CRYPTOGRAPHY AND NETWORK SECURITY

LAB PROGRAMS

## Write a C program for Caesar cipher involves replacing each lefier of the alphabet with the letter standing k places further down the alphabet, for k in the range 1 through 25.

###### PROGRAM:

sinclude <stdio.h> sinclude <string.h»

void caesarEncrypt(char text[], int key) (

int lengh = strien(text);

for (int i = 0; i < length; i++) ( if(text[i] >= 'A' &@ tEXt[l] <= 'Z') (

text|i] = (text[i] - 'A' + key) 96 26 + 'A';

else if(text[i] >= 'a' && text[i] <='z') (

text|i] = (text[i] - 'a' + ley) 46 26 + 'a':

int main(j (

char message[100]:

int key;

printf("Enter a message: “);

fgets(message, sizeof(message), stdin);

printf("Enter the key (0-25):”);

scanf(“8d“, &Iey);

## @t. Scanned with OKEN Scanner

if(key < 0 || key > 2S)

printf("Invalid key! Please enter a key between 0 and 25.\n");



size t length = strIen(message);

if(length » 0 && message[length - 1]  ’\n’) ( message[length - 1] = ’\0",

caesarEncrypt(message, key); prinh(°Encrypted message: 46s\n", message); return 0;

OUTPUT:

Enter a message: cryptography Enter thR key (0-25): 4 Encrypted message: gvctxskvetlc

#### Write a C program for monoalphabet1c subst1tut1on ciphRr maps a plalntext

alphabet to a ciphertext alphabet, so that each IeRer of the plaintE:xt alphabet maps to a single unique leber of the ciphertext alphabet.

PROGRAM:

#include <stdio.h> #incIude <strfng.h> #include cctype.h>

void monoalphabeticSubstitution(char \*plaintext, char ”ciphertext, char “key} (

int i;

int len = strlen(plaintext);

for (i  0; i < len; i++) (

if(isalpha(plaintext[i])) {

char currentChar = tolower(pIaintext[i]);

#### Scanned with OKEN Scanner

int index = currentChar • 'a’;

ciphertext[i] = isupper(plaintext[i]) ? toupper(key[index]): key[index];

) else (

ciphertext|i] = plaintextjlj;

ciphertext|i] = ’\O';

int main()

char plaintext[I0o]; char ciphertext|10O];

char key[] = “QWERTYUIOPASDFGHJKLZXCVBNM";

printf("Enter the plaintext: "); fgets(plaintext, siZeof(plaintext), Stdin); plaintext[strcspn(plaintext, '/n")] = '\0';

monoaIphabeñcSubsñtvtJon(pIaIrtext, ciphe<ext, key);

printf("Ciphertex: 4ts\n", cipheyext);

return 0:

OUTPUT:

Enter the plaintext: hello Ciphertext: ITSSG

## Write a C program for Playfair algorithm is based on the use of a 5 X 5 matrix of letters constructed using a keyword. Plaintext is encrypted two IeRers at a time using this matrix.

PROGRAM:

## Scanned with OKEN Scanner

#include <stdio,h> #incIude <string.h> #include <ctype.h>

#define SIZE 5

void prepareKey(char key[], char matrix[SIZ6][SIZE]| (

int i, j, k  0;

int isPresent[26] = (0};

for (i  0; i < SIZE; i++) ( for (j  0; j < SIZE; j++) (

if (k < strIen(key)) (

if (IisPresent[key[k] - 'A’]) ( matrix[i]|j) = key|k]; isPresent[key[k] - 'A'] = 1; k++;

) else (

) else (

break;

for (i = 0; i < SIZE; i++) j

for (j = 0; j < SIZE: j++) (

if (matrix[i]|jj  '\0')

*lor* (k = 0; k « 26; k••) (

if (IisPresent|k]) (

## Scanned with OKEN Scanner

matrix[i]ÿ]  k + ’A", isPresent[k]  1; break;

void findPosition(char matrix|5IZE]|5IZE], char ch, int 'row, int ”col) (

if(ch  'J') // Treat 'J' as’I'

ch = ’Iï

for (”row= 0; row < SIZE; (’row)++) ( for (’col  0; ”col < SIZE; (’col)++) (

if (matrix['row][\*col] == ch) (

return;

void encryptPaIr(rhar matrix[SIZE}[SIZE], char ch1, char ch2, char encryptedPair{2]) (



