Nama : Join Valentino Tampubolon

NPM : 140810190020

Kelas : B

Enkripsi

```
#include <iostream>
#include <vector>
using namespace std;

int main(){
    int i,j,k,m;
    cout << "Masukkan Jumlah Pembagi\n";
    cin >> m;
    cout << "Matriks Kunci\n";
    int key[m][m];
    for(i=0; i < m; i++){
        for(j=0; j < m; j++){
            cin >> key[i][j];
        }
    }
    cout<<"Enter the message to encrypt\n";
    string s;
    cin>>s;
    int temp = (m-s.size()%m)%m;
    for(i=0;i<temp;i++){
        s+='x';
    }
    k=0;
    string ans="";
    while(k<s.size()){
        for(i=0;i<m;i++){
            int sum = 0;
            int temp = k;
            for(j=0;j<m;j++){
                sum += (key[i][j]%26*(s[temp++]-'a')%26)%26;
                sum = sum%26;
            }
            ans += (sum+'a');
        }
        k+=m;
    }
    cout << ans << '\n';

    return 0;
}
```

Proses enkripsinya berdasarkan algoritma di atas. Pertama kita perlu menentukan jumlah pembagi untuk kata yang akan kita enkripsi yang nantinya akan menjadi matriks kunci dimana akan menjadi matrik m x m. kemudian kita masukkan matriks key nya dengan looping kita akan memasukkan satu persatu kolom dan baris. Setelah itu kita ingin memasukkan kalimat yang akan kita enkripsi. Nah kita akan periksa jika jumlah kalimat tidak habis dibagi jumlah pembagi maka akan ditambah dengan huruf acak dimana pada code diatas ditambahin huruf 'x' sebanyak huruf sisa agar habis dibagi jumlah pembagi.

Kita siapkan k sebagai indicator untuk operasi while dimana nilai k ini akan meningkatkan sampai sama dengan jumlah kalimat yang ada. Serta kita siapkan ans untuk menampung jawaban akhir. Di dalam operasi while akan dilakukan enkripsi terus menerus sebanyak iterasi yang didapat sebelumnya. Nilai sum ada wadah untuk menampung sementara hasil dari perkalian matriks key dengan kalimat dan di mod 26. Kemudian nilai sum akan dimasukkan ke wadah ans dan ditambah 'a' sehingga akan menjadi sebuah susunan huruf bukan susunan angka.

Dekripsi

```
#include<iostream>
#include<vector>
using namespace std;

int modInverse(int a, int m){
    int y;
    int temp=a%m;
    for(int x =-m; x < m; x++){
        y = x;
        if((temp*x)%m==1){
            return y;
        }
    }
    return 0;
}

void getCofactor(vector<vector<int> > &a, vector<vector<int> > &temp, int p, i
nt 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){
        cout << "Inverse does not exist";
        return false;
    }
    int invDet = modInverse(det,26);
    cout<<det%26<<' '<<invDet<<'\n';
    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)%26;
    return true;
}
int main(){
    int i,j,k,n;
    cout << "Enter the size of key matrix\n";
    cin >> n;
    cout<<"Enter the key matrix\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)){
        cout<<"Inverse exist\n";
    }
    cout<<"Enter the message to decrypt\n";
    string s;
    cin>>s;
    k=0;
    string ans;
    while(k<s.size()){
        for(i=0;i<n;i++){
            int sum = 0;
            int temp = k;
            for(j=0;j<n;j++){
                sum += ((inv[i][j] + 26)%26*(s[temp++]-'a')%26)%26;
                sum = sum%26;
            }
            ans+=(sum+'a');
        }
        k+=n;
    }
    //ans+='\0';
    int f=ans.size()-1;
    while(ans[f]=='x'){
        f--;
    }
    for(i=0;i<=f;i++){
        cout<<ans[i];
    }
    cout<<'\n';
    return 0;
}
```

Dalam algoritma ini digunakan bentuk modul – modul diantaranya

1. ModInverse : Modul ini untuk mengcek elemen matrik merupakan invertible
2. getCoFactor : Modul ini untuk mencari ko factor dari sebuah matriks dimana digunakan operasi looping tetapi jika indeks I dan j sama maka akan dilewati tetapi jika tidak maka akan dilakkan operasi pengelompokkan dan pemindahan seperti mencari kofaktor biasanya dan disimpan sementara pada matriks temp

