VLSI Design and Testing Laboratory Manual

DEPARTMENTS OF COMPUTER SCIENCE AND ENGINEERING

NAME : VIVEK KUMAR
ADM. NO: 2012JE0739

SESSION : 2015-16

ADDITIVE CIPHER

CODE:

```
// Additive Cipher C Program
#include <stdio.h>
#include <string.h>
int main()
{
	// String input
	char a[35];
	printf("Enter the string: ");
	scanf("%s",a);
	// KEy input
	printf("Enter the additive key\n");
	int k;
	scanf("%d",&k);
	if(k < 0) {
		k += 26;
	}
	for(int i = 0; i < strlen(a); i++)
	{
		a[i] = (a[i] - 'A' + k) % 26 + 'A';
	}
	// Printing the encryption
	printf("After Encryption the string is : %s\n",a);
	return 0;
}
```

OUTPUT:

```
C:\Users\Mohit\Desktop\crypto programs>gcc additive_cipher.c

C:\Users\Mohit\Desktop\crypto programs>a
Enter the string: cryptography
Enter the additive key
12
After Encryption the string is : UJQHLGYJSHZQ
```

Figure: Output Additive Cipher

BRUTE FORCE ATTACK ON ADDITIVE CIPHER

CODE:

```
// Additive Cipher Brute Force Attack C Program
#include <iostream>
#include <stdio.h>

using namespace std;

int main() {
	string pt;
	string ct;
	cin >> pt;
	int k = 1;
	while(k < 26) {
		ct = "";
		int i;
		for(i = 0; i < pt.size(); i++) {
			char ch = (pt[i] - 'A' - k)%26;
			ch += 'A';
			if(ch < 'A') {
				ch += 26;
			}
			ct.push_back(ch);
		}
		cout << "String after taking key = " << k << " is " << ct << endl;
		k++;
	}
	return 0;

}
```

OUTPUT:

```
C:\Users\Mohit\Desktop\crypto programs>g++ brute_force_additive.cc

C:\Users\Mohit\Desktop\crypto programs>a
cryptography
String after taking key = 1 is HWDUYTLWFUMD
String after taking key = 2 is GVCTXSKVETLC
String after taking key = 3 is FUBSWRJUDSKB
String after taking key = 4 is ETARVQITCRJA
String after taking key = 5 is DSZQUPHSBQIZ
String after taking key = 6 is CRYPTOGRAPHY
String after taking key = 7 is BQXOSNFQZOGX
String after taking key = 8 is APWNRMEPYNFW
String after taking key = 9 is ZOVMQLDOXMEV
String after taking key = 10 is YNULPKCNWLDU
String after taking key = 11 is XMTKOJBMVKCT
String after taking key = 12 is WLSJNIALUJBS
String after taking key = 13 is VKRIMHZKTIAR
String after taking key = 14 is UJQHLGYJSHZQ
String after taking key = 15 is TIPGKFXIRGYP
String after taking key = 16 is SHOFJEWHQFXO
String after taking key = 17 is RGNEIDVGPEWN
String after taking key = 18 is QFMDHCUFODVM
String after taking key = 19 is PELCGBTENCUL
String after taking key = 20 is ODKBFASDMBTK
String after taking key = 21 is NCJAEZRCLASJ
String after taking key = 22 is MBIZDYQBKZRI
String after taking key = 23 is LAHYCXPAJYQH
String after taking key = 24 is KZGXBWOZIXPG
String after taking key = 25 is JYFWAVNYHWOF
```

Figure: Brute Force Attack on Additive Cipher(Key 6 makes sense)

STATISTICAL ATTACK ON ADDITIVE CIPHER

CODE:

```
// Additive Cipher Statistical Attack C Program
#include <stdio.h>
#include <limits.h>
#include<string.h>

int main() {
    char a[100] =
    "XLILSYWIMWRSAJSVWEPIJSVJSYVQMPPMSRHSPPEVWMXMWASVXLQSVILYVVCFIJSVIX
    LIWIPPIVVIGIMZIWQSVISJJIVW";
        int hash[26] = {0},k=0;
        char ct[100];
        int i;
        for(i = 0; i < strlen(a); i++) {
                hash[a[i] - 'A']++;
        }
        int maxIndex = INT_MIN, maxValue = INT_MIN;
        for(i = 0; i < 26; i++) {
                if(maxValue < hash[i]) {
                        maxValue = hash[i];
                        maxIndex = i;
                }
        }
        int b = maxIndex - 4;
        if(b < 0) {
                b += 26;
        }
        for(i = 0; i < strlen(a); i++) {
                char ch = (a[i] - 'A' - b) % 26 + 'A';
                if(ch < 'A') {
                        ch += 26;
                }
                ct[k++]=ch;
        }
       printf("Encrypted string is %s\n",ct);
       return 0;
}
```

OUTPUT:

```
C:\Users\Mohit\Desktop\crypto programs>gcc statistical_attack_additive.c

C:\Users\Mohit\Desktop\crypto programs>a
Applying Statistical Attack:
Encrypted string is THEHOUSEISNOWFORSALEFORFOURMILLIONDOLLARSITISWORTHMOREHURRYB
EFORETHESELLERRECEIVESMOREOFFERS
```

