 3 1 2 5 6 7*  Plaintext:  a t t a c k p  o s t p o n e  d u n t i l t  w o a m x y z  Ciphertext: TTNAAPTMTSUOAODWCOIXKNLYPETZ |
| --- | --- | --- |
| Nhận xét | Mật mã hoán vị thuần túy rất dễ nhận ra bởi nó giữ nguyên tần suất xuất hiện ký tự đơn (và làm thay đổi tần suất của các cặp, các bộ kí tự của plaintext)  Để tăng độ phức tạp, người ta có thể tiến hành đổi chỗ nhiều lần, hoặc kết hợp với các thuật toán mã hoá khác. |  |
|  |  |  |

**Bài tập 20: So sánh mm cổ điển(đối xứng) và mm hiện đại(Bất đối xứng)**

* Mật mã cổ điển: dùng 2 kĩ thuật thay thế và hoán vị nhưng đối tượng là…,dùng 1 khóa cho mã hóa và giải mã
* MM hiện đại: dùng 2 kĩ thuật thay thế và hoán vị nhưng đối tượng là..,khóa cho mã hóa và giải mã khác nhau

| mm đối xứng | mm bất đối xứng |
| --- | --- |
| -dùng chung 1 khóa cho quá trình mã hóa và giải mã  -dựa trên nền tảng của phép thay thế và hoán vị | -sử dụng 2 khóa riêng biệt cho quá trình mã hóa và giải mã  -dựa trêm nền tảng của các hàm toán học là chủ yếu  ⇒ Có khả năng chống lại tất cả các tấn công đã được biết đến.  Đảm bảo tốc độ thực hiện trên nhiều nền tảng kiến trúc.  Có thiết kế đơn giản |

# ***3-Chuẩn mã hóa dữ liệu DES***

## ***3.1 Mật mã khối và DES***

| MM luồng(stream cipher) | MM khối(Block cipher) |
| --- | --- |
| là loại mật mã lần lượt mã hóa từng bit hay từng byte dữ liệu số của một luồng dữ liệu. | mã hóa mỗi lần cả một khối plaintext và thường kết xuất ra một khối ciphertext có chiều dài như khối plaintext.  - Độ dài 1 khối thường 64 bit hay 128 bit |

## ***3.2 Mật mã khối Feistel***

| KN | -là cơ sở của mm khối hiện đại  -Đầu vào: 1 khối plaintext dài m bit và khóa k dài k bit  -Quá trình: tác động lên plaintext bằng các kỹ thuật thay thế và hoán vị, lặp lại nhiều lần  -Đầu ra: để thu được ciphertext ở đầu ra cũng có chiều dài *m* bít. | -Đầu vào: 1 khối plaintext dài m bit và khóa k dài k bit  -Quá trình: tác động lên plaintext bằng các kỹ thuật thay thế và hoán vị, lặp lại nhiều lần  -Đầu ra: để thu được ciphertext ở đầu ra cũng có chiều dài *m* bít. |
| --- | --- | --- |
|  | -Kỹ thuật thay thế có tác dụng làm *xáo trộn* thông tin thống kê của plaintext, làm phức tạp hóa mối quan hệ thống kê giữa ciphertext và mật khoá, nhằm ngăn cản nỗ lực tìm khoá.  -Kỹ thuật hoán vị có tác dụng làm *khuếch tán* thông tin thống kê của plaintext, pha loãng các cấu trúc thống kê của plaintext ra một phạm vi rộng hơn, làm phức tạp hóa mối quan hệ thống kê giữa ciphertext và plaintext.  ⇒ Có thể lặp lại phép thay thế và hoán vị nhiều lần để tăng độ phức tạp, nâng cao tính bảo mật  Hoạt động Feistel:  Plaintext có chiều dài *m* = 2*w* bit, được chia thành hai nửa, *L*0 và *R*0.  Hai nửa dữ liệu đi qua *n* vòng xử lý rồi được tổ hợp lại thành khối ciphertext.  Mỗi vòng xử lý *i* có các đầu vào *L*i-1 và *R*i-1 lấy từ vòng trước cùng với khóa con *K*i, lấy từ khóa *K* gốc.  Các khóa con *K*i đều khác nhau và khác khóa *K*.  Cấu trúc của 1 vòng xử lý  Đầu tiên, **phép thay thế** được thực hiện trên L0, bằng cách XOR L0 với F(R0,K1).  Hàm F là một phép biến đổi, càng phức tạp càng tốt  Sau đó thực hiện **phép hoán vị** bằng cách hoán đổi giá trị của hai nửa dữ liệu  Kết quả thu được: R1 = L0 F(R0,K1)  L1 = R0  Lấy L1, R1 làm đầu vào của vòng thứ 2, lặp lại *n* vòng.  Công thức tổng quát của mỗi vòng:  Ri = Li-1 F(Ri-1,Ki)  Li = Ri-1  **Kích thước khối (*m*)**: Kích thước khối càng lớn càng bảo mật tốt, nhưng làm giảm tốc độ của thuật toán. Thường thì *m = 64* bít.  **Kích thước khóa (*k*)**: Kích thước khóa càng lớn càng bảo mật tốt, nhưng làm giảm tốc độ của thuật toán. Trước đây DES sử dụng *k=56* bít, nhưng hiện không còn an toàn nữa. Ngày nay *k = 128* bít là phổ biến.  **Số vòng xử lý (*n*)**: Một vòng đơn thì tính bảo mật không cao, nhưng nhiều vòng kết hợp sẽ làm tăng khả năng an ninh. Số vòng thường là 16.  **Thuật toán sinh khóa con**: Thuật toán càng phức tạp thì càng khó bị phân tích phá mã.  **Hàm F**: Hàm càng phức tạp thì càng khó bị phân tích phá mã. | Thời gian trung bình để dò tìm khóa theo thuật toán brute-force |
|  |  |  |