3. determinant : fungsi untuk menemukan determinan dimana jika hanya ada 1 elemen matriks maka determinannya adalah nilai matriks tersbeut. Jika tidak maka akan dicari lagi dengan looping dimana dirubah ke matriks paling sederhana yaitu ke matriks 2 x 2 kemudian dicari hasil perkalian silangnya juga yang bernilai negatif akan didapat ketika sign bernilai negatif.
4. Adjoint : Pada modul ini hanya untuk merubah tanda dari setiap elemen dimana elemen akan menjadi susunan postif negatif postif negatif dst.
5. Inverse : dalam modul ini semua modul tadi digabung untuk mengecek invers matriks.

Itu tadi beberapa modul untuk algoritma ini, kemudian pada fungsi main program dijalankan. Pertama, masukkan jumlah pembagi dan juga matriks keynya yang disesuaikan dengan jumlah pembagi. Lalu kita lihat apakah matriks key tersebut invertible atau tidak dengan menggunakan modul inverse sehingga harus mengeluarkan nilai true. Jika tidak maka program selesai, tetapi jika iya maka akan terjadi proses inverse matriks tersebut. Ketika sudah di inversi maka hasil inverse tinggal dikali dengan matriks yang akan didekripsi prosesnya sama seperti enkripsi dan mengeluarkan jawabannya.

Mencari Key

```
#include<iostream>
#include<iomanip>
#include<math.h>
#include<stdlib.h>

#define SIZE 10

using namespace std;

int main(){

        float p[SIZE][SIZE], x[SIZE], ratio, c[SIZE][SIZE], key[SIZE][SIZE];
        int i,j,k,n;

        cout<< setprecision(3)<< fixed;

        cout << "Bentuk Matriks : ";
        cin >> n;

        cout<<"Matrix Cipher : " << endl;
        for(i=1;i<=n;i++){
            for(j=1;j<=n;j++){
                cout<<"c["<< i<<"]"<< j<<"]= ";
                cin>> c[i][j];
            }
        }

        cout<<"Matrix Plain : " << endl;
        for(i=1;i<=n;i++){
```

```
        for(j=1;j<=n;j++){
            cout<<"p["<< i<<"]"<< j<<"]= ";
            cin>> p[i][j];
        }
    }

    for(i=1;i<=n;i++){
        for(j=1;j<=n;j++){

            if(i==j)
            {
                p[i][j+n] = 1;
            }
            else
            {
                p[i][j+n] = 0;
            }
        }
    }

    for(i=1;i<=n;i++){
        if(p[i][i] == 0.0)
        {
            cout<<"Mathematical Error!";
            exit(0);
        }
        for(j=1;j<=n;j++){
            if(i!=j){

                ratio = p[j][i]/p[i][i];
                for(k=1;k<=2*n;k++){
                    p[j][k] = p[j][k] - ratio*p[i][k];
                }
            }
        }
    }

    for(i=1;i<=n;i++)
    {
        for(j=n+1;j<=2*n;j++)
        {
            p[i][j] = p[i][j]/p[i][i];
        }
    }

    for(i=0; i < n; i++){
        for(j=0; j < n; j++){
            key[i][j]=0;
```

```
                for(k=0; k < n; k++){
                    key[i][j]+= (c[i][k]*p[k][j]);
                }
            }
        }

        cout<< endl<<"Key :"<< endl;
        for(i=1;i<=n;i++)
        {
            for(j=n+1;j<=2*n;j++)
            {
                cout<< key[i][j]<<"\t";
            }
            cout<< endl;
        }
        return(0);
}
```

Pada algoritma ini, pertama kita akan mengumpulkan bahan yang diperlukan yaitu bagian Cipher Teks dan juga Plainteksnya dengan operasi looping biasa dimana I sebagai row dan j sebagai coloumn. Ketika sudah kekumpul dengan rumus C x P^{-1} kita terlebih dahulu akan menginvers plainteks sehingga akan di dapat key nya.Ketika sudah dapat kita tinggal mengalikan keduanya dan mengeluarkan key.