

**Buổi 1:**

*Bài 1:*

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

Lập trình khôi phục lại chuỗi ban đầu

#include<iostream> using namespace std;l

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;

//Giai ma

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

{

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

}

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

}

*Bài 2:*

-Lập trình nhập một chuỗi ký tự từ bàn phím, cộng mỗi phần tử của chuỗi với K(K là một số nguyên 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;

//Giai ma

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

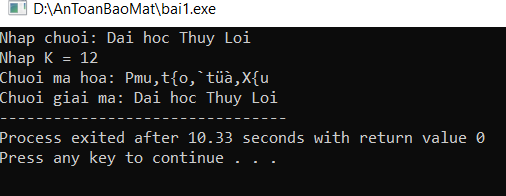
{

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

}

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

}



P ký tự bản rõ ->E hàm mã hóa -> C bản mã hóa -> D hàm giải mã Mật mã là mã hóa nhằm mục đích giữ bí mật

Mã hóa là biến đổi thông tin từ dạng này sang dạng khác Vành 16 là tập hợp gồm 26 số nguyên

Vành là tập hợp các số nguyên theo vòng tròn

**Buổi 2:**

*Bài 1: Caesar*

-Lập trình nhập một chuỗi ký tự từ bàn phím, mã hóa chuỗi bằng thuật toán CAESAR tổng quát với khóa K nhập từ bàn phím. Hiện chuỗi mới ra màn hình.

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

#include<iostream> //Mat ma Caesar using namespace std;

int Kt\_So(char c) //Kí tu doi sang so

{ return c-'A'; }

char So\_Kt(int n) //So sang ky tu

{ return 'A' +n; }

int main()

{

string P, C; int K;

cout<<"Nhap chuoi: ";getline(cin,P); C=P; cout<<"Nhap K = "; cin>>K;

//Mã hóa

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;

//giai ma

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

{

int m; m = Kt\_So(C[i]); m = (m - K + 26) % 26;

P[i] = So\_Kt(m);

}

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

}

*Bài 2: Bẻ khóa Caesar*

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

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

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

#include<iostream> // Be khoa Mat ma Caesar using namespace std;

int Kt\_So(char c) //Kí tu doi sang so

{ return c-'A'; }

char So\_Kt(int n) //So sang ky tu

{ return 'A'+n; } int main()

{

string P, C; int K;

cout<<"Nhap chuoi: ";getline(cin,C); P=C; for(K = 1; K<=25; K++)

{

//Giai ma

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

{

int m; m = Kt\_So(C[i]); m = (m - K + 26) % 26;

P[i] = So\_Kt(m);

}

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

}

}

*Bài 3: a , a-1 trong Z26*

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

**Bài 4: Affine**

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

#include<iostream> // Mat ma Affine using namespace std;

int Kt\_So(char c) //Kí tu doi sang so

{ return c-'A'; }

char So\_Kt(int n) //So sang ky tu

{ return 'A'+n; }

int main()

{

string P, C; int a = 0, b;

cout<<"Nhap chuoi: ";getline(cin,P); C=P; while (a<1 || a>25 || a%2==0 || a==13)

{

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

}

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

//Ma hoa

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

{

int m; m = Kt\_So(P[i]); m = (a \* m + b) % 26; C[i] = So\_Kt(m);

}

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

**Bài 5: Giải mã Affine**

Lập trình giải mã Affine để khôi phục lại chuỗi ban đầu:

-Đầu vào chương trình là cặp số {a, b} và chuỗi ký tự cipher text từ Bài tập 1.

-Đầu ra chương trình là chuỗi ký tự plain text.

#include<iostream> // Mat ma Affine using namespace std;

int Kt\_So(char c) //Kí tu doi sang so

{ return c-'A'; }

char So\_Kt(int n) //So sang ky tu

{ return 'A'+n; }

int main()

{

string P, C; int a = 0, b;

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

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

{

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

}

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

//Ma hoa

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

{

int m; m = Kt\_So(P[i]); m = (a \* m + b) % 26; C[i] = So\_Kt(m);

}

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

//Giai ma int a1;

for(int i = 1; i<=25; i++) if((i\*a)%26==1) {a1 = i;break; }

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

{

int m; m = Kt\_So(C[i]); m = a1\*(m-b+26)%26;

P[i]= So\_Kt(m);

}

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

}

Buổi 3:

**Bài 1: Bẻ khóa Affine**

Lập trình bẻ khóa mật mã Afine bằng phương pháp Brute-force:

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

-Hãy xác định cặp số {a, b} đã sử dụng và nội dung của plain text ban đầu. #include<iostream> // Be khoa Mat ma Affine

using namespace std;

int Kt\_So(char c) //Kí tu doi sang so

{ return c-'A'; }

char So\_Kt(int n) //So sang ky tu

{ return 'A'+n; }

int main()

{

string P, C; int a = 0, b;

cout<<"Nhap chuoi: ";getline(cin,C); P=C; for(a = 1; a<=25; a = a+2)

if (a!=13)

{

//Giai ma

int a1;

for(int i = 1; i<=25; i++) if((i\*a)%26==1) {a1 = i;break; }

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

{

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

{

int m; m = Kt\_So(C[i]); m = a1\*(m-b+26)%26;

P[i]= So\_Kt(m);

}

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

}

}

}

**Bài 2: Mã hóa Monoalphabetic**

-Lập trình nhập một chuỗi ký tự từ bàn phím, mã hóa chuỗi bằng thuật toán Monoalphabetic với khóa K nhập từ bàn phím (Khóa K là một chuỗi gồm 26 chữ cái có trật tự bất kỳ). Hiện chuỗi mới ra màn hình.

#include<iostream> // Mat ma MonoAiphabetic using namespace std;

int main()

{

string P, C, K;

string B = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

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;

}

**Bài 3: Giải mã Monoalphabetic**

Lập trình giải mã Monoalphabetic để khôi phục lại chuỗi ban đầu:

-Đầu vào chương trình là khóa K và chuỗi ký tự cipher text từ Bài tập 1.