**So sánh DES và AES**

|  | Chuẩn mã hóa dl DES  (Data Encryption standard)-1977 | Chuẩn mã hóa cải tiến AES  (*Advanced Encryption Standard*) |
| --- | --- | --- |
|  | -DES sử dụng cấu trúc Feistel với độ dài khối *m = 64* bít, và độ dài khóa *k = 56* bít  -DES được sử dụng rộng rãi trong một thời gian dài (hơn 20 năm), cho tới khi nó bị bẻ gãy bởi tấn công brute-force năm 1998. | -Chuẩn mã hoá cải tiến AES (*Advanced Encryption Standard*) được ban hành bởi NIST (*National Institute of Standards and Technology* ) vào năm 2001.  -AES là mật mã khối để thay thế cho DES trong các ứng dụng thương mại. Nó sử dụng kích thước khối 128 bit và kích thước mật khoá 128, 192 hay 256 bit |
|  |  | *AES không mang cấu trúc Feistel. Mỗi vòng của nó chứa bốn hàm phân biệt:*  **Thay thế byte**: Thực hiện một phép thay thế từng byte cho khối.  **Dịch chuyển hàng**: Chỉ là một hoán vị đơn giản.  **Trộn cột**: Một phép thay thế sử dụng số học  **Thêm khoá vòng**: Một phép XOR bít đơn giản của khối hiện tại với một phần của khoá mở rộng. |
| Đặc trưng |  | -Có khả năng chống lại tất cả các tấn công đã được biết đến.  -Đảm bảo tốc độ thực hiện trên nhiều nền tảng kiến trúc.  -Có thiết kế đơn giản  ⇒ đc sd phổ biến nhất hiện nay |

-----------------------

Thi giữa kỳ(T2-16/08)

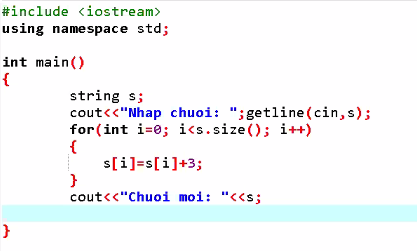
* Thi 50p
* thi viết
* Bài làm:2 câu:lập trình/mã hoặc hoặc giải mã bằng tay))
* gửi bài: làm vào giấy có 10p để chụp và nộp qua email:binhgl@yahoo.com
* tiêu đề email và tên file :50 PhamThiPhuongNga Lop60TH1
* ND thi: Mật mã cổ điển

**Bài tập lập trình**

----------------------

***Bài tập 1:cho K***

- *Lập trình nhập một chuỗi kí tự từ bàn phím, cộng mỗi phần tử của chuỗi với 3. Hiện chuỗi mới ra màn hình.*

**

#include <iostream>

using namespace std;

int main()