Figure: Statistical Attack on Additive Cipher

MULTIPLICATIVE CIPHER

CODE:

```
// Multiplicative Cipher C Program
#include <iostream>
#include <stdio.h>
using namespace std;

int main() {
	string pt;
	cout << "Enter the string to be encrypted: ";
	cin >> pt;
	cout << "Enter the key: ";
	int key;
	cin >> key;
	if(key % 13 == 0 || key % 2 == 0) {
		cout << "Not a valid Key";
	} else {
		for(int i = 0; i < pt.size(); i++) {
			pt[i] = ((pt[i] - 'A') * key) % 26 + 'A';
		}
		cout << "String after encryption: " << pt;
	}
	return 0;
}
```

OUTPUT:

```
C:\Users\Mohit\Desktop\crypto programs>g++ multiplicative_cipher.cc

C:\Users\Mohit\Desktop\crypto programs>a
Enter the string to be encrypted: sapphirehostel
Enter the key: 5
String after encryption: QEBBNSLYNWQUYH
C:\Users\Mohit\Desktop\crypto programs>
```

Figure: Output Multiplicative Cipher using key 5

AFFINE CIPHER

CODE:

```
// Affine Cipher C Program
#include <stdio.h>
#include <string.h>
int main()
{
        char str[30];
        int keym,keya,i;
        printf("Enter plaintext\n");
        scanf("%s",str);
        int len=strlen(str);
        printf("Enter the additive key\n");
        scanf("%d",&keya);
       printf("Enter the multiplicative key\n");
        scanf("%d",&keym);
        for(i=0;i<len;i++)
        {
                int t = str[i]-'a';
                t=(t*keym+keya);
                t=t%26;
                str[i]=(char)(t+'a');
        }
        printf("Cipher Text is \n%s\n",str);
        return 0;
}
```

OUTPUT:

```
C:\Users\Mohit\Desktop\crypto programs>a
Enter plaintext
mohitchawla
Enter the additive key
2
Enter the multiplicative key
5
Cipher Text is
kulqtmlcifc

C:\Users\Mohit\Desktop\crypto programs>
```

Figure: Output Affine Cipher

PLAYFAIR CIPHER

CODE:

```
// Playfair Cipher C Program
#include <stdio.h>
#include <conio.h>
#include <string.h>
int main() {
    char v,w,ch,string[100],arr[5][5],key[10],a,b,enc[100];
    int temp,i,j,k,l,r1,r2,c1,c2,t,var;
    FILE * fp;
    fp=fopen("playfair_ip.txt","r");
    //keep message in sk.txt (e.g. jamia)
    printf("Enter the key\n");
    fflush(stdin);
    scanf("%s",&key);
    l=0;
    while(1) {
        ch=fgetc(fp);
        if(ch!=EOF) {
            string[l++]=ch;
        }
        if(ch==EOF)
            break;
    }
    string[l]='\0';
    puts(string);
    for (i=0;key[i]!='\0';i++) {
        for (j=i+1;key[j]!='\0';j++) {
            if(key[i]==key[j]) {
                temp=1;
                break;
            }
        }
    }
    if(temp==1)
    printf("invalid key"); else {
        k=0;
        a='a';
        //printf("%c",b);
```

```
for (i=0;i<5;i++) {
    for (j=0;j<5;j++) {
        if(k<strlen(key))
            arr[i][j]=key[k]; else if(k==strlen(key)) {
            b:
                for (l=0;l<strlen(key);l++) {
                if(key[l]==a) {
                    a++;
                    goto b;
                }
            }
            arr[i][j]=a;
            if(a=='i')
                a=a+2; else
                a++;
        }
        if(k<strlen(key))
            k++;
    }
}
printf("\n");
printf("The matrix is\n");
for (i=0;i<5;i++) {
    for (j=0;j<5;j++) {
        printf("%c",arr[i][j]);
    }
    printf("\n");
}
t=0;
if(strlen(string)%2!=0)
var=strlen(string)-1;
for (i=0;i<var;) {
    v=string[i++];
    w=string[i++];
    if(v==w) {
        enc[t++]=v;
        enc[t++]='$';
    } else {
        for (l=0;l<5;l++) {
            for (k=0;k<5;k++) {
                if(arr[l][k]==v||v=='j'&&arr[l][k]=='i') {
                    r1=l;
```

```
                    c1=k;
                }
                if(arr[l][k]==w||w=='j'&&arr[l][k]=='i') {
                    r2=l;
                    c2=k;
                }
            }
        }
        if(c1==c2) {
            r1++;
            r2++;
            if(r1==5||r2==5) {
                r1=0;
                r2=0;
            }
        } else if(r1==r2) {
            c1++;
            c2++;
            if(c1==5||c2==5) {
                c1=0;
                c2=0;
            }
        } else {
            temp=r1;
            r1=r2;
            r2=temp;
        }
        enc[t++]=arr[r1][c1];
        enc[t++]=arr[r2][c2];
      }
    }
    if(strlen(string)%2!=0)
    enc[t++]=string[var];
    enc[t]='\0';
  }
  printf("The encrypted text is\n");
  puts(enc);
  return 0;
}
```

OUTPUT:

```
C:\Users\Mohit\Desktop\crypto programs>gcc playfair.c

C:\Users\Mohit\Desktop\crypto programs>a
Enter the key
1234
mohitchawla

The matrix is
1234a
bcdef
ghikl
mnopq
rstuv
The encrypted text is
npikds2lhaa
```