ÈndPosiùon(matrix, ch1, &row1, &col1I:

findPosition(matrix, ch2, &row2, &col2);

Scanned with OKEN Scanner

if(row1  row2) (

encryptedPair[0] = matrix|row1][(coil + 1) B SIZE];

encryptedPaIr[1] = matrlx|row2][(col2 + 1) % SIZE];

) else if (coil coI2) {

encryptedPair|0] = matrix|(row1 + 1) 96 SIZE][coll];

encryptedPair[1] = matrix|(row2 + 1) 46 SIZE][coI2];

) else (

encryptedPair|o] = matrix|row1][col2];

encryptedPair|1] = matrix|row2][colt];

void encryptPlayfair(char matrix[SIZE](SIZE], char text[], char encryptedText[]) (

int i,Iength = strlen(text);

for ( i  0; i < length; i  2) {

char ch1 = toupper(text[i]);

char ch2  {i + 1 < length) ? toupper(text[i + 1]):'X’; char encryptedPalr[2];

encryptPair(matrix ch1, ch2, encryptedPair); encryptedText[i] encryptedPair[0]; encryptedText|i + 1] = encryptedPair|1];

encryptedText|Ien@h) '\0':

int main() (

char key[25];

char matrix[SIZE][SJE];

char plaintext[10D];

char encryptedText[100];

#### Scanned with OKEN Scanner

printf("Enter the key: “);

scanf(”Bs", key);

prepareKey(key, matrix);

printf("Enter the plaintext: ");

scanf(“Bs", plaintext);

encryptPIayfair(mstrix, plaintext, encryptedText);

prinh("Encrypted text: 96s\n", encryptedText);

return 0;

OUTPUT:

Enter the key: MONARCHY Enter the plaintext: ATTACK Encrypaed text: NUUNYJ

#### Write a C program for palyalphabeñc subst1tuñon cipher uses a separate monoalphabetic substituñon clpher for each successive le¢ter of plaintext, depending on a key.

PROGRAM:

t/includR <5tdio.h> #include <string.h> #incIude cctype.h»

void vigenereEncrypt(char \*plaintext, const char ’key) ( int keyLength = strlen(key);

int i,textLength = strlen(plaintext);

for (i  0: i < textLengh: i++) (

if (isaÏpha(plaintext[i])) (

#### Scanned with OKEN Scanner

char keyChar= key[i 46 keyLength];

int keyshih = isupper(keyChar) ? keyChar - 'A' : keyChar - 'a':

if (isupper(pIainte€[i])) (

plaintext[i] = (plaintext[i] - 'A' + keyshih) 9t 26 + 'A“,

plaintext[i] = (pIaintext[i] - 'a' + keyshik) 'łó 26 + 'a“,

void vigenereDecrypt(char ’ciphertext, const char ’key) (

int keyLength = str!en(key);

int i,textLength = strlen(ciphertext);

for ( i  0; i < textLength; i++) ( if (isalpha(ciphertext[i])) (

char keyChar = key[i \b keyLength];

int keyshift = isupper(keyChar) ? keyChar - 'A' : keyChar - 'a';

if (isupper(ciphertextji])) (

ciphertext|i] = (ciphertext[i]- 'A' - keyshih + 26) & 26 + 'A',

) elR (

ciphertext[i] = (ciphertext[i] - 'a' - keyshih • 26)B 26 + 'a’;

int main() (

char pIaintext[100];

char hey[100];

#### Scanned with OKEN Scanner

printf("Enter plaintext:”);

$ets(pIainteû, si2eof(plaintext), stdln);

plaintext[strcspn(plaintext, "\n“)] = '\O’; // Remove newline chaæcter

printf("Enter key: “);

fgets(key, sizeof(key), stdin);

key|strcspn(key. “\n“)] = '\0’, // Remove newliæ character

vigenereEncrypt{pIaintext, key); printf("Encrypted text: äs\n”, plaintext);

vigenereDecrypt(pIaintext, key);

printf("Decrypted text: 9às\n", plaintext);

return 0;

OUTPUT:

Enter plaintext: hello Enter key: apple Encrypted text: htaws Decrypted ten: hello

1. Writ+ï a C program for generalizaùon of the Caesar cipher, known as thR affine Caesar cipher, has the following form: For each plaintext IeRer p, substitute the ciphertext le¥er C: C = E([a, b], p) = (ap + b) mod 26 A basic requirement of any encryption algorithm is that it be one-to-one. That is, if p q, then E(k, p) E(k, q). Otherwise, decrypflon Is impossible, because more than one plaintext character maps into the same ciphertext character. The affine Caesar cipher is not only-to-one for all values of a. For example, for a  2 and b = 3, then E([a, b], 0) = E([», b], 13)  3.
   1. Are there any limitations on the value of b?
   2. Determine which values Df a are nDt allowed?