{

string s;

cout<<"Nhap chuoi: ";getline(cin,s);

for(int i=0; i<s.size(); i++)

{

s[i]=s[i]+3;

}

cout<<"Chuoi moi: "<<s;

}

- *Lập trình khôi phục lại chuỗi ban đầu.*

#include <iostream>

using namespace std;

int main()

{

string s;

cout<<"Nhap chuoi: ";getline(cin,s);

for(int i=0; i<s.size(); i++)

{

s[i]=s[i]+3;

}

cout<<"Chuoi ma hoa: "<<s<<endl;

for(int i=0; i<s.size(); i++)

{

s[i]=s[i]-3;

}

cout<<"Chuoi giai ma: "<<s;

}

**Bài tập 2:K từ bàn phím**

- *Lập trình nhập một chuỗi kí tự từ bàn phím, cộng mỗi phần tử của chuỗi với K (K là một giá trị nhập từ bàn phím). Hiện chuỗi mới ra màn hình.*

- *Lập trình khôi phục lại chuỗi ban đầu.*

#include <iostream>

using namespace std;

int main()

{

string s;

int K;

cout<<"Nhap chuoi: ";getline(cin,s);

cout<<"Nhap K= "; cin>>K;

for(int i=0; i<s.size(); i++)

{

s[i]=s[i]+K;

}

cout<<"Chuoi ma hoa: "<<s<<endl;

for(int i=0; i<s.size(); i++)

{

s[i]=s[i]-K;

}

cout<<"Chuoi giai ma: "<<s;

}

**Bài tập 3:Caesal Mã hóa/Giải mã**

- *Lập trình nhập một chuỗi kí tự từ bàn phím, mã hoá chuỗi bằng thuật toán CAESAR tổng quát với khoá K nhập từ bàn phím. Hiện chuỗi mới ra màn hình.*

- *Lập trình giải mã để khôi phục lại chuỗi ban đầu.*

#include <iostream>

using namespace std;

int kt\_so(char c)

{return c- 'A';} // doi ki tu thanh so nguyen moi

char so\_kt(int n)

{return 'A' +n;} // doi so nguyen da ket hop khoa thanh ki tu

int main()

{

string P,C;

int K;

cout<<"Nhap chuoi P= ";getline(cin,P);

C=P;//gan do dai C = P NEU K GAN THI CHAY CT RA RONG

cout<<"Nhap khoa K= "; cin>>K;

for(int i=0; i<P.size(); i++)

{

int m;

m=kt\_so(P[i]);

m=(m+K)%26;

C[i] =so\_kt(m);

}

cout<<"Chuoi ma hoa: "<<C<<endl;

for(int i=0; i<C.size(); i++)

{

int m;

m=kt\_so(C[i]);

m=(m-K+26)%26; // CONG 26 de tranh phep tru am Chú ý: Đối với a thuộc Z26 thì **a + 26 = a**

P[i] =so\_kt(m);

}

cout<<"Chuoi giai ma: "<<P;

}

**Bài tập 4:Bẻ khóa/Tìm khóa**

*Lập trình bẻ khoá mật mã Caesar bằng phương pháp Brute-force.*

- *Đầu vào chương trình là chuỗi kí tự cipher text thu được từ Bài tập 1.*

*- Hãy xác định khoá K đã sử dụng và nội dung của plain text ban đầu.*

#include <iostream>

using namespace std;

int kt\_so(char c)

{return c- 'A';} // doi ki tu thanh so nguyen moi

char so\_kt(int n)

{return 'A' +n;} // doi so nguyen da ket hop khoa thanh ki tu

int main()

{

string P,C;

int K;

cout<<"Nhap chuoi C= ";getline(cin,C);

P=C;//gan do dai C = P NEU K GAN THI CHAY CT RA RONG

for(K=1;K<=25;K++)

{

for(int i=0; i<C.size(); i++)

{

int m;

m=kt\_so(C[i]);

m=(m-K+26)%26; // CONG 26 de tranh phep tru am

P[i] =so\_kt(m);

}

cout<<K<<": "<<P<<endl;

}

}

### 

**Bài tập 5:Affine**

Xác định các giá trị có thể có của a trong Z26, và tính a-1 tương ứng

Các giá trị của a:

1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25

Các a-1 tương ứng:

1, 9, 21, 15, 3, 19, 7, 23, 11, 5, 17, 25

Bẻ khoá mật mã Affine

Có tất cả bao nhiêu cặp số nguyên *{a,b}* trong Z26 có thể dùng làm khoá của mật mã Affine?