Figure: Output Playfair Cipher

VIGENERE CIPHER

CODE:

```
#include <iostream>
using namespace std;
int main() {
	string str, strOutput = "";
	cin >> str;
	int n;
	int array[26];
	int i;
	cout << "Enter private key size: ";
	cin >> n;
	cout << "Enter the private key: ";
	for(i = 0; i < n; i++) {
		cin >> array[i];
	}
	int k = 0;
	for(i = 0; i < str.size(); i++) {
		if(k % n == 0) {
			k = 0;
		}
		char ch = (str[i] + array[k++] - 'A') % 26 + 'A';
		strOutput.push_back(ch);
	}
	cout << "Cipher Text: " << strOutput << endl;
	return 0;
}
```

OUTPUT:

```
C:\Users\Mohit\Desktop\crypto programs>g++ vignere.cpp

C:\Users\Mohit\Desktop\crypto programs>a
mohitchawlalivesinsapphire
Enter private key size: 7
Enter the private key: mohi
Cipher Text: SEDSZWTGMHKRCHKIEXYUBVXEBK

C:\Users\Mohit\Desktop\crypto programs>
```

Figure: Output of Vigenere Cipher

HILL CIPHER

CODE:

```
#include <iostream>
using namespace std;
int matrix[4][4] = {{9, 7, 11, 13}, {4, 7, 5, 6}, {2, 21, 14, 9}, {3, 23, 21, 8}};
int decr[4][4] = {{2, 15, 22, 3}, {15, 0, 19, 3}, {9, 9, 3, 11}, {17, 0, 4, 7}};
int mult[100][4];
int createMatrix(string temp) {
	int cnt = 0, i;
	int row = 0;
	for(i = 0; i < temp.size(); i++) {
		mult[row][cnt % 4] = temp[i] - 'a';
		cnt++;
		if(cnt % 4 == 0) {
			row++;
		}
	}
	while(cnt % 4 != 0) {
		mult[row][cnt % 4] = 'z' - 'a';
		cnt++;
		if(cnt % 4 == 0)
			row++;
	}
	row--;
	return row;
}
string encrypt(string temp) {
	int i, j, k;
	string ret = "";
	int finalMat[100][4];
	int rows = createMatrix(temp);
```

```
        for(i = 0; i <= rows; i++) {
                for(j = 0; j < 4; j++) {
                        finalMat[i][j] = 0;
                        for(k = 0; k < 4; k++) {
                                finalMat[i][j] = (finalMat[i][j] + mult[i][k] * matrix[k][j]) % 26;
                        }
                }
        }
        for(i = 0; i <= rows; i++) {
                for(j = 0; j < 4; j++) {
                        ret.push_back('a' + finalMat[i][j]);
                }
        }
        return ret;
}

string decrypt(string temp1) {
        int i, j, k;
        string ret = "";
        int finalMat[100][4];
        int rows = createMatrix(temp1);
        for(i = 0; i <= rows; i++) {
                for(j = 0; j < 4; j++) {
                        finalMat[i][j] = 0;
                        for(k = 0; k < 4; k++) {
                                finalMat[i][j] = (finalMat[i][j] + mult[i][k] * decr[k][j]) % 26;
                        }
                }
        }
        for(i = 0; i <= rows; i++) {
                for(j = 0; j < 4; j++) {
                        ret.push_back('a' + finalMat[i][j]);
                }
        }
        return ret;
}


int main() {
        string str, strOutput = "", decrypted = "";
        cout << "Enter the string to be encrypted: ";
        cin >> str;
```

```
        strOutput = encrypt(str);
        cout << "Encrypted text " << strOutput << endl;
        decrypted = decrypt(strOutput);
        cout << "Decrypted text " << decrypted;
        return 0;
}
```

OUTPUT:

```
C:\Users\Mohit\Desktop\crypto programs>g++ hill_cipher.cpp

C:\Users\Mohit\Desktop\crypto programs>a
Enter the string to be encrypted: mohitchawla
Encrypted text utadlifkfaqg
Decrypted text mohitchawlaz
C:\Users\Mohit\Desktop\crypto programs>
```

Figure: Output of Hill Cipher

DES

CODE:

```
// DES C Program
#include <stdio.h>
void left_shift(int *a)
{
	int i,tmp=a[0];
	for(i=0;i<27;i++)
		a[i]=a[i+1];
	a[27]=tmp;
}
int main()
{
	int i,j;
	int
p[64]={0,0,0,1,0,0,1,0,0,0,1,1,0,1,0,0,0,1,0,1,0,1,1,0,1,0,1,0,1,0,1,1,1,1,0,0,1,1,0,1,0,0,0,1,0,0,1,1,0,0,
1,0,0,1,0,1,0,0,1,1,0,1,1,0};
	int
key[64]={1,0,1,0,1,0,1,0,1,0,1,1,1,0,1,1,0,0,0,0,1,0,0,1,0,0,0,1,1,0,0,0,0,0,1,0,0,1,1,1,0,0,1,1,0,1,1,0,1
,1,0,0,1,1,0,0,1,1,0,1,1,1,0,1};
```

```
	//parity drop
	int parity_drop[8][7]={{57,49,41,33,25,17,9},
				{1,58,50,42,34,26,18},
				{10,2,59,51,43,35,27},
				{19,11,3,60,52,44,36},
				{63,55,47,39,31,23,15},
				{7,62,54,46,38,30,22},
				{14,6,61,53,45,37,29},
				{21,13,5,28,20,12,4}};
	int key_comp[8][6]={{14,17,11,24,1,5},
				{3,28,15,6,21,10},
				{23,19,12,4,26,8},
				{16,7,27,20,13,2},
				{41,52,31,37,47,55},
				{30,40,51,45,33,48},
				{44,49,39,56,34,53},
				{46,42,50,36,29,32}};
	int key_parity[56];
	int z=0;
	for(i=0;i<8;i++)
	{
		for(j=0;j<7;j++)
		key_parity[z++]=key[parity_drop[i][j]-1];
	}
//	for(i=0;i<56;i++)
//	printf("%d",key_parity[i]);
	int k1[28],k2[28];
	for(i=0;i<56;i++)
	{
		if(i<28)
			k1[i]=key_parity[i];
		else
			k2[i-28]=key_parity[i];
	}
	int round=1,shift;
	while(round<=16)
	{
		printf("key for round %d:\n",round);
		if(round==1 || round==2 || round==9 ||round==16)
			shift=1;
		else
			shift=2;
```

```
            while(shift--)
            {
                  left_shift(k1);
                  left_shift(k2);
            }
            for(i=0;i<8;i++)
            {
                  for(j=0;j<6;j++)
                  {
                        if(key_comp[i][j]<=28)
                              printf("%d",k1[key_comp[i][j]-1]);
                        else
                              printf("%d",k2[key_comp[i][j]-29]);
                  }
            }
            printf("\n");
            round++;
      }
return 0;
}
```