PROGRAM:

#include ctdio.k>

#### Scanned with OKEN Scanner

int gcd(int a, int b)

if(b  0)

eturn a;

return gcd(b, a 96 b);

int main()

prinh(°VaIues of 'a' not allowed {because they are not relañvely prime with 2b):\n")i for (int a = O; a < 26; a++)

if(gcd(a, 26) I= 1)

printf("%d ”, a);

printf("\n“);

return 0;

OUTPUT:

Values of'a' not allowed (because they are not relatively prime with 26):

0 2 4 6 8 10 12 13 14 16 18 20 22 24

1. Write a C program for ciphertext has been generated with an alTine cipher. The most frequent lerter of the cipheNext is ”B,” and the second most frequent leRer of the ciphertext is “U/’ Break this code.

##### PROGRAM:

#incIude <stdio.h> #include <string.h»

## Scanned with OKEN Scanner

char decryptChar(int c, int a, int b) { return ((a ” (c - b)) 4ó 26 + 26) 9ó 26 + 'A’;

int main() {

char cïphertex[1DØ]:

printf("Enter the ciphertext: ”); scanf(“9f›s", ciphertext);

int mostFrequent = ciphertew[0]:

int secondMostFrequent = ciphertext[1];

prìntf("Finding possible keys.. \n”);

for (int a 1; a < 26; a++) {

for {înt b = 0: b c 26; b++) {

if (decryptChar(mostFrequent, a, b) == mostFrequent && decryptChar(secondMostFrequent, a, b) == secondMostFrequent) ( prìntf(”Possible key found: a 4sd, b 96d\n", a, b);

return 0;

OUTPUT:

F.iiter the ciş)lim1ext SI/AMARl IDA

Finding possible keys...

Pmsiblc key found. a 1. b 13 Pnssit›le key foitn‹t: a = I ü. I› = 0 Possible źeş’ fouud: a )1. h 11

1. Write a C program for the fDllowing ciphertext was generated using a simple subsütuñon algorithm. 53t†t3D5))6\*;4B26)4T.)4†):BO6\*;48t8Ş6O))B5::]B\*;:†\*8tB3 (8R)ź\*t:ú5(:8R\*9ć›\*’f:R)\*,(,18fij;fi\*|2:\*,:(:19ź5\*ž(È —=ł)8Ț8\*

.JU09û8ž):)ct8Hțț.lÏ;'9:46Ud l:8:6] 1;48l 6ź,1g8ž I žû8â05°8I

(ț9.48.(88.Ț ț?14.48g,.ISI,.188,ț",

1. As you know, the most frequently occurring leher in English is e. Therefore, the first or second (or perhaps third?) mast common character in the message is likely to stand for e. Also, e is o8en seen in pairs (e.g., meet, fleet, speed, seen. been,

agicc. ztt:.). Ti y to Fiuá A cliAiactca iii the ï:iylica'tcxt llal JccuJcx tu c.

1. The moØ commDn word in English is “the.” Use this fact to guess the characters that stand

før t and h.

1. Decipher the rest of the message by deducing addİùonal WDrds.

PROGfL-I.\t

def decrypt simple substitutiDn(ciphertext, key):

decryption ""

Scanned with OKEN Scanner

fDF char in ciphertext:

if char.isaIpha():

decrypted char = key[char]

decryption += decrypted char

else:

decrypt›on + - char

return decryphon

ciphertext - "53tt 1305))6•,t8Z6)4t.)4t);806“,48\*8160))85;:)8’,:T”8t83 " \

"(88)5\*t;46(;88’96”?';8)I(;4B5):5\*2:T(:4956 •2(5—4)B18\* " \ ";d069285);)6\*8)4tt;1(t9;48081;8:811;48\*85;4)485\*528806“81 " \ "(19,48;(88,4(1734:48)4†;161;:188;T7;"