Có 12 giá trị của a

Có 26 giá trị của b

Tổng cộng có 12 x 26 = 312 cặp số {a,b}

**Bài tập 6:Affine Mã hóa/Giải mã**

- *Lập trình nhập một chuỗi kí tự từ bàn phím, mã hoá chuỗi bằng thuật toán Affine với cặp số {a,b} nhập từ bàn phím. Hiện chuỗi mới ra màn hình.*

#include <iostream>

using namespace std;

int kt\_so(char c)

{return c- 'A';} // doi ki tu thanh so nguyen moi

char so\_kt(int n)

{return 'A' +n;} // doi so nguyen da ket hop khoa thanh ki tu

int main()

{

string P,C;

int a,b;

cout<<"Nhap chuoi C= ";getline(cin,C);

P=C;//gan do dai C = P NEU K GAN THI CHAY CT RA RONG

while(a<1 || a>25 || a%2==0 || a==13)

{

cout<<"Nhap a= "; cin>>a;

}

cout<<"\nNhap b= ";cin>>b;

for(int i=0; i<P.size(); i++)

{

int m;

m=kt\_so(P[i]);

m=(a\*m+b)%26; // CONG 26 de tranh phep tru am

C[i] =so\_kt(m);

}

cout<<"Chuoi ma hoa: "<<C;

//giai ma

int a1;

for(int i=1; i<=25; i++)

if((a\*i)%26==1) {a1=i;break;}

for(int i=0; i<P.size(); i++)

{

int m;

m=kt\_so(C[i]);

m=a1\*(m-b+26)%26; // CONG 26 de tranh phep tru am

P[i] =so\_kt(m);

}

cout<<"\nChuoi giai ma: "<<P;

}

**Bài tập 7:Bẻ khóa/Tìm khóa**

*Lập trình bẻ khoá mật mã Affine bằng phương pháp Brute-force:*

- *Đầu vào chương trình là chuỗi kí tự cipher text thu được từ đã cho.*

*- Hãy xác định cặp số {a,b} đã sử dụng và nội dung của plain text ban đầu.*

#include <iostream>

using namespace std;

int kt\_so(char c)

{return c- 'A';} // doi ki tu thanh so nguyen moi

char so\_kt(int n)

{return 'A' +n;} // doi so nguyen da ket hop khoa thanh ki tu

int main()

{

string P,C;

int a=0,b;

cout<<"Nhap chuoi C= ";getline(cin,C);

P=C;//gan do dai C = P NEU K GAN THI CHAY CT RA RONG

//giai ma

for(a=1; a<=25 ; a=a+2)

if(a!=13)

{

int a1;

for(int i=1; i<=25; i++)

if((a\*i)%26==1) {a1=i;break;}

for(b=0;b<=25; b++)

{

for(int i=0; i<P.size(); i++)

{

int m;

m=kt\_so(C[i]);

m=a1\*(m-b+26)%26; // CONG 26 de tranh phep tru am

P[i] =so\_kt(m);

}

cout<<a<<","<<b<<": "<<P<<endl;

}

}

}

**Bài tập 8:Monoaphabetic Mã hóa/Giải mã**

*Lập trình nhập một chuỗi kí tự từ bàn phím, mã hoá chuỗi bằng thuật toán Monoalphabetic với khoá K nhập từ bàn phím (Khoá K là một chuỗi gồm 26 chữ cái có trật tự bất kì). Hiện chuỗi mới ra màn hình.*

#include <iostream>

using namespace std;

int main()

{

string P,C,K;

string B="ABCDEFGHIJKLMNOPQRSTUVWXZY";

cout<<"Nhap plaintext : ";cin>>P;

C=P;

cout<<"Nhap chuoi K = ";cin>>K;

//ma hoa

for(int i=0; i<P.size(); i++)

for(int j=0;j<26;j++)

if(P[i]==B[j]) C[i]=K[j];

cout<<"Chuoi ma hoa: "<<C<<endl;

//giai ma

for(int i=0; i<C.size(); i++)

for(int j=0;j<26;j++)

if(C[i]==K[j]) P[i]=B[j];

cout<<"Chuoi giai ma: "<<P;

}

**Bài tập 9: Đếm kí tự**

Lập trình nhập một chuỗi kí tự từ bàn phím. Đếm xem kí tự đầu tiên của chuỗi xuất hiện bao nhiêu lần trong chuỗi?