-Đầu ra chương trình là chuỗi ký tự plain text #include<iostream> // Mat ma MonoAiphabetic using namespace std;

int main()

{

string P, C, K;

string B = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

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;

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

}

Bài tập về nhà:

1. Lập trình nhập một chuỗi ký tự từ bàn phím. Đếm xem ký tự đầu tiên của chuỗi xuất hiện bao nhiêu lần trong chuỗi?
2. Hãy giải mã thông điệp tiếng Anh dưới đây(được mã hóa bởi phương pháp Monoalphabetic):

UZQSOVUOHXMOPVGPOZPEVSGZWSZOPFP ESXUDBMETSXAIZVUEPHZHMDZSHZZOW SFPAPPDTSVPQUZWYMXUZUHSXEPYEPOP DZSZUFPOMBZWPFUPZHMDJUDTMOHMQ

**Buổi 4:**

**Bài 1: BT Playfair**

-Chọn một từ khóa bất kỳ rồi xây dựng ma trận khóa Playfair

-Chọn một plaintext bất kỳ, áp dụng ma trận khóa trên để tạo ra ciphertext

-Thử giải mã ciphertext rồi so sánh với plaintext ban đầu

Vd của thầy

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| DA | IH | OC | TH | UY | LO | IX |
| BR | FB | MH | PD | WC | PM | SA |
| BÀI 1: |  |  |  |  |  |  |

ABCDEFGHIJKLMNOPQRSTUVWXYZ P: HOANGTHIMAI

K: GIADINH

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

P = HOANGTHIMAI

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| P | HO | AN | GT | HI | MA | IX |
| C | EK | DG | DQ | BG | CX | AW |

**Bài 2: Code PlayFair**

1. Nhập một từ khóa bất kỳ. Hãy loại bỏ các ký tự trùng lặp trong chuỗi
2. Điền thêm vào chuỗi trên những ký tự còn lại trong bảng chữ cái
3. Đặt chuỗi thu được vào ma trận khóa 5x5

//a)

#include<iostream> // Ma tran PlayFair 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 ma hoa: "<<K;

}

//b)

#include<iostream> // Ma tran PlayFair using namespace std;

int main()

{

string K;

string B = "ABCDEFGHIKLMNOPQRSTUVWXYZ";

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

//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;

}

//c)

#include<iostream> // Ma tran PlayFair using namespace std;

int main()

{

string K;

string B = "ABCDEFGHIKLMNOPQRSTUVWXYZ";

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

//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 <<endl;

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<<"Ma tran khoa: "<<endl; for(int i = 0; i<5; i++)

{

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

}

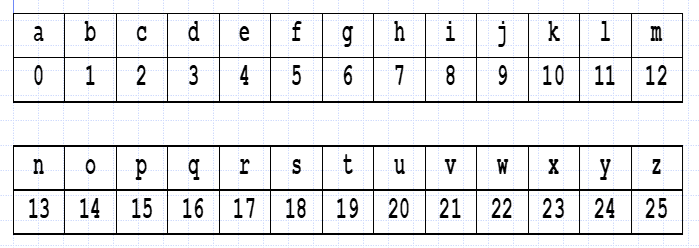
}

**Buổi 5:**

**Bài 1: BT Mật mã Hill**

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

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



Cách 1 :

paymoremoney pay mor emo ney

p1p2p3

1. pay p1 = ‘p’=15 p2 = ‘a’ = 0 p3 = ‘y’ = 24

c1 = 17xp1 + 17xp2 + 5xp3 = 17x15 + 17x0 + 5x24=375 = 11 = ‘L’ c2 = 21xp1 + 18xp2 + 21xp3 = 21x15 + 18x0 + 21x24=819=13=’N’ c3 = 2xp1 + 2xp2 + 19xp3 = 2x15 + 2x0 + 19x24=486 = 18 = ‘S’

PAY được thay bằng LNS Giải mã:

p1 = c1 \* 4 + c2 \* 9 + c3 \* 15 = 11 \* 4 + 13 \* 9 + 18 \* 15 = 431 = 15 = ‘P’

p2 = c1 \* 15 + c2 \*17 + c3 \* 6 = 11\* 15 + 13 \* 17 + 18 \* 6 = 494 = 0 = ‘A’

p3 = c1\*24 + c2 \* 0 + c3 \* 17= 11 \* 24 + 13 \* 0 + 18 \* 17 = 570 = 24 = ‘Y’

1. mor

p1 = ‘m’ = 12 p2 = ‘o’ = 14 p3 = ‘r’ = 17

c1 = 17xp1 + 17xp2 + 5xp3 = 17x12 + 17x14 + 5x17=527 = 7 = ‘H’ c2 = 21xp1 + 18xp2 + 21xp3 = 21x12 + 18x14 + 21x17=861=3=’D’ c3 = 2xp1 + 2xp2 + 19xp3 = 2x12 + 2x14 + 19x17=375 = 11 = ‘L’

MOR được thay bằng HDL Giải mã:

p1 = c1 \* 4 + c2 \* 9 + c3 \* 15 = 7 \* 4 +3 \* 9 + 11 \* 15 = 220 = 12 = ‘M’

p2 = c1 \* 15 + c2 \*17 + c3 \* 6 = 7\* 15 + 3 \* 17 + 11 \* 6 = 222 = 14 = ‘O’

p3 = c1\*24 + c2 \* 0 + c3 \* 17= 7 \* 24 + 3 \* 0 + 11 \* 17 =355 = 17 = ‘R’

1. emo

p1 = ‘e’ = 4 p2 = ‘m’ = 12 p3 = ‘o’ = 14

c1 = 17xp1 + 17xp2 + 5xp3 = 17x4 + 17x12 + 5x14=342 = 4 = ‘E’ c2 = 21xp1 + 18xp2 + 21xp3 = 21x4 + 18x12 + 21x14=594=22=’W’ c3 = 2xp1 + 2xp2 + 19xp3 = 2x4 + 2x12 + 19x14=298 = 12 = ‘M’