OUTPUT:

```
C:\Users\Mohit\Desktop\crypto programs>gcc des.c

C:\Users\Mohit\Desktop\crypto programs>a
key for round 1:
000110010100110011010000011100101101111010001100
key for round 2:
010001010110100001011000000011010101111001100111O
key for round 3:
000001101111011011010010010101100111101011011O101
key for round 4:
110110100010110100000011001010110110111011100011
key for round 5:
011010011010011000101001111111110110010010001O011
key for round 6:
110000011001010010001110100001110100011101011110
key for round 7:
011100001000101011010010110111011011000111100000O
key for round 8:
001101001111110000010001011110000110001100110110l
key for round 9:
100001001011101101000100011100111101110011001100
key for round 10:
000000100111011001010111000010001011010110111111
key for round 11:
011011010101010110000010101111011111001010010l
key for round 12:
110000101100000011110100101101010010010111111001l
key for round 13:
100110011100001100010011100101111100100100011111
key for round 14:
001001010001101110001011110001110001011111010000
key for round 15:
001100110011000011000101110110011010001101101101
key for round 16:
000110000000111000101110101110101110001100110l101
```

Figure: Output DES

AES

CODE:

```
// AES C Program
#include <stdio.h>
#include <stdlib.h>
int sbox[256] =   {
        //0     1   2     3     4   5     6     7      8   9     A      B   C     D     E     F
        0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5, 0x30, 0x01, 0x67, 0x2b, 0xfe, 0xd7, 0xab,
0x76, //0
        0xca, 0x82, 0xc9, 0x7d, 0xfa, 0x59, 0x47, 0xf0, 0xad, 0xd4, 0xa2, 0xaf, 0x9c, 0xa4, 0x72,
0xc0, //1
        0xb7, 0xfd, 0x93, 0x26, 0x36, 0x3f, 0xf7, 0xcc, 0x34, 0xa5, 0xe5, 0xf1, 0x71, 0xd8, 0x31,
0x15, //2
        0x04, 0xc7, 0x23, 0xc3, 0x18, 0x96, 0x05, 0x9a, 0x07, 0x12, 0x80, 0xe2, 0xeb, 0x27, 0xb2,
0x75, //3
        0x09, 0x83, 0x2c, 0x1a, 0x1b, 0x6e, 0x5a, 0xa0, 0x52, 0x3b, 0xd6, 0xb3, 0x29, 0xe3, 0x2f,
0x84, //4
        0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1, 0x5b, 0x6a, 0xcb, 0xbe, 0x39, 0x4a, 0x4c, 0x58,
0xcf, //5
        0xd0, 0xef, 0xaa, 0xfb, 0x43, 0x4d, 0x33, 0x85, 0x45, 0xf9, 0x02, 0x7f, 0x50, 0x3c, 0x9f, 0xa8,
//6
        0x51, 0xa3, 0x40, 0x8f, 0x92, 0x9d, 0x38, 0xf5, 0xbc, 0xb6, 0xda, 0x21, 0x10, 0xff, 0xf3, 0xd2,
//7
        0xcd, 0x0c, 0x13, 0xec, 0x5f, 0x97, 0x44, 0x17, 0xc4, 0xa7, 0x7e, 0x3d, 0x64, 0x5d, 0x19,
0x73, //8
        0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90, 0x88, 0x46, 0xee, 0xb8, 0x14, 0xde, 0x5e, 0x0b,
0xdb, //9
        0xe0, 0x32, 0x3a, 0x0a, 0x49, 0x06, 0x24, 0x5c, 0xc2, 0xd3, 0xac, 0x62, 0x91, 0x95, 0xe4,
0x79, //A
        0xe7, 0xc8, 0x37, 0x6d, 0x8d, 0xd5, 0x4e, 0xa9, 0x6c, 0x56, 0xf4, 0xea, 0x65, 0x7a, 0xae,
0x08, //B
        0xba, 0x78, 0x25, 0x2e, 0x1c, 0xa6, 0xb4, 0xc6, 0xe8, 0xdd, 0x74, 0x1f, 0x4b, 0xbd, 0x8b,
0x8a, //C
        0x70, 0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6, 0x0e, 0x61, 0x35, 0x57, 0xb9, 0x86, 0xc1, 0x1d,
0x9e, //D
        0xe1, 0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e, 0x94, 0x9b, 0x1e, 0x87, 0xe9, 0xce, 0x55, 0x28,
0xdf, //E
        0x8c, 0xa1, 0x89, 0x0d, 0xbf, 0xe6, 0x42, 0x68, 0x41, 0x99, 0x2d, 0x0f, 0xb0, 0x54, 0xbb,
0x16 }; //F

int round[10]={0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80,0x1b,0x36};
```

```
int find(int a)
{
	return sbox[a];
}

int * calculate_delta(int *key,int r)
{
	int *t = (int *)malloc(4*sizeof(int));
	t[0]=key[13];
	t[1]=key[14];
	t[2]=key[15];
	t[3]=key[12];

	t[0]=find(t[0])^round[r];
	t[1]=find(t[1]);
	t[2]=find(t[2]);
	t[3]=find(t[3]);

	return t;
}

int *find_xor(int *a,int s,int *b,int s1)
{
	int *t = (int *)malloc(4*sizeof(int));
	int i;
	for(i=0;i<4;i++)
	{
		t[i]=a[s+i]^b[s1+i];
	}
	return t;
}

void overwrite(int *a,int s,int *b)
{
	int i;
	for(i=0;i<4;i++)
		a[s+i]=b[i];
}

void fn(int *key)
{
	int i;
```

```
	int *temp = (int *)malloc(4*sizeof(int));
	for(i=0;i<10;i++)
	{
		int *delta = calculate_delta(key,i);
		int *temp = find_xor(key,0,delta,0);
		overwrite(key,0,temp);
		temp = find_xor(key,0,key,4);
		overwrite(key,4,temp);
		temp = find_xor(key,4,key,8);
		overwrite(key,8,temp);
		temp = find_xor(key,8,key,12);
		overwrite(key,12,temp);

		int j;
		char strr[100];
		printf("KEY %d : " ,i+1);
		for(j=0;j<16;j++)
		{
			itoa(key[j],strr,16);
			printf("%s ",strr);
		}
		printf("\n");
	}
}

int main()
{
	int key[]={0x24, 0x75 ,0xA2 ,0xB3 ,0x34 ,0x75 ,0x56 ,0x88 ,0x31 ,0xE2 ,0x12 ,0x00 ,0x13 ,
0xAA ,0x54 ,0x87};
	fn(key);
	return 0;
}
```