#include <iostream>

using namespace std;

int main()

{

string ch;

cout<<"Nhap chuoi ki tu ch: ";cin>>ch;

int dem=0;

for(int i=0;i<ch.size();i++)

{

if(ch[i]==ch[0]) dem=dem+1;

}

cout<<endl<<"So lan xuat hien cua ki tu dau trong chuoi la: "<<dem;

}

**Bài tập 10:Quan trọng) Tần xuất xh**

**Lập trình tính tần suất xuất hiện của các kí tự trong một đoạn văn bản cho trước(Quan trọng)**

#include <iostream>

using namespace std;

int main()

{

string ch;

cout<<"Nhap chuoi ki tu: ";cin>>ch;

for(int i=0;i<ch.size();i++)

{

int dem=0;

for(int j=0;j<ch.size();j++)

{

if(ch[i]==ch[j])

{

if(i<=j)

{

dem++;

}

else

{break;}

}

}

if(dem !=0)

{

cout<<endl<<"Tan so xuat hien cua cac ki tu trong chuoi la: "<<dem;

}

}

}

**Bài tập 11:Playfair**

xóa ký tự chuỗi trùng lặp

#include<iostream>

using namespace std;

int main()

{

string K;

cout<<"Nhap chuoi khoa: ";getline (cin,K);

//xoa trung lap

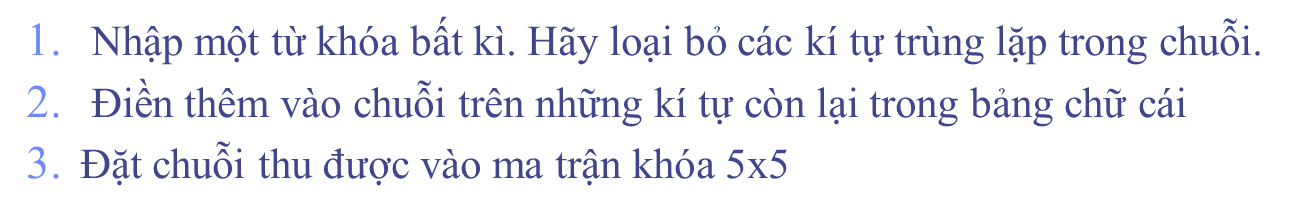
for(int i=0; i<K.size()-1; i++)

for(int j=i+1; j<K.size(); j++)

if(K[j]==K[i]) {K.erase(j,1);j--;}

cout<<"Chuoi vua xoa: "<<K;

}



#include<iostream>

using namespace std;

int main()

{

string K;

string B="ABCDEFGHIKLMNOPQRXTUVWSYZ";

cout<<"Nhap chuoi khoa: ";getline (cin,K);

K=K+B; // dien them vào chuoi nhung ki tu con lai trong bang chu cai

//xoa trung lap

for(int i=0; i<K.size()-1; i++)

for(int j=i+1; j<K.size(); j++)

if(K[j]==K[i]) {K.erase(j,1);j--;} //Nhap mot tu khóa b?t kì. Hãy lo?i b? các kí t? trùng l?p trong chu?i.

cout<<"Chuoi vua xoa: "<<K;

char A[5][5] ;int t=0;

for(int i=0 ; i<5; i++)

for(int j=0; j<5; j++)

{A[i][j]=K[t];t++;}

//Hien ma tran khoa

cout<<"\nMa tran khoa: "<<endl;

for(int i=0 ; i<5; i++)

{

for(int j=0; j<5; j++) cout<<A[i][j]<<" ";

cout<<endl;

}

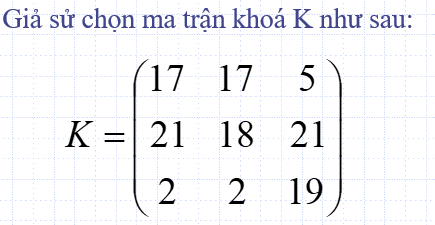
}

**Bài tập 12:VỀ NHÀ ?**

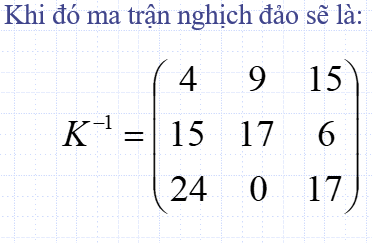
Lập trình mã hoá và giải mã thông điệp theo thuật toán Playfair