MOR được thay bằng EWM Giải mã:

p1 = c1 \* 4 + c2 \* 9 + c3 \* 15 = 4 \* 4 +22 \* 9 + 12 \* 15 = 394 = 4 = ‘E’

p2 = c1 \* 15 + c2 \*17 + c3 \* 6 = 4\* 15 + 22 \* 17 + 12 \* 6 = 506 = 12 = ‘M’

p3 = c1\*24 + c2 \* 0 + c3 \* 17= 4 \* 24 + 22 \* 0 + 12 \* 17 =300= 14 = ‘O’

1. ney

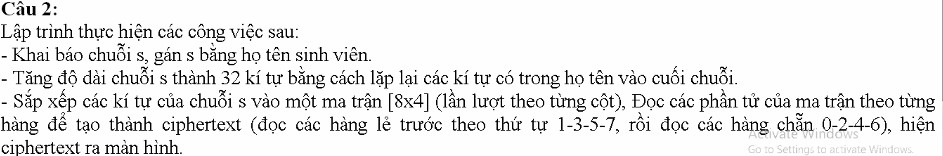
p1 = ‘n’ = 13 p2 = ‘e’ = 4 p3 = ‘y’ = 24

c1 = 17xp1 + 17xp2 + 5xp3 = 17x13 + 17x4 + 5x24=409 = 19 = ‘T’ c2 = 21xp1 + 18xp2 + 21xp3 = 21x13 + 18x4 + 21x24=849=17=’R’ c3 = 2xp1 + 2xp2 + 19xp3 = 2x13 + 2x4 + 19x24=490 = 22 = ‘W’

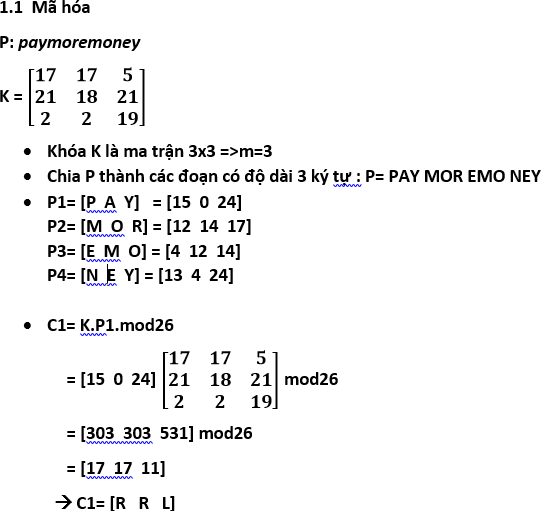
MOR được thay bằng TRW Giải mã:

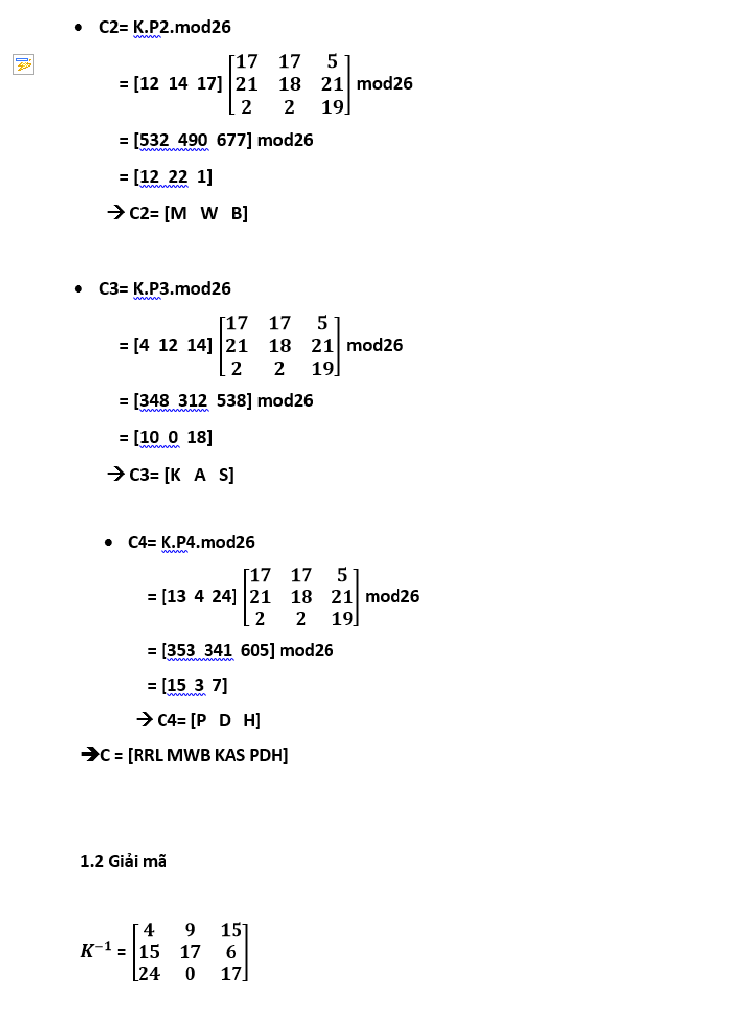
p1 = c1 \* 4 + c2 \* 9 + c3 \* 15 = 19 \* 4 +17 \* 9 +22 \* 15 = 559 = 13 = ‘N’

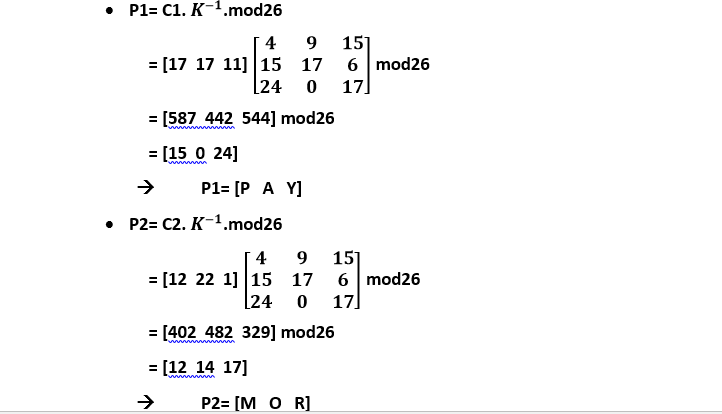
p2 = c1 \* 15 + c2 \*17 + c3 \* 6 = 19\* 15 + 17 \* 17 + 22 \* 6 = 706 = 4 = ‘E’