# Hii›ts

hints (

'\*’: 'E', # E is the most frequent letter

’4': T', /f T is one of the most common letters

'8': 'H', # H oken follows T

'I’: 'E', # E is ohen seen in pairs

'3’: 'R', # R is common and could fellow H 'I': 'A', # A is common and could follow T N', # N is common and could follow A

'6': 'I', t/ I is common and could follow A '5’: ’S', # S is common and could follow H '0': 'O', # O is common and could follow T

— 'F', If F is common and could follow 0

'U', # U is commo n and co uld follow Q ']': 'I', # L coul d follow U

'(’: 'W'. # W is a possibility far second most colt man letter

#### Scanned with OKEN Scanner

' ': 'W', t/ W is a possibility for second most common letter

')': ‘Y', # Y could follow W

?’: 'G', # G ie a possibility for third molt common letter

# Decrypt the message using the provided hints decryption key = (k: v for k, v in hints.items() if k,isaIpha())

decrypted message - decrypt simple subsfitufion(ciphertext, decryption key I

print(’Decrypted Message:") print(decrypted message) OUTPUT:

Decrypted Messsge:

5.3t4\*305))6 •;4826)4†.}4†):806”;48\*8\6O))B5;;]B”;:?”8t83

(88)S”\*:46(,88’96” 7;8)t(,485);5\*2:4(;4956”2(5—4)8$8”

:4069285):}6\*8}4tt;1(T9;48081;8:811;48185;4)48S\*S28806•81

(t9.48:(88,4(t?34:4B)4t,161;:188;J?;

1. Write a C program for monoalphabet1c cipher is that both sender and receiver must commit the permuted cipher sequence to memory. A CDmmon technique fDr avoiding this is to use a keyword from which the cipher sequence can be generated.

l't›1 ‹i.\‹t1lI|t1‹i. tlsI1tR I1i‹i 1‹ii11't›I ‹I *b'/t'I#L'£* \1’I tl‹i tt\II the Lñi’t1 tl1tl It›(l‹›tlñt) it)’ Itllllsfitl

l•itt«i”x in n‹›ixiiaI ‹›i”1cl” ailJ a»ati:If t1›ix akaii›•t tl\« j›laia›tcxt Irttcl•

{›litii1 :t It c• tl c t g 111 ] k 1 ill 11 t1 ]1 th 1 x 1.11 “ 11“ X ›“ r

cij›Iici I"I F II I: It .1 It IJ I'“ I i .I I^t 1. .II h’ (J tg ii "I" I" 1" It ?•? 'i” ?'

PROGRAM.

def generate cipher sequence(keyword):

keyword keyword.upper()

alphabet "ABCDE FGHIJKLMNOPQRSTUVWXYZ"

cipher sequence = ’"

for cl ar in keyword:

if char not in cipher sequence: cipher sequence •= char

for char in alphabet:

if char not in cipher sequence:

## Scanned with OKEN Scanner

cipher sequence + - char

ret urn cipher sequence

def encrypt(plain text, cipher sequence): plain text plain text.upper() encrypted text ""

for char in plam text:

if char.is alpha():

index = ord(char) - ord(’A’)

encrypted text +• cipher sequen ce[index

else:

encrypted text • char rnt urn encrypted text

def decrypt(encry pted text, cipher seque nce): encrypted text encrypted text.upper() decrypted text ""

for char in encrypted text:

if char.is alpha():

index cipher sequence.index{char)

decrypted text ‹= chr(index + ord{'A’)) else:

decrypted text • char

return decrypted text keyword "CIPHER"

cipher sequence generate c ul er seque nce(keyword)

plain text "hells world"

encrypted text = encrypt(plain text, cipher sequence) decrypted text = decrypt(encrypted text, cipher sequence) print("Plain Text:", plain text)

print("Cipher:", encrypted text)

print("Decrypted Text:", decrypted text)

OUTPUT:

#### Scanned with OKEN Scanner

Plain Text: hello world

Cipher: BEJJM WMOH

Decrypted Text: HELLO WORLD

1. Write a C program for PT-109 American patFDl boat, under the command of Lieutenant John F. Kennedy, was sunk by a Japanese destroyer, a mexage was received at an Australian wireless station in Playfair code:

UREBE ZWEHE WRYTU HEh\*FS KkliHE GOYF1 WTTTU OLKSh" CAJPO BOTEl ZONTX BYBNT GÜ›NEY TUZ\YM GDSi iN SXBOU

\'4$'RHE BAAHY USEDQ

###### PROGRAhI:

tiincliide <stdio.h>

äinclude <strinp.h>

void decryptPlayfair(char

messape[]. char key[]) ]



int keylndex 0:

for (i 0, i < 5. i++) {

for (j 0: j < 1. ja) (

u›aßLx(iJ[j] keg’[keyüi6x++j:



for (i  0. i <

stJeu(n essape): i +' ?) ( cüer c l = iiiessege} i]: chaz c2 iuessape(i +

ist r1. c l index. r2.

c2 index.

for (j = 0. j < S. j+>) { for (int k = 0: k < 5:

if (nmirix[j]|k]

c l ) {

rl = j.

cl iiider k.



.

c2 index k

### Scanned with OKEN Scanner

>«nyrq;«i »v>+ sl%

5], iilatrix[r?][(c2 index + 4)



) else ’if(cl index ==

c? iixlex) J

###### printf(‘%«c%a”.

aiaeix](rl + 4) %

5)[cl index]. matrix[(r2 + 4)

% iS][c? index]).

} else {

inseix[rl)[c? index].







char ninsageQ = "KXJEYUREBEZWEHEWR YTUHEYFSKREHEGOYFI WTTTUOLKSYCAJPOBOT EIZONTXBYBNTGONEYC UZWRGDSONSXBOUYWR HEBAAHYUSEDQ".

char key]j

“PLAYFIREXMBCDGHKN

OQSTtnWZ•.

decrypiP&yfair(message. key).



OUTPUT:

qioawieoien+wxrOzxoitn

YQSIRDMDQAYXTZZZTNASQLG

LDAKDK1'IMTNKWIGPCKWBNK WCHONKQMDKWLUX

DMDPFDLWOMGO

# Write a C program for Playfair matrix:

M F H 1/1 K

U N O P

## Z V W X Y

E L A R G

### Scanned with OKEN Scanner

I-.i»rzyt thix ilacxxagc: .1\ttxt x» v‹»tt ‹»”cr £”aJo¡jait \1 cx\. f'‹»it titg at ‹›nL-c.

PROGRAM:

def create playfa r matrix(key):

key •= ".join(chr(65 • i) for i in range(25) if cf r(65 • i) not in key and cf r(Gs • i j != ' 'j

matrix [list(key[i: i+5]) for i in range(0, 25, 5)]

ret urn mat rix

def encrypt message(matrix, message):

message = |message|i:i•2| if i • 1 < Ien(message) else message|i:i• 1] for i in range(0, len(message), 2ll

def *h*nd coordinates(r):

for row, row vals in enumerate(matrix):

if c in row vals:

ret urn row, row vals.index(c)

return None, None

encrypted

for paif in message:

rJ, c) = hnd coordinates(pair|0])

r2, c 2 = hnd too rdinates(pair|1]) if len(pai r) -- 2 else tr1, c1)

if r1 is not None and r2 is not None and c1 is not None and c2 s not None:

if r1 == r2:

encrypted •• matr x|r1]|(c1 • 1) 4 5| • matrix|r2][(c2 • 1) % 5

elif c1 •• c2:

encrypted = matrix[(r1 1) % 5]|c1| matrix[(r2 I) P• 5]|c2| else:

encrypted •= matrix[rt][c2] • matrix[r2][c1]

else:

enc rypted = pair

## Scanned with OKEN Scanner

return encrypted

def main():

key "MFHIJKUNOPQZVWXYF LARGDSTBC"

matrix create playfa ir matrix(key)

message "MUSTSEEYOUOVE RCADOGANWESTCOMiNGATONCE"

encrypted = encrypt mesSage(matrix, message)

print(' Original Message:", message)

print(' £ncrypted Message:", encrypted)

if name == " main "’

OUTPUT:

Original Message: MUSTSEEYOUOVE RCADOGANWfiSTCOMINGATONCE

Enc rypted Message: FKT BDLLL PNNWLYCATUTYOVLDTCKIHOTYIWNCLL

1. Write a C program for possible keys does the Playfai r cipher have? Ignore the fact that some keys might produce idenñcal encrypñon results. EX fRSS your answer as an approximate power of 2.
   1. Now take into account the fact that some Playfair keys produce the same encryption results. How many effect1vely unique keys does the Playfair cipher have?

PROGRAM:

#incIude <stdio.h> #incIude <string.h>

unsigned long long factoriaI(int n) (

if (n <= I)

return 1:

return n ” factoriaI(n - 1);

int main() {

int keyLength 25;

## Scanned with OKEN Scanner

unsigned long long totalPossibleKeys — factoriaI(keyLength);

printf("Total possible keys (without considering idenfical enc rypñon results): %hu\n”, totalPossibleKeys),

unsigned long long effecfivelyU niqueKeys = totalPossibleKeys / keyLengt h;

printf("E ffec tively unique Leys (co nsider ing idenhcal enc rypLion results): %hu\n”,

effechvelyU nique Keys):

return 0:

OUTPUT:

Total possi ble keys (without consiclering identical encrypño n results): 70345 352 77573563776

EWecnvely unique keys (co nsiclering idenhcal encrypden results): 281 3814 1 1 10295855 I

#### a. Write a C program to Encrypt the message ”meet me at the usual place at ten rather than eight oclock” using the Hill cipher with the key.



##### » Show your calculations and the result.

I› Show the calcularions for the corresponding decryprion of the

ciphertext to recover the original plaintext.

##### PROGRAM:

#incIude <stdio. h› #include stdIib.h> #incIude <string,h> #incIude xc type.h> #ir elude <n atf ,h>

//dehne MAX LEN 100

int charTONurn(Char c) (

i( (isupper(c))

return c - ’A‘;

) else if (islower(c)) (

return c - 'a':

return 1:

char numToChar(int num) ( return num + 'A':

void encryptHill(char • text, int ”keyMatr ix, int key size) (

int i,j,k,textLen = strIen(text);

int encrypted[MAX LEN| - (0}:

## Scanned with OKEN Scanner

for ( i 0, i < text Len, i •— keysize) {

for ( j 0, j < keysize; j••)

int sun = 0:

for (k 0: k < keysize, k++) {

sum + keyMatrix§ keysize • k] \* charToNum(text|i + k]);

encrypted|i + j] sum % 26;

for {i 0; i < textLen; i+ +) (

text[i] numToChar(encrypted[i]);

int main() {

char plaintext|MAX LENj;

int keysize:

printf("Enter the plaintext: ");

gets{pIaintext);

printf("E nter the size of the Ley matrix: "):

scanf("46d", &keysize);

int i,j,keyMatrix|MAX LEN \* MAX LEN];

printf( 'Enter the key matrix (row by row):\n"j: far ( i 0; i < LeySize; i•+) {

for ( j 0; j keysize: j+•) (

scanf("had", &keyMatr ix|i \* keysize j]);

int text Len - strlen(pIaintext);

int padding - keysize - (text Len ’1 keysize); if (padding < keysize) (

for ( 0: i padding: i ) {

pIaintext|textLen + i] - 'X';

pIaintext[text Len • parlcling] = 'TO':

encryptHill(pIaintext, keyMatrix, keysize); printf(“Encrypted text: %s\n", plaintext); return 0;

##### OUTPUT:

Enter the plaintext: meet me at the usual place at ten rather then eight oclock

## Scanned with OKEN Scanner

Enter the size of the key matrix: 2

Enter t he key matrix (row by row):

9 4

5 7

Encrypted text: UKIXNBGNYDPYBLTFIWSZZWVDIM8«LKFTJGXHROAJPY8UGQYEBLKEGXGC

#### Write a C program for Hill cipher succumbs to a known plaintext attack if sufficient plaintext— ciphertext pairs are provided. It is even easier to solve the Hill cipher if a chosen plaintext attack can be mounted.

##### PROGRAM:

#incIude <stdio.h>

#inc lude <stdIib. h>

//inc lude <string. h> #incIude <ctype. h› Itinclude <math. h>

#dehne MAX LEN 100

int charToNum(char c) { if (isupper(c)) {

feturn c 'A',

} else if (islower(c)) { return c - 'a',

feturn 1:

char numToChar(int num} (

fet urn num + 'A',

void encryptHiII(char ° text, int \* keyMatrix, int