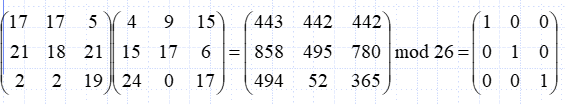
**Bài tập 13:Hill**

Với plaintext là *”paymoremoney”* và sử dụng với mật khóa *K* nói trên, hãy xác định ciphertext.

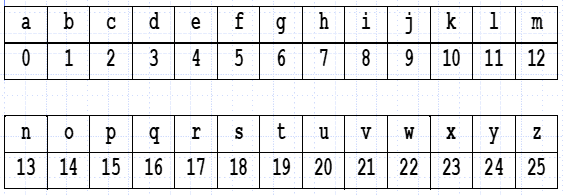


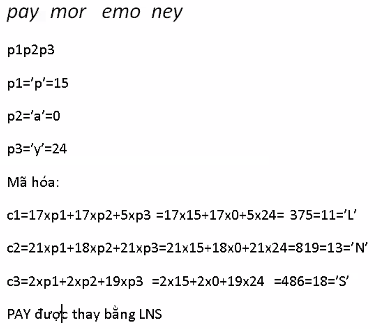
Giải mã ciphertext thu được bằng cách nhân nó với *K-1*, rồi so sánh kết quả với plaintext ban đầu





Giải:





p1=’m’=12

p2=’o’=14

p3=’r’=17

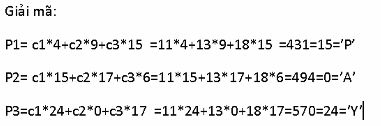
Mã hóa:

**C** = **KP** mod 26

c1=17xp1+17xp2+5xp3=17x12+17x14+5x17=204+238+85=

c2=21

**P** = **K-1C** mod 26



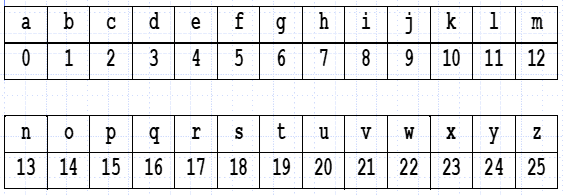
**Bài tập 14:VỀ NHÀ ?**

Lập trình mã hoá và giải mã thông điệp theo thuật toán Hill

**Bài tập 15:Vigenere**

Chọn một plaintext và một khoá K bất kì

Áp dụng thuật toán Vigenère để xác định ciphertext



Giả sử lấy *m=7* và khoá *K* là PHAMNGA, plaintext là:

NICETOOMEETYOU

Hãy xác định ciphertext.

- Do m=7, ta sẽ tách plaintext thành từng nhóm 7 kí tự:

NICETOO/MEETYOU

Viết theo dạng số là:

13 8 2 4 19 14 14 / 12 4 4 19 24 14 20

Từ khoá CIPHER tương ứng với:

*K* = (15,7,0,12,13,6,0)

Cộng từng nhóm 7 kí tự của plaintext với *K* ta có:

13 8 2 4 19 14 14 / 12 4 4 19 24 14 20

15 7 0 12 13 6 0 / 15 7 0 12 13 6 0

28/26 15 2 18 31 20 14 / 27 11 4 31 37 20 20

2 5 1 5 11

C P C S F U O / B L E F L U U

Ciphertext tương ứng là: CPCSFUOBLEFLUU

**Bài tập 16:**

*Lập trình nhập chuỗi kí tự plaintext từ bàn phím, mã hoá chuỗi bằng thuật toán* Vigenère *với khoá K (là một chuỗi kí tự) nhập từ bàn phím. Hiện chuỗi mới ra màn hình.*

*Lập trình giải mã để khôi phục lại chuỗi ban đầu.*

#include <iostream>

using namespace std;

int kt\_so(char c)

{return c- 'A';} // doi ki tu thanh so nguyen moi

char so\_kt(int n)

{return 'A' +n;} // doi so nguyen da ket hop khoa thanh ki tu

int main()