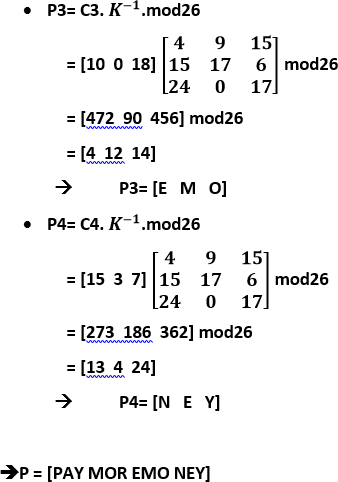
p3 = c1\*24 + c2 \* 0 + c3 \* 17= 19 \* 24 +17 \* 0 + 22 \* 17 =830 = 24 = ‘Y’

Cách 2:









**Bài 2: BT Vigenere**

-Chọn một plaintext và một khóa K bất kỳ

-Áp dụng thuật toán Vigenere để xác định cirphertext Gợi ý: Có thể sử dụng cách tra bảng ở trang sau

plaintext: ILOVEFAMILY ILOVEF/AMILY

Viết theo dạng số là:8 11 14 21 4 5/0 12 8 11 24 Từ khóa: MAIMAI tương ứng với:

K = (12, 0, 8, 12, 0, 8)

Cộng từng nhóm 6 ký tự của plaintext với K ta có: 8 11 14 21 4 5 /0 12 8 11 24

12 0 8 12 0 8 /12 0 8 12 0

20 11 22 7 4 13 / 12 12 16 23 24

Cirphertext tương ứng là: ULWHEN/MMQXY

**Bài 3: Mã hóa Vigenere**

1. Lập trình nhập chuỗi ký tự plaintext từ bàn phím, mã hóa chuỗi bằng thuật toán Vigenere với khóa K (là một chuỗi ký tự) nhập từ bàn phím. Hiện chuỗi mới ra màn hình.
2. Lập trình giải mã để khôi phục lại chuỗi ban đầu. a)

#include<iostream> // Be khoa Mat ma Vigenere using namespace std;

int Kt\_So(char c) //Kí tu doi sang so

{ return c-'A'; }

char So\_Kt(int n) //So sang ky tu

{ return 'A'+n; }

int main()

{

string P, C, K;

cout<<"Nhap chuoi plaintext: ";cin>>P; C=P; cout<<"Nhap chuoi 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;

}

b)

#include<iostream> // Be khoa Mat ma Vigenere using namespace std;

int Kt\_So(char c) //Kí tu doi sang so

{ return c-'A'; }

char So\_Kt(int n) //So sang ky tu

{ return 'A'+n; }

int main()

{

string P, C, K;

cout<<"Nhap chuoi plaintext: ";cin>>P; C=P; cout<<"Nhap chuoi 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;

//Giai ma j = 0;

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 giai ma: "<<P;

}

**Bài 4: BT Vernman**

-Giải sử plaintext là “HA NOI” (các ký tự trong chuỗi có thể được biểu diễn dưới dạng nhị phân bảng mã chuẩn ASCII), khóa K là một dãy nhị phân 8 bit 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” = 48 41 20 4E 4F 49 = 0100 1000 0100 0001 0010 0000 0100 1110 0100

1111 0100 1001

-Mã hóa:

Plaintext = 0100 1000 0100 0001 0010 0000 0100 1110 0100 1111 0100 1001

Khóa K = 1001 0011

Ciphertext = 1101 1011 1101 0010 1011 0011 1101 1101 1101 1100 1101 1010

-Giải mã:

Ciphertext = 1101 1011 1101 0010 1011 0011 1101 1101 1101 1100 1101 1010

Khóa K = 1001 0011 1001 0011 1001 0011 1001 0011 1001 0011 1001 0011

Plaintext = 0100 1000 0100 0001 0010 0000 0100 1110 0100 1111 0100 1001

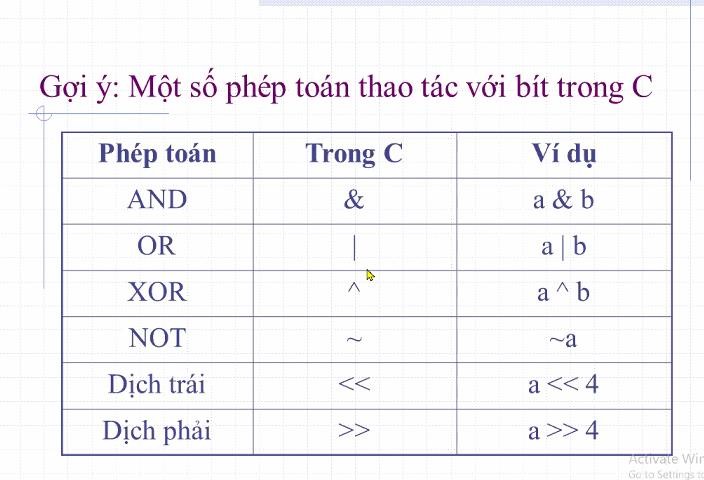
Buổi 6:

**Bài 1: Mã hóa Vernman**

-Nhập một chuỗi plaintext từ bàn phím

-Nhập một khóa K(dài 8 bit)

-Mã hóa chuỗi ban đầu bằng cách XOR các ký tự của nó với K

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

\*CHUỖI MÃ HÓA

#include<iostream> // Mat ma Vernman using namespace std;

int main()

{

string P, C; char K;

cout<<"Nhap plaintext: ";getline(cin, P); C=P; cout<<"Nhap chuoi K = ";cin>>K;

//Ma hoa

for(int i = 0; i<P.size(); i++) C[i]=P[i]^K; cout<<"Chuoi ma hoa: "<<C;

}

\*CHUỖI GIẢI MÃ

#include<iostream> // Mat ma Vernman using namespace std;

int main()

{

string P, C; char K;

cout<<"Nhap plaintext: ";getline(cin, P); C=P; cout<<"Nhap chuoi 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<C.size(); i++) P[i]=C[i]^K; cout<<endl<<"Chuoi giai ma: "<<P;