OUTPUT:

```
C:\Users\Mohit\Desktop\crypto programs>a
KEY 1 : 89 55 b5 ce bd 20 e3 46 8c c2 f1 46 9f 68 a5 c1
KEY 2 : ce 53 cd 15 73 73 2e 53 ff b1 df 15 60 d9 7a d4
KEY 3 : ff 89 85 c5 8c fa ab 96 73 4b 74 83 13 92 e 57
KEY 4 : b8 22 de b8 34 d8 75 2e 47 93 1 ad 54 1 f fa
KEY 5 : d4 54 f3 98 e0 8c 86 b6 a7 1f 87 1b f3 1e 88 e1
KEY 6 : 86 90 b 95 66 1c 8d 23 c1 3 a 38 32 1d 82 d9
KEY 7 : 62 83 3e b6 4 9f b3 95 c5 9c b9 ad f7 81 3b 74
KEY 8 : ee 61 ac de ea fe 1f 4b 2f 62 a6 e6 d8 e3 9d 92
KEY 9 : e4 3f e3 bf e c1 fc f4 21 a3 5a 12 f9 40 c7 80
KEY 10 : db f9 2e 26 d5 38 d2 d2 f4 9b 88 c0 d db 4f 40
```

Figure: Output AES

RSA

CODE:

```
#include <stdio.h>
int powmod(int n,int m,int d)
{
    int ans=1;
    while(d--)
        ans=(ans*m)%n;
    return ans;
}
int main()
{
    int p,q,n,d,e;
    printf("Enter the value of p:\n");
    scanf("%d",&p);
    printf("Enter the value of q:\n");
    scanf("%d",&q);
    printf("Enter the value of d:\n");
    scanf("%d",&d);
    printf("Enter the value of e:\n");
    scanf("%d",&e);
    n=p*q;
    int phi=(p-1)*(q-1);
    int m=46;
    m=m%n;
    int s=powmod(n,m,d);
    int v=powmod(n,s,e);

    if(v==m)
        printf("message is accepted\n");
    else
        printf("message is rejected\n");
    return 0;
}
```

OUTPUT:

```
C:\Users\Mohit\Desktop\crypto programs>gcc rsa_digital.c

C:\Users\Mohit\Desktop\crypto programs>A
Enter the value of p:
5
Enter the value of q:
7
Enter the value of d:
11
Enter the value of e:
17
message is accepted
```