{

string P,C,K;

cout<<"Nhap chuoi Plaintext= ";getline(cin,P);

C=P;//gan do dai C = P NEU K GAN THI CHAY CT RA RONG

cout<<"Nhap khoa K= "; cin>>K;

//Ma hoa

int j=0;

for(int i=0; i<P.size(); i++)

{

int m; m=kt\_so(P[i]);

int k; k=kt\_so(K[j]);

m=(m+k)%26;

C[i] =so\_kt(m);

j=j+1;

if(j==K.size()) j=0;

}

cout<<"Chuoi ma hoa: "<<C<<endl;

//Giai ma

for(int i=0; i<C.size(); i++)

{

int m; m=kt\_so(C[i]);

int k; k=kt\_so(K[j]);

m=(m-k+26)%26;

P[i] =so\_kt(m);

j=j+1;

if(j==K.size()) j=0;

}

cout<<endl<<"Chuoi ma hoa: "<<P;

}

**Bài tập 17:Verman**

Nhập một chuỗi plaintext từ bàn phím

Nhập một khóa *K* (dài 8 bít)

Mã hóa chuỗi ban đầu bằng cách XOR các kí tự của nó với *K*

Giải mã ciphertext thu được, rồi so sánh với chuỗi ban đầu.

**K ngắn**

**#include <iostream>**

**using namespace std;**

**int main()**

**{**

**string P,C; char K;**

**cout<<"Nhap chuoi Plaintext= ";getline(cin,P);**

**C=P;//gan do dai C = P NEU K GAN THI CHAY CT RA RONG**

**cout<<"Nhap khoa K= "; cin>>K;**

**//Ma hoa**

**for(int i=0; i<P.size(); i++) C[i]=P[i]^K;**

**cout<<"Chuoi ma hoa: "<<C;**

**//Giai ma**

**for(int i=0; i<P.size(); i++) P[i]=C[i]^K;**

**cout<<endl<<"Chuoi giai ma: "<<P;**

**}**

**K dài**

**#include <iostream>**

**using namespace std;**

**int main()**

**{**

**string P,C,K;**

**cout<<"Nhap chuoi Plaintext= ";getline(cin,P);**

**C=P;**

**cout<<"Nhap khoa K= "; cin>>K;**

**//Ma hoa**

**int j=0;**

**for(int i=0; i<P.size(); i++)**

**{**

**C[i]=P[i]^K[j];**

**j++;**

**if(j==K.size()) j=0;**

**}**

**cout<<"Chuoi ma hoa: "<<C;**

**//Giai ma**

**for(int i=0; i<P.size(); i++)**

**{**

**P[i]=C[i]^K[j];**

**j++;**

**if(j==K.size()) j=0;**

**}**

**cout<<endl<<"Chuoi ma hoa: "<<P;**

**}**

**Bài tập 18:rain-fence**

Nhập một chuỗi plaintext

Mã hóa chuỗi bằng kỹ thuật rain-fence, hiện ciphertext ra màn hình

Giải mã ciphertext, so sánh kết quả với chuỗi ban đầu.

include <iostream>

using namespace std;

int main()

{

string P,C,K;

cout<<"Nhap chuoi Plaintext= ";getline(cin,P);

C=P;

//Ma hoa

string h1,h2;

for(int i=0; i<P.size(); i++)

{

if(i%2==0) h1=h1+P[i]; //hang 1 ki tu hang chan

else h2=h2+P[i]; //hang 2 ki tu hang le

}

C=h1+h2;

cout<<"Chuoi ma hoa: "<<C;

//Giai ma

P="";h1="";h2="";//reset ve rong vi o tren da khai bao va co gia tri

if(C.size()%2==0)

{

for(int i=0;i<C.size();i++)

{

if(i<C.size()/2) h1=h1+C[i];

else h2=h2+C[i];

}

}

else

{

for(int i=0;i<C.size();i++)

{

if(i<=C.size()/2) h1=h1+C[i];

else h2=h2+C[i];

}

}

for (int i=0;i<C.size()/2;i++) P=P+h1[i] +h2[i];

if(C.size()%2==1) P=P+h1[h1.size()-1];

cout<<endl<<"Chuoi giai ma: "<<P;

}

**Bài tập 19:Hoán đổi hàng cột matran**

-Nhập một chuỗi plaintext (dài không quá 25 kí tự)

-Sắp xếp các kí tự của chuỗi vào một ma trận 5 x 5 (lần lượt theo từng hàng)

-Đọc các phần tử của ma trận theo từng cột để tạo thành ciphertext

-Giải mã ciphertext, so sánh kết quả với chuỗi ban đầu.