}

BÀI HOÀN THIỆN

#include<iostream> // Mat ma Vernman using namespace std;

int main()

{

string P, C, K;

cout<<"Nhap chuoi plaintext: ";getline(cin, P); C=P; cout<<"Nhap chuoi K = ";getline(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 j = 0;

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

{

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

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

}

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

}

Bài 2: BT Vernman

Giả sử plaintext là “he is a nice person”

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Key: | 4 | 3 | 1 | 2 | 5 |
| Plaintext: | h | e | i | s | a |
|  | n | i | c | e | p |
|  | e | r | s | o | n |
|  | l | m | q | t | u |
|  | v | w | x | y | z |

Ciphertext: icsqxseotyeirmwhnelvapnuz

ABCDEFGHIKLMNOPQRSTUVWXYZ

Bài 3: Mã hóa Rain-fence

-Nhập một chuỗi plaintext

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

-Giải mã ciphertext

\*Mã hóa

#include<iostream> // Ky thuat Rain-fence using namespace std;

int main()

{

string P, C;

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];

else h2 = h2+P[i];

}

C = h1 + h2;

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

}

\*Giải mã

#include<iostream> // Ky thuat Rain-fence using namespace std;

int main()

{

string P, C;

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];

else h2 = h2+P[i];

}

C = h1 + h2;

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

//Giai ma P="";h1="";h2="";

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]; //phan tu cuoi cung cua h1 cout<<endl<<"Chuoi giai ma: "<<P;

}

**Bài 4: Code đổi hàng cột.**

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

-Sắp xếp các ký tự của chuỗi vào một ma trận 5x5(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> // Ky thuat Doi cho hang cot using namespace std;

int main()

{

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

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

//Ma hoa 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++) //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++)

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

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

}

//Giải mã

#include<iostream> // Ky thuat Doi cho hang cot using namespace std;

int main()

{

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

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

//Ma hoa 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++) //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++)

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[j][i]=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[i][j];

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

}

Buổi 7:

Bài 1: Mật mã Feistel

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

-Nhập một khối plaintext từ bàn phím, khối có độ dài m = 2w = 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 khối ciphertext ra màn hình.

#include<iostream> // feistel 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 khoi 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; //dich trai i lan 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;

}

\*Hoàn chỉnh #include<iostream> // feistel using namespace std;

char F(char R, char Ki)

{ return R+Ki; }

MHKhoi(char P0, char P1, char k) int main()

{

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

string C=" ";

//Ma hoa khoi R[0]=P0;

L[0]=P1;

K[0]=k;

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

{

K[i]=K[0]<<i; //dich trai i lan 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;

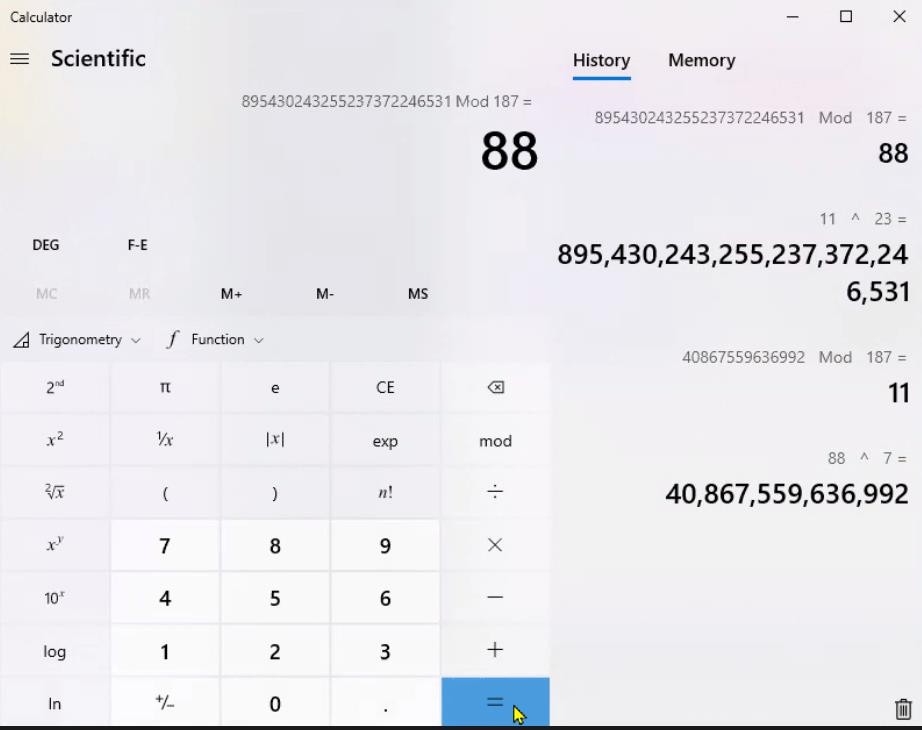
cout<<"Nhap chuoi plaintext: ";getline(cin,P); cout<<"Nhap khoa K: ";cin>>k;

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

for(int i = 0; i<P.size();i = i+2) C= C+MHKhoi(P[i],P[i+1],k); cout<<"Chuoi ma hoa: "<<C;

}

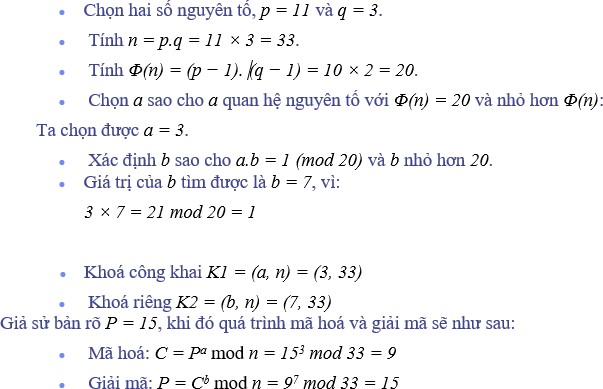
Buổi 8:



Bài 1: RSA

-Chọn một cặp số nguyên tố p, q

-Thực hiện các bước tạo khóa, mã hóa và giải mã theo thuật toán RSA

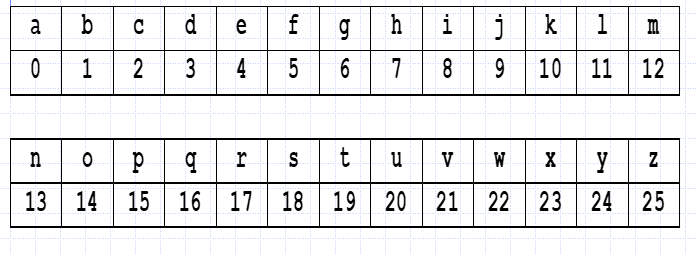
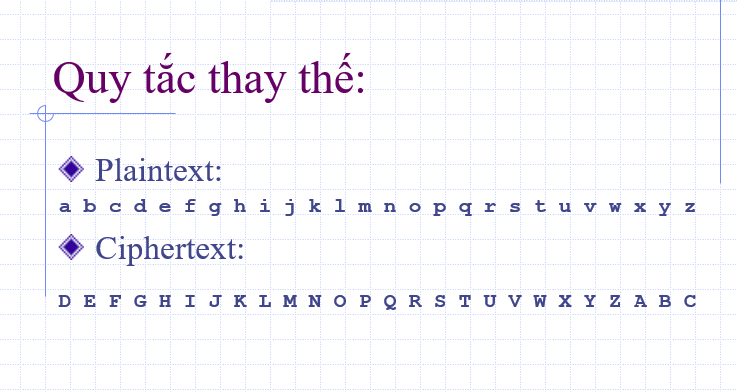


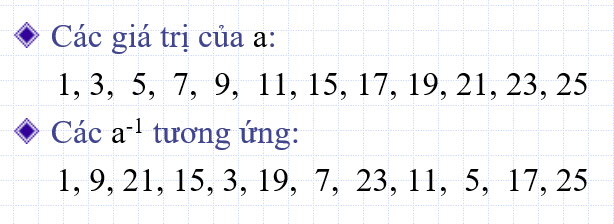
***Ôn giữa kì***

* 1. Lập trình
  2. Mã hóa và giải mã bằng tay SBD: 42

**Mật mã cổ điển**

1. Kỹ thuật thay thế
   * Mật mã CAESAR



* + - Mật mã Affine
    - Mật mã Monoalphabetic
    - Mật mã Polyalphabetic…

1. Kỹ thuật chuyển dịch - hoán vị Đề 1:

Câu 1: Mật mã Vigenere

Cho chuỗi Plaintext sau : KHONGBENHLATIENSUONGTUYETTRAN

1. Hãy mã hóa chuỗi nói trên bằng mật mã Vigenere, với khóa K là họ tên của sinh viên, m là độ dài họ tên
2. Giải mã ciphertext thu được và so sánh plaintext ban đầu. Bài làm:

**Mai:**

A.Mã hóa

-Khóa K: HOANGTHIMAI => m = 11

-Plaintext: KHONGBENHLATIENSUONG TUYETTRAN

-Do m = 11, ta sẽ tách thành plaintext thành từng nhóm 11 ký tự: KHONGBENHLA/TIENSUONGTU/YETTRAN

-Viết theo dạng số là:

10 7 14 13 6 1 4 13 7 11 0/19 8 4 13 18 20 14 13 6 19 20/24 4 19 19 17 0 13

-Từ khóa HOANGTHIMAI tương ứng với:

K = (7, 14, 0, 13, 6, 19, 7, 8, 12, 0, 8)

-Cộng từng nhóm 11 ký tự của plaintext với K ta có:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | 7 | 14 13 6 1 4 13 7 11 0/19 8 4 | 13 18 20 14 13 6 19 20/24 | 4 19 19 17 0 | 13 |
| 7 | 14 | 0 13 6 19 7 8 12 0 8/7 14 0 | 13 6 19 7 8 12 0 8/ 7 | 14 0 13 6 19 7 | |
| 17 | 21 | 14 0 12 20 11 21 19 11 8/0 22 4 | 0 24 13 21 21 18 19 2/ 5 | 18 19 6 23 19 20 | |

-Ciphertext tương ứng là: RVOAMULVTLIAWEAYNVVSTCFSTGXTU

Giải mã

-Khóa K: HOANGTHIMAI => m =11

-Ciphertext: RVOAMULVTLIAWEAYNVVSTCFSTGXTU

-Do m = 11, ta sẽ tách Ciphertext thành từng nhóm 11 ký tự: RVOAMULVTLI/AWEAYNVVSTC/FSTGXTU

-Viết theo dạng số là:

17 21 14 0 12 20 11 21 19 11 8/0 22 4 0 24 13 21 21 18 19 2/5 18 19 6 23 19 20

* Trừ từng nhóm 16 ký tự của Ciphertext với *K* ta có:

17 21 14 0 12 20 11 21 19 11 8/0 22 4 0 24 13 21 21 18 19 2/ 5 18 19 6 23 19 20

7 14 0 13 6 19 7 8 12 0 8/7 14 0 13 6 19 7 8 12 0 8/ 7 14 0 13 6 19 7

10 7 14 13 6 1 4 13 7 11 0/19 8 4 13 18 20 14 13 6 19 20/

24 4 19 19 17 0 13

* + Plaintext : KHONGBENHLATIENSUONGTUYETTRAN

**Hải**

A. Mã hóa

* + - Khóa K: NGUYENTHIHONGHAI =>m=16
    - Plaintext : KHONGBENHLATIENSUONGTUYETTRAN
    - Do m=16, ta sẽ tách plaintext thành từng nhóm 16 kí tự: KHONGBENHLATIENS/UONGTUYETTRAN
    - Viết theo dạng số là:

10 7 14 13 6 1 4 13 7 11 0 19 8 4 13 18 / 20 14 13 6 19 20 24 4 19 19 17 0 13

* + - Từ khoá NGUYENTHIHONGHAI tương ứng với:

*K* = (13, 6, 20, 24, 4, 13, 19, 7, 8, 7, 14, 13, 6, 7, 0, 8)

* + - Cộng từng nhóm 16 ký tự của plaintext với *K* ta có:

10 7 14 13 6 1 4 13 7 11 0 19 8 4 13 18 / 20 14 13 6 19 20 24 4 19 19 17 0 13

13 6 20 24 4 13 19 7 8 7 14 13 6 7 0 8 / 13 6 20 24 4 13 19 7 8 7 14 13 6

23 13 8 11 10 14 23 20 15 18 14 6 14 11 13 0/ 7 20 7 4 23 7 17 11 1 0 5 13 19

* + - Ciphertext tương ứng là: XNILKOXUPSOGOLNA HUHEXHRLBAFNT

B.Giải mã

* + Khóa K: NGUYENTHIHONGHAI =>m=16
  + Ciphertext: XNILKOXUPSOGOLNA HUHEXHRLBAFNT
  + Do m=16, ta sẽ tách Ciphertext thành từng nhóm 16 kí tự:

XNILKOXUPSOGOLNA / HUHEXHRLBAFNT

* + Viết theo dạng số là:

23 13 8 11 10 14 23 20 15 18 14 6 14 11 13 0/ 7 20 7 4 23 7 17 11 1 0 5 13 19

* + Trừ từng nhóm 16 ký tự của Ciphertext với *K* ta có:

23 13 8 11 10 14 23 20 15 18 14 6 14 11 13 0/ 7 20 7 4 23 7 17 11 1 0 5 13 19

13 6 20 24 4 13 19 7 8 7 14 13 6 7 0 8 / 13 6 20 24 4 13 19 7 8 7 14 13 6

10 7 14 13 6 1 4 13 7 11 0 19 8 4 13 18 / 20 14 13 6 19 20 24 4 19 19 17 0 13

* + Plaintext : KHONGBENHLATIENSUONGTUYETTRAN

Câu 2:

Lập trình thực hiện các công việc sau:

1. Khai báo chuỗi s, gán s bằng họ tên sinh viên
2. Tăng độ dài chuỗi s bằng 30 ký tự bằng cách lặp lại các ký tự trong họ tên vào cuối chuỗi
3. Sắp xếp các ký tự chuỗi s vào 1 ma trận [3x10] (lần lượt theo từng hàng) . Đọc các phần tử của ma trận theo từng cột để tạo thành ciphertext hiện ra màn hình.

#include<iostream> #include<string> using namespace std; int main()

{

string s,c; int k[5];

char A[5][5]; cout<<"Nhap chuoi s: "; getline(cin,s);

cout<<"Nhap khoa k: "<<endl; for(int i=0;i<5;i++){ cout<<"k"<<i+1<<": "; cin>>k[i];

}

if(s.size()<25){

int dem;

dem= 25 - s.size(); int staticChuoi=0; while (dem!=0){ s=s+s[staticChuoi]; staticChuoi++; dem--;

}

}

int index=0;

for (int i=0;i<5;i++){ for (int j=0;j<5;j++){ A[i][j]=s[index];

cout<<A[i][j]<<" "; index ++;

}

cout<<endl;

}

cout<<" "<<endl;

//ma hoa

for (int j=0;j<5;j++){ for (int i=0; i<5;i++){ c=c+A[i][k[j]-1];

}

}

cout<<endl<<"chuoi ciphertext: "<<c<<endl<<endl;

//giai ma string giaima; int index1=0;

for (int i=0;i<5;i++){ for (int j=0;j<5;j++){ A[j][k[i]-1]=c[index1]; index1++;

}

}

for(int i=0;i<5;i++){ for(int j=0;j<5;j++) giaima=giaima + A[i][j];

}

cout<<"Chuoi plaintext: "<<giaima<<endl;

return 0;

}

Đề 2 :

Câu 1: Mật mã Verman (Giải mã bằng tay)

-Giải sử plaintext là “TIEN VE HA NOI 1945” (các ký tự trong chuỗi có thể được biểu diễn dưới dạng nhị phân bảng mã chuẩn ASCII), khóa K là Họ và Tên sinh viên dưới dạng mã nhị phân. Hãy mã hóa chuỗi ban đầu bằng phương pháp Vernman

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

Khóa K = HOANG THI MAI = 48 4F 41 4E 47 20 54 48 49 20 4D 41 49

= 0100 1000 0100 1111 0100 0001 0100 1110 0100 0111 0010 0000 0101 0100 0100

1000 0100 1001 0010 0000 0100 1101 0100 0001 0100 1001

“TIEN VE HA NOI 1945” =54 49 45 4E 20 56 45 20 48 41 20 4E 4F 49 20 31 39 34 35

=0101 0100 0100 1001 0100 0101 0100 1110 0010 0000 0101 0110 0100 0101 0010

0000 0100 1000 0100 0001 0010 0000 0100 1110 0100 1111 0100 1001 0011 0001 0011

1001 0011 0100 0011 0101

-Mã hóa:

Plaintext = 0101 0100 0100 1001 0100 0101 0100 1110 0010 0000 0101 0110 0100

0101

0010 0000 0100 1000 0100 0001 0010 0000 0100 1110 0100 1111 0100 1001 0010 0000

0011 0001 0011 1001 0011 0100 0011 0101

Khóa K = 0100 1000 0100 1111 0100 0001 0100 1110 0100 0111 0010 0000 0101

0100

0100 1000 0100 1001 0010 0000 0100 1101 0100 0001 0100 1001

Ciphertext=0001 1100 0000 0110 0000 0100 0000 0000 0110 0111 0111 0110 0001

0001

0110 1000 0000 0001 0110 0001 0110 1101 0000 1111 0000 0110 0000 0001 0110 1111

0111 0000 0111 0111 0111 0011 0001 0101

-Giải mã:

Ciphertext=0001 1100 0000 0110 0000 0100 0000 0000 0110 0111 0111 0110 0001

0001

0110 1000 0000 0001 0110 0001 0110 1101 0000 1111 0000 0110 0000 0001 0110 1111

0111 0000 0111 0111 0111 0011 0001 0101

Khóa K = 0100 1000 0100 1111 0100 0001 0100 1110 0100 0111 0010 0000 0101

0100

0100 1000 0100 1001 0010 0000 0100 1101 0100 0001 0100 1001

Plaintext = 0101 0100 0100 1001 0100 0101 0100 1110 0010 0000 0101 0110 0100

0101

0010 0000 0100 1000 0100 0001 0010 0000 0100 1110 0100 1111 0100 1001 0010 0000

0011 0001 0011 1001 0011 0100 0011 0101

Câu 2:(Lập Trình)

-Nhập vào chuỗi S, và gán S là Họ và Tên của sinh viên

-Tăng kích thước của chuỗi S lên thành 36 kí tự

VD: ký tự của chuỗi S là ViSaoLangLe thì chuỗi S mới 36 ký tự là ViSaoLangLeViSaoLangLeViSaoLangLeViS

-Sắp xếp các ký tự của chuỗi mới vào một ma trận 6 x 6 (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.

Đề 3:

Câu 1: Mật mã Affine Mã hóa bằng tay

Cho chuỗi plaintext sau: KHOACONGNGHETHONGTIN

-Hãy mã hóa chuỗi nói trên bằng mật mã Affine, với khóa K là cặp số {a,b} lấy từ 2 số cuối của mã sinh viên

* a là số hàng chục(nếu a chẵn thì cộng thêm 1)
* b là số hàng đơn vị

-Giải mã ciphertext thu được và so sánh với plaintext ban đầu.

**Mai**

-K = {a,b} = {3, 9}, a-1 = 9

-P = KHOACONGNGHETHONGTIN

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Pi | K | H | O | A | C | O | N | G | N | G | H | E | T | H | O | N | G | T | I | N |
|  | 10 | 7 | 14 | 0 | 2 | 14 | 13 | 6 | 13 | 6 | 7 | 4 | 19 | 7 | 14 | 13 | 6 | 19 | 8 | 13 |

-Mã hóa theo công thức:

Ci = E(Pi, {a,b}) = (aPi + b) mod 26 = (3Pi + 9) mod 26

=>Áp dụng công thức trên ta có:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 13 | 4 | 25 | 9 | 15 | 25 | 22 | 1 | 22 | 1 | 4 | 21 | 14 | 4 | 25 | 22 | 1 | 14 | 7 | 22 |
|  | N | E | Z | J | P | Z | W | B | W | B | E | V | O | E | Z | W | B | O | H | W |

-Giải mã theo công thức:

*Pi* = *D*(*Ci, {a,b}*) = *a-1*(*Ci* – *b*) mod 26 = 9(*Ci* – *9*) mod 26

=>Áp dụng công thức trên ta có:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | N | E | Z | J | P | Z | W | B | W | B | E | V | O | E | Z | W | B | O | H | W |
|  | 10 | 7 | 14 | 0 | 2 | 14 | 13 | 6 | 13 | 6 | 7 | 4 | 19 | 7 | 14 | 13 | 6 | 19 | 8 | 13 |
|  | K | H | O | A | C | O | N | G | N | G | H | E | T | H | O | N | G | T | I | N |

=>KL: Giải mã được kết quả trùng với Plaintext ban đầu

**Hải**

Mật mã Affine

* K= {a, b} ={7,1} , a-1=15
* P= KHOACONGNGHETHONGTIN

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Pi | K | H | O | A | C | O | N | G | N | G | H | E | T | H | O | N | G | T | I | N |
|  | 10 | 7 | 14 | 0 | 2 | 14 | 13 | 6 | 13 | 6 | 7 | 4 | 19 | 7 | 14 | 13 | 6 | 19 | 8 | 13 |

* **Mã hóa** theo công thức :

Ci = E(Pi, {a,b}) = (aPi + b) mod 26 = (7Pi + 1) mod 26

=>Áp dụng công thức trên ta có:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| C  i | C 1 | C 2 | C 3 | C 4 | C 5 | C 6 | C 7 | C 8 | C 9 | C 10 | C 11 | C 12 | C 13 | C 14 | C 15 | C 16 | C 17 | C 18 | C 19 | C 20 |
|  | 1  9 | 2  4 | 2  1 | 1 | 1  5 | 2  1 | 1  4 | 1  7 | 1  4 | 17 | 24 | 3 | 4 | 24 | 21 | 14 | 17 | 4 | 5 | 14 |
|  | T | Y | V | B | P | V | O | R | O | R | Y | D | E | Y | V | O | R | E | F | O |

* + Giải mã theo công thức:

*Pi* = *D*(*Ci, {a,b}*) = *a-1*(*Ci* – *b*) mod 26 = 15(*Ci* – *1*) mod 26

=>Áp dụng công thức trên ta có:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| C  i | T | Y | V | B | P | V | O | R | O | R | Y | D | E | Y | V | O | R | E | F | O |
|  | 1  9 | 2  4 | 2  1 | 1 | 1  5 | 2  1 | 1  4 | 1  7 | 1  4 | 1  7 | 2  4 | 3 | 4 | 2  4 | 2  1 | 1  4 | 1  7 | 4 | 5 | 1  4 |
| Pi | 1  0 | 7 | 1  4 | 0 | 2 | 1  4 | 1  3 | 6 | 1  3 | 6 | 7 | 4 | 1  9 | 7 | 1  4 | 1  3 | 6 | 1  9 | 8 | 1  3 |
|  | K | H | O | A | C | O | N | G | N | G | H | E | T | H | O | N | G | T | I | N |

=>KL: Giải mã được kết quả trùng với Plaintext ban đầu Câu 2: Lập trình thực hiện các công việc sau:

-Khai báo chuỗi khóa K, gán K bằng Họ Tên sinh viên

-Loại bỏ các ký tự trùng lặp trong K. Điền thêm vào cuối chuỗi ký tự còn lại của bảng chữ cái nhằm tạo ra chuỗi K có 26 ký tự khác nhau

-Nhập plaintext từ bàn phím mã hóa nó bằng thuật toán Monoalphabetic với khóa K nói trên.

-Giải mã ciphertext thu được và so sánh với plaintext ban đầu.