Figure: Output RSA

CHOSEN CIPHERTEXT ATTACK ON RSA

CODE:

```
#include <bits/stdc++.h>
using namespace std;

bool isPrime(int a) {
	for (int i = 2; i * i <= a; ++i) {
		if (a % i == 0) return false;
	}
	return true;
}

vector<int> getMeTuple(int a) {
	vector<int> primes (2, 0);
	for (int i = 2; i <= a; ++i) {
		if (a % i == 0 && isPrime(i) && isPrime(a / i)) {
			primes[0] = i;
			primes[1] = a / i;
		}
	}
	return primes;
}

int fastExpo(int a, int b, int MOD) {
	if (b == 0) return 1;
	int result = fastExpo(a, b >> 1, MOD);
	result = (result * result) % MOD;
```

```
        if (b & 1) result = (result * (a % MOD)) % MOD;
        return result;
}

int inverse(int e, int MOD) {
        int result = 0;
        while (true) {
                if ((e * result) % MOD == 1) return result;
                ++result;
        }
        return result;
}

int main() {
        int p, e, n, c;
        cin >> p >> c >> e >> n;
        vector<int> myPQ = getMeTuple(n);
 int d = inverse(e, (myPQ[0] - 1) * (myPQ[1] - 1));
        int decrypted = fastExpo(c, d, n);
        cout << (decrypted == p) << endl;
        cout << decrypted << endl;
        return 0;
}
```

OUTPUT:

```
C:\Users\Mohit\Desktop\crypto programs>gcc m.c

C:\Users\Mohit\Desktop\crypto programs>a
Enter the c,e,n
9 2 9979
Plain text is 4
C:\Users\Mohit\Desktop\crypto programs>
```

Figure: Chosen Ciphertext Attack on RSA

COMMON MODULUS ATTACK ON RSA

CODE:

```
#include <iostream>
```

```
using namespace std;
long long int e, f;
long long int n;

long long int cffs, cffb;

void calcInverse(long long int a, long long int b) {
	long long int newa = b;
	long long int newb = a % b;
	if(newa % newb == 0) {
		cffs = (a / b) * (-1);
		cffb = 1;
		//cout << cffb << " " << cffs << endl;
		return;
	}
	calcInverse(b, a % b);
	long long int store = cffs;
	cffs = cffb - (a / b) * (cffs);
	cffb = store;
	//cout << cffb << " " << cffs << endl;

}
long long int power(long long int plaintext, long long int e) {

	long long int ans = 1;
	while(e) {
		if(e & 1) {
			ans = (ans * plaintext) % n;
		}
		plaintext = (plaintext * plaintext) % n;
		e = e >> 1;
		//cout << plaintext;
	}
	return ans;
}
long long int convertToNum(string str) {
	int i;
	long long int ans = 0;
	for(i = 0; i < str.size(); i++) {
		ans = (ans * 26 + str[i] - 65) % n;
	}
	return ans;
```

```
}
int main() {
	cout<<"Enter common modulus"<<endl;
	cin >> n;
	cout<<"Enter public key1:"<<endl;
	cin >> e;
	cout<<"Enter public key2"<<endl;
	cin >> f;

	string cipher1;
	string cipher2;
	string ans = "";
	long long cipherNum1, cipherNum2;
	long long int plaintext;
	cout<<"Enter cipher text for key 1:"<<endl;
	cin >> cipher1;
	cout<<"Enter cipher text for key 2:"<<endl;
	cin >> cipher2;
	cipherNum1 = convertToNum(cipher1);
	cipherNum2 = convertToNum(cipher2);
	//cout << cipherNum1 << " " << cipherNum2 << endl;
	long long int x, y;
	if(e < f) {
		calcInverse(f, e);
		y = cffb;
		x = cffs;
	//	cout << x << " " << y << endl;
		/*x = x % n;
		y = y % n;
		if(x < 0) {
			x += f;
		}
		if(y < 0) {
			y += e;
		}*/
	} else {
		calcInverse(e, f);
		y = cffs;
		x = cffb;
		/*x = x % f;
		y = y % e;
		if(x < 0) {
```

```
                x += f;
            }
            if(y < 0) {
                y += e;
            }*/
        }
        //cout << x << " " << y << endl;
        //cout << power(cipherNum1, x) << " " << power(cipherNum2, y) << endl;
        if(y < 0) {
            calcInverse(n, cipherNum2);
            //cout << cffs << endl;
            plaintext = (power(cipherNum1, x) * power(cffs, -y)) % n;
        } else if(x < 0) {
            calcInverse(n, cipherNum1);
            //cout << cffs << endl;
            plaintext = (power(cffs, -x) * power(cipherNum2, y)) % n;
        }
        //plaintext = (power(cipherNum1, x) * power(cipherNum2, y)) % n;
        while(plaintext) {
            int rem = plaintext % 26;
            string s = "";
            s.push_back(rem + 65);
            ans = s + ans;
            plaintext /= 26;
        }
        cout<< "The plaintext is:"<<endl;
        cout << ans << endl;
        return 0;
}
```

OUTPUT:

```
C:\Users\Mohit\Desktop\crypto programs>g++ common_modulus.cpp

C:\Users\Mohit\Desktop\crypto programs>a
Enter common modulus
352211
Enter public key1:
491
Enter public key2
509
Enter cipher text for key 1:
RRMN
Enter cipher text for key 2:
QAUE
The plaintext is:
COMP

C:\Users\Mohit\Desktop\crypto programs>
```

Figure: Common Modulus Attack on RSA

ELGAMAL DIGITAL SIGNATURE

CODE:

```
#include<stdio.h>
int powmod(int a,int b,int m)
{
	int ans=1;
	int d=a;
	while(b)
	{
		if(b&1)
			ans=(ans*d)%m;
		d=(d*d)%m;
		b>>=1;
	}
	return ans;
}
int premitive_root(int p)
{
	int i,a;
	for(a=2;a<p;a++)
	{
		for(i=2;i<p;i++)
		{
			if(powmod(a,i,p)==1 && i==p-1)
				return a;
			else if(powmod(a,i,p)==1 && i!=p-1)
```

```
                                        break;
                                }
                        }
}
int inverse(int a,int b)
{
        int inv=1;
        while((inv*b-1)%a!=0)
                inv++;
        return inv;
}
int main()
{
        int m,p,d,r;
        //User enter a large prime number, say 123
        printf("Enter message:\n");
        scanf("%d",&m);
        //Simulation: 3119,127,307
        printf("enter p,d and r:\n");
        scanf("%d %d %d",&p,&d,&r);

//      int e1=2;
//      printf("debug");
        int e1=premitive_root(p);
        int e2=powmod(e1,d,p);
//      printf("e1=%d e2=%d",e1,e2);
        // signature generation
        int s1=powmod(e1,r,p);
        int cal1=(m-d*s1);
        cal1=cal1%(p-1)+p-1;
        cal1=cal1%(p-1);
        int cal2=inverse(p-1,r);
        int s2=(cal1*cal2)%(p-1);

        // signature verification
        int v1= powmod(e1,m,p);
        int v2= (powmod(e2,s1,p)*powmod(s1,s2,p))%p;
//      printf("ca1=%d cal2=%d\nv1=%d v2=%d\ns1=%d s2=%d\n",cal1,cal2,v1,v2,s1,s2);
        printf("Using Elgaml Digital Signature Technique... \nVerfying Signature... \n");
        if(v1==v2)
                printf("signature verified\n");
        else
```

```
                printf("signature rejected\n");
        return 0;
}
```

OUTPUT:

```
C:\Users\Mohit\Desktop\crypto programs>gcc elgamal_digital_sign.c

C:\Users\Mohit\Desktop\crypto programs>a
Enter message:
123
enter p,d and r:
3119 127 307
Using Elgaml Digital Signature Technique...
Verfying Signature...
signature verified
```

Figure: El Gamal Digital Signature

SHA

CODE:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
/*
Output should be:
a9993e364706816aba3e25717850c26c9cd0d89d
84983e441c3bd26ebaae4aa1f95129e5e54670f1
34aa973cd4c4daa4f61eeb2bdbad27316534016f
*/

// Signed variables are for wimps
#define uchar unsigned char
#define uint unsigned int

// DBL_INT_ADD treats two unsigned ints a and b as one 64-bit integer and adds c to it

//Define left rotation with wrapping
#define ROTLEFT(a,b) ((a << b) | (a >> (32-b)))
//Define addition modulo 2^w
#define DBL_INT_ADD(a,b,c) if (a > 0xffffffff - c) ++b; a += c;
```

```
typedef struct {
   uchar data[64];
   uint datalen;
   uint bitlen[2];
   uint state[5];
   uint k[4];
} SHA1_CTX;


void sha1_transform(SHA1_CTX *ctx, uchar data[])
{
   uint a,b,c,d,e,i,j,t,m[80];

   for (i=0,j=0; i < 16; ++i, j += 4)
      m[i] = (data[j] << 24) + (data[j+1] << 16) + (data[j+2] << 8) + (data[j+3]);
   for ( ; i < 80; ++i) {
      m[i] = (m[i-3] ^ m[i-8] ^ m[i-14] ^ m[i-16]);
      m[i] = (m[i] << 1) | (m[i] >> 31);
   }

   //Define the 5 variables
   a = ctx->state[0];
   b = ctx->state[1];
   c = ctx->state[2];
   d = ctx->state[3];
   e = ctx->state[4];

   for (i=0; i < 20; ++i) {
      t = ROTLEFT(a,5) + ((b & c) ^ (~b & d)) + e + ctx->k[0] + m[i];
      e = d;
      d = c;
      c = ROTLEFT(b,30);
      b = a;
      a = t;
   }
   for ( ; i < 40; ++i) {
      t = ROTLEFT(a,5) + (b ^ c ^ d) + e + ctx->k[1] + m[i];
      e = d;
      d = c;
      c = ROTLEFT(b,30);
```

```
      b = a;
      a = t;
    }
    for ( ; i < 60; ++i) {
      t = ROTLEFT(a,5) + ((b & c) ^ (b & d) ^ (c & d))  + e + ctx->k[2] + m[i];
      e = d;
      d = c;
      c = ROTLEFT(b,30);
      b = a;
      a = t;
    }
    for ( ; i < 80; ++i) {
      t = ROTLEFT(a,5) + (b ^ c ^ d) + e + ctx->k[3] + m[i];
      e = d;
      d = c;
      c = ROTLEFT(b,30);
      b = a;
      a = t;
    }

    ctx->state[0] += a;
    ctx->state[1] += b;
    ctx->state[2] += c;
    ctx->state[3] += d;
    ctx->state[4] += e;
}

void sha1_init(SHA1_CTX *ctx)
{
    ctx->datalen = 0;
    ctx->bitlen[0] = 0;
    ctx->bitlen[1] = 0;
    ctx->state[0] = 0x67452301;
    ctx->state[1] = 0xEFCDAB89;
    ctx->state[2] = 0x98BADCFE;
    ctx->state[3] = 0x10325476;
    ctx->state[4] = 0xc3d2e1f0;
    ctx->k[0] = 0x5a827999;
    ctx->k[1] = 0x6ed9eba1;
    ctx->k[2] = 0x8f1bbcdc;
    ctx->k[3] = 0xca62c1d6;
}
```

```
void sha1_update(SHA1_CTX *ctx, uchar data[], uint len)
{
  uint t,i;

  for (i=0; i < len; ++i) {
    ctx->data[ctx->datalen] = data[i];
    ctx->datalen++;
    if (ctx->datalen == 64) {
      sha1_transform(ctx,ctx->data);
      DBL_INT_ADD(ctx->bitlen[0],ctx->bitlen[1],512);
      ctx->datalen = 0;
    }
  }
}

void sha1_final(SHA1_CTX *ctx, uchar hash[])
{
  uint i;

  i = ctx->datalen;

  // Pad whatever data is left in the buffer.
  if (ctx->datalen < 56) {
    ctx->data[i++] = 0x80;
    while (i < 56)
      ctx->data[i++] = 0x00;
  }
  else {
    ctx->data[i++] = 0x80;
    while (i < 64)
      ctx->data[i++] = 0x00;
    sha1_transform(ctx,ctx->data);
    memset(ctx->data,0,56);
  }

  // Append to the padding the total message's length in bits and transform.
  DBL_INT_ADD(ctx->bitlen[0],ctx->bitlen[1],8 * ctx->datalen);
  ctx->data[63] = ctx->bitlen[0];
  ctx->data[62] = ctx->bitlen[0] >> 8;
  ctx->data[61] = ctx->bitlen[0] >> 16;
  ctx->data[60] = ctx->bitlen[0] >> 24;
  ctx->data[59] = ctx->bitlen[1];
```

```
  ctx->data[58] = ctx->bitlen[1] >> 8;
  ctx->data[57] = ctx->bitlen[1] >> 16;
  ctx->data[56] = ctx->bitlen[1] >> 24;
  sha1_transform(ctx,ctx->data);

  // Since this implementation uses little endian byte ordering and MD uses big endian,
  // reverse all the bytes when copying the final state to the output hash.
  for (i=0; i < 4; ++i) {
    hash[i]    = (ctx->state[0] >> (24-i*8)) & 0x000000ff;
    hash[i+4]  = (ctx->state[1] >> (24-i*8)) & 0x000000ff;
    hash[i+8]  = (ctx->state[2] >> (24-i*8)) & 0x000000ff;
    hash[i+12] = (ctx->state[3] >> (24-i*8)) & 0x000000ff;
    hash[i+16] = (ctx->state[4] >> (24-i*8)) & 0x000000ff;
  }


}




void print_hash(unsigned char hash[])
{
  int idx;
  for (idx=0; idx < 20; idx++)
    printf("%02x",hash[idx]);
  printf("\n");
}

int main()
{
  unsigned char text1[]={"abc"},
            text2[]={"abcdbcdecdefdefgefghfghighijhijkijkljklmklmnlmnomnopnopq"},
            text3[]={"aaaaaaaaaa"},
            hash[20];
  int idx;
  SHA1_CTX ctx;

  printf("Printing final hash output: \n\n");
  // Hash one
  sha1_init(&ctx);
  sha1_update(&ctx,text1,strlen(text1));
```

```
    sha1_final(&ctx,hash);
    print_hash(hash);

    // Hash two
    sha1_init(&ctx);
    sha1_update(&ctx,text2,strlen(text2));
    sha1_final(&ctx,hash);
    print_hash(hash);

    // Hash three
    sha1_init(&ctx);
    for (idx=0; idx < 100000; ++idx)
      sha1_update(&ctx,text3,strlen(text3));
    sha1_final(&ctx,hash);
    print_hash(hash);

    getchar();
    return 0;
  }
```

OUTPUT:

```
C:\Users\Mohit\Desktop\crypto programs>gcc SHA1_mohit.c

C:\Users\Mohit\Desktop\crypto programs>a
Printing final hash output:

a9993e364706816aba3e25717850c26c9cd0d89d
84983e441c3bd26ebaae4aa1f95129e5e54670f1
34aa973cd4c4daa4f61eeb2bdbad27316534016f
```

Figure: SHA

ELLIPTIC CURVE CRYPTOSYSTEM

CODE:

```
//El Gamal Cryptosystem
#include<stdio.h>
int powmod(int a,int b,int m)
{
        int ans=1;
```

```
	int d=a;
	while(b)
	{
		if(b&1)
			ans=(ans*d)%m;
		d=(d*d)%m;
		b>>=1;
	}
	return ans;
}
int premitive_root(int p)
{
	int i,a;
	for(a=2;a<p;a++)
	{
		for(i=2;i<p;i++)
		{
			if(powmod(a,i,p)==1)
				return a;
		}
	}
}
int main()
{
	printf("Welcome to elgamal cryptosystem\n");
	int m,p;
	//User enter a large prime number, say 10007
	printf("Enter message:\n");
	//USer enters a msg in form of a number, say 123
	scanf("%d",&m);
	printf("enter p:\n");
	scanf("%d",&p);
	int e1=premitive_root(p);
	int d=p/2,r=7;
	int e2=powmod(e1,d,p);
	int c1=powmod(e1,r,p);
	int c2=(m%p*powmod(e2,r,p))%p;

	int decrypt=(c2*powmod(powmod(c1,d,p),p-2,p))%p;
	printf("decrypted message is: %d\n",decrypt);
	return 0;
}
```

OUTPUT:

```
C:\Users\Mohit\Desktop\crypto programs>gcc elgamal_crypto.c

C:\Users\Mohit\Desktop\crypto programs>a
Welcome to elgamal cryptosystem
Enter message:
123
enter p:
10007
decrypted message is: 123

C:\Users\Mohit\Desktop\crypto programs>
```

Figure: Output El Gamal Cryptosystem