#include <iostream>

using namespace std;

int main()

{

string P,C;char A[5][5];

cout<<"Nhap chuoi Plaintext= ";getline(cin,P);

//Ma hoa

string h1,h2;

int t=0;

for(int i=0; i<5; i++)

for(int j=0; j<5; j++)

{

A[i][j]=P[t];

t++;

if(t==P.size()) t=0;

}

for(int i=0;i<5;i++)

{

for(int j=0; j<5; j++) cout<<A[i][j]<<" ";

cout<<endl;

}

for(int i=0; i<5; i++)

for(int j=0; j<5; j++)

C=C+A[j][i];

cout<<"Chuoi ma hoa: "<<C<<endl;

//Giai ma

t=0;P="";

for(int i=0; i<5; i++)

for(int j=0; j<5; j++)

{

A[i][j]=C[t];

t++;

if(t==C.size()) t=0;

}

for(int i=0;i<5;i++) //Hien ma tran

{

for(int j=0; j<5; j++) cout<<A[i][j]<<" ";

cout<<endl;

}

for(int i=0; i<5; i++)

for(int j=0; j<5; j++)

P=P+A[j][i];

cout<<"Chuoi giai ma: "<<P;

}

**Bài tập 21:Feistel**

Lập trình mô phỏng hoạt động của mật mã Feistel đơn giản với 2 vòng xử lý:

Nhập chuỗi plaintext từ bàn phím, chia chuỗi thành các khối dài *m* = 2*w* = *16* bít.

Nhập khóa *K* (dài *8* bít) từ bàn phím. Khóa *Ki* được sinh ra từ khóa *K* nhờ phép dịch trái *K* i lần.

Hàm *F* thực hiện phép cộng giữa *Ri-1* với *Ki*.

Hiện chuỗi ciphertext ra màn hình.

#include <iostream>

using namespace std;

char F(char R,char Ki)

{

return R+Ki;

}

int main()

{

string P,C;char K[3],L[3],R[3];

cout<<"Nhap chuoi Plaintext= ";getline(cin,P); C=P;

cout<<"Nhap khoa K: ";cin>>K[0];

//Ma hoa

R[0]=P[0];

L[0]=P[1];

for(int i=1; i<=2; i++)

{

K[i]=K[0]<<i;

R[i]=L[i-1]^F(R[i-1],K[i]);

L[i]=R[i-1];

}

C[0]=L[2];

C[1]=R[2];

cout<<"Khoi ma hoa: "<<C;

}

**Bài tập 22:**

Lập trình mô phỏng hoạt động của mật mã Feistel đơn giản với 2 vòng xử lý:

Nhập chuỗi plaintext từ bàn phím, chia chuỗi thành các khối dài *m* = 2*w* = *16* bít.

Nhập khóa *K* (dài *8* bít) từ bàn phím. Khóa *Ki* được sinh ra từ khóa *K* nhờ phép dịch trái *K* i lần.

Hàm *F* thực hiện phép cộng giữa *Ri-1* với *Ki*.

Hiện chuỗi ciphertext ra màn hình.

#include <iostream>

using namespace std;

char F(char R,char Ki)

{return R+Ki;}

string MHKhoi(char P0,char P1, char K)

{

char K[3],L[3],R[3];

//Ma hoa khoi

string C=" ";

R[0]=P0;

L[0]=P1;

K[0]=k;

for(int i=1;i<=2; i++)

{

K[i]=K[0]<<i;

R[i]=L[i-1]^F(R[i-1],K[i]);

L[i]=R[i-1];

}

C[0]=L[2];

C[1]=R[2];

return C;

}

int main()

{

string P,C;char k;

cout<<"Nhap chuoi plaintext: ";getline(cin,P);

cout<<"Nhap khoa K: ";cin>>k;

if(P.size()%2==1) P=P+'X';

for(int i=0;i<P.size()/2;i=i+2) C=C+MHKhoi(P[i],P[i+1,k);

cout<<"Chuoi ma hoa: "<<C;

}