**Assignment – 7**

**Implement the following Modern Block Ciphers techniques.  
1) Electronic Codebook (ECB) Mode  
2) Cipher Block Chaining (CBC) Mode  
3) Cipher Feedback (CFB) Mode  
4) Output Feedback (OFB) Mode**

**5) Counter (CTR) Mode**

1. **Electronic Codebook (ECB) Mode:**

**Code:**

#include<bits/stdc++.h>

using namespace std;

string generateKey(string key, int x)

{

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

{

if (x == i)

i = 0;

if (key.size() == x)

break;

key.push\_back(key[i]);

}

return key;

}

string cipherText(string str, string key)

{

string cipher\_text;

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

{

char x = (str[i] + key[i]) %26;

x += 'a';

cipher\_text.push\_back(x);

}

return cipher\_text;

}

int main()

{

int n;

cout<<"Enter the value of n(size of each block) : ";

cin>>n;

string plain, cipher="";

cout<<"Enter the plain text : ";

cin>>plain;

string key;

cout<<"Enter the key for vigenere cipher :";

cin>>key;

key = generateKey(key,n);

cout<<"key "<<key<<"\n";

int blocks;

if(plain.length()%n!=0)

{

int k= (plain.length()/n) \* n;

int g= plain.length()-k;

g=n-g;

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

plain.append("z");

}

blocks= plain.length()/n;

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

{ string tp= plain.substr(i\*n, i\*n+n);

string ci= cipherText(tp,key);

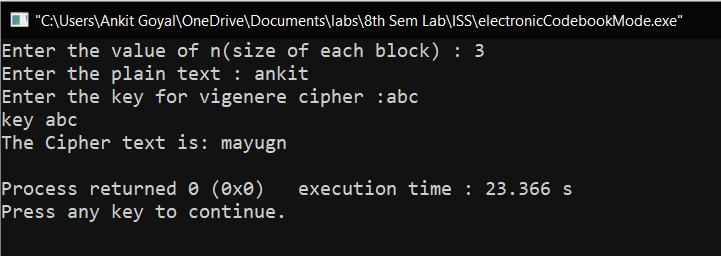
cipher.append(ci);

}

cout<<"The Cipher text is: "<<cipher<<"\n";

}

**Output:**



1. **Cipher Block Chaining (CBC) Mode**

**Code:**

#include<bits/stdc++.h>

using namespace std;

string xor\_operation(string a, string b)

{

string ans="";

int n=a.length();

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

{

char k= ((a[i]^b[i])%26 )+'a';

ans+=k;

}

return ans;

}

string generateKey(string key, int x)

{

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

{

if (x == i)

i = 0;

if (key.size() == x)

break;

key.push\_back(key[i]);

}

return key;

}

string cipherText(string str, string key)

{

string cipher\_text;

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

{

char x = (str[i] + key[i]) %26;

x += 'a';

cipher\_text.push\_back(x);

}

return cipher\_text;

}

int main()

{

int n;

cout<<"Enter the value of n(size of each block) : ";

cin>>n;

string plain, cipher="";

cout<<"Enter the plain text : ";

cin>>plain;

string key;

cout<<"Enter the key for vigenere cipher :";

cin>>key;

key = generateKey(key,n);

int blocks;

if(plain.length()%n!=0)

{

int k= (plain.length()/n) \* n;

int g= plain.length()-k;

g=n-g;

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

plain.append("z");

}

blocks= plain.length()/n;

string x;

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

{

string tp= plain.substr(i\*n, i\*n+n);

if(i!=0)

{

tp= xor\_operation(tp,x);

}

string ci= cipherText(tp,key);

x=ci;

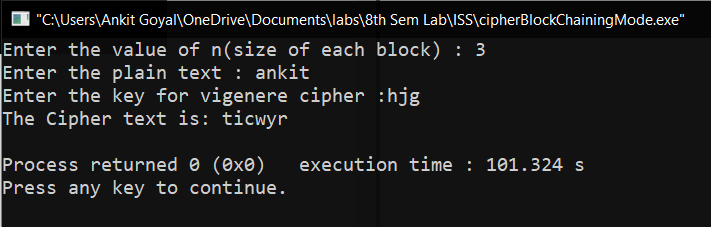
cipher.append(ci);

}

cout<<"The Cipher text is: "<<cipher<<"\n";

}

**Output:**



1. **Cipher Feedback (CFB) Mode**

**Code:**

#include<bits/stdc++.h>

using namespace std;

string xor\_operation(string a, string b)

{

string ans="";

int n=a.length();

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

{

char k= ((a[i]^b[i])%26 )+'a';

ans+=k;

}

return ans;

}

string generateKey(string key, int x)

{

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

{

if (x == i)

i = 0;

if (key.size() == x)

break;

key.push\_back(key[i]);

}

return key;

}

string cipherText(string str, string key)

{

string cipher\_text;

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

{

char x = (str[i] + key[i]) %26;

x += 'a';

cipher\_text.push\_back(x);

}

return cipher\_text;

}

int main()

{

int r;

cout<<"Enter the value of r(size of each block) : ";

cin>>r;

string plain, cipher="", S;

cout<<"Enter the plain text : ";

cin>>plain;

string key;

cout<<"Enter the key for vigenere cipher : ";

cin>>key;

cout<<"Enter the initial value of shift register : ";

cin>>S;

int n=S.length();

key = generateKey(key,n);

cout<<"\nKey : "<<key;

int blocks;

if(plain.length()%r!=0)

{

int k= (plain.length()/r) \* r;

int g= plain.length()-k;

g=r-g;

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

plain.append("z");

}

blocks= plain.length()/r;

cout<<"\nBlocks : "<<blocks;

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

{

string cip=cipherText(S,key);

cip= cip.substr(0,r);

string tp= plain.substr(i\*r, i\*r+r);

tp= xor\_operation(tp,cip);

S=S.substr(r, n);

S.append(tp);

cout<<"\nCipher : "<<tp<<" new S : "<<S<<"\n";

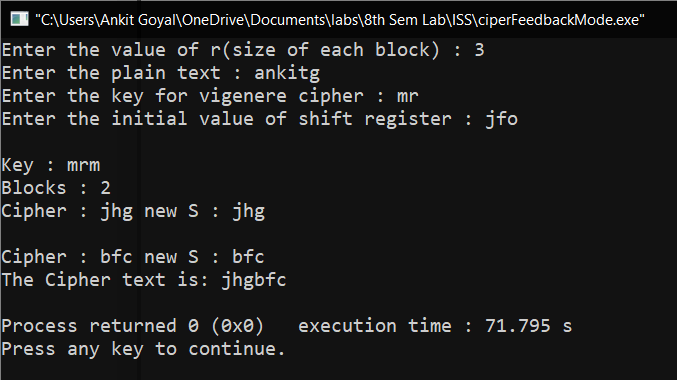
cipher.append(tp);

}

cout<<"The Cipher text is: "<<cipher<<"\n";

}

**Output:**



1. **Output Feedback (OFB) Mode**

**Code:**

#include<bits/stdc++.h>

using namespace std;

string xor\_operation(string a, string b)

{

string ans="";

int n=a.length();

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

{

char k= ((a[i]^b[i])%26 )+'a';

ans+=k;

}

return ans;

}

string generateKey(string key, int x)

{

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

{

if (x == i)

i = 0;

if (key.size() == x)

break;

key.push\_back(key[i]);

}

return key;

}

string cipherText(string str, string key)

{

string cipher\_text;

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

{

char x = (str[i] + key[i]) %26;

x += 'a';

cipher\_text.push\_back(x);

}

return cipher\_text;

}

int main()

{

int r;

cout<<"Enter the value of r(size of each block) : ";

cin>>r;

string plain, cipher="", S;

cout<<"Enter the plain text : ";

cin>>plain;

string key;

cout<<"Enter the key for vigenere cipher : ";

cin>>key;

cout<<"Enter the initial value of shift register : ";

cin>>S;

int n=S.length();

key = generateKey(key,n);

cout<<"\nKey : "<<key;

int blocks;

if(plain.length()%r!=0)

{

int k= (plain.length()/r) \* r;

int g= plain.length()-k;

g=r-g;

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

plain.append("z");

}

blocks= plain.length()/r;

cout<<"\nBlocks : "<<blocks;

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

{

string cip=cipherText(S,key);

cout<<"\nEncrypted Shift Register :"<<cip<<"\n";

cip= cip.substr(0,r);

string tp= plain.substr(i\*r, i\*r+r);

tp= xor\_operation(tp,cip);

S=S.substr(r, n);

S.append(cip);

cout<<"\nCipher : "<<tp<<" new S : "<<S<<"\n";

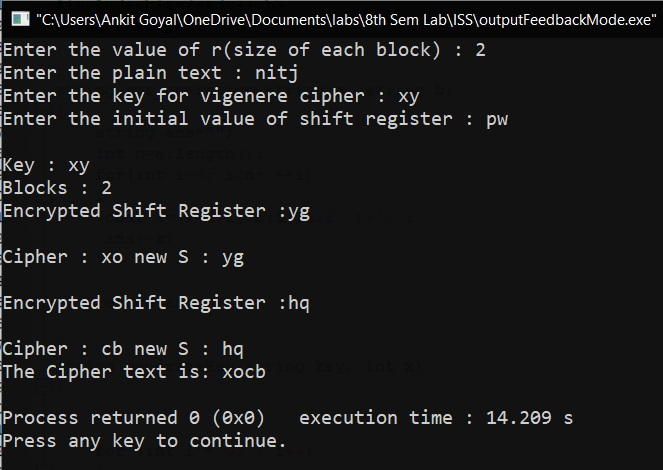
cipher.append(tp);

}

cout<<"The Cipher text is: "<<cipher<<"\n";

}

**Output:**



1. **Counter (CTR) Mode**

**Code:**

#include<bits/stdc++.h>

using namespace std;

string xor\_operation(string a, string b)

{

string ans="";

int n=a.length();

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

{

char k= ((a[i]^b[i])%26 )+'a';

ans+=k;

}

return ans;

}

string generateKey(string key, int x)

{

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

{

if (x == i)

i = 0;

if (key.size() == x)

break;

key.push\_back(key[i]);

}

return key;

}

string cipherText(string str, string key)

{

string cipher\_text;

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

{

char x = (str[i] + key[i]) %26;

x += 'a';

cipher\_text.push\_back(x);

}

return cipher\_text;

}

int main()

{

int n;

cout<<"Enter the value of n(size of each block) : ";

cin>>n;

string plain, cipher="";

cout<<"Enter the plain text : ";

cin>>plain;

string key;

cout<<"Enter the key for vigenere cipher :";

cin>>key;

key = generateKey(key,n);

int blocks;

if(plain.length()%n!=0)

{

int k= (plain.length()/n) \* n;

int g= plain.length()-k;

g=n-g;

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

plain.append("z");

}

blocks= plain.length()/n;

string counter(n,'0');

int count=0;

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

{

string x = to\_string(count);

counter = counter.substr(0,n-x.length())+x;

cout<<counter<<"\n";

string tp= plain.substr(i\*n, i\*n+n);

string ci= cipherText(counter,key);

tp= xor\_operation(tp,ci);

cipher.append(tp);

count++;

}

cout<<"The Cipher text is: "<<cipher<<"\n";

}

**Output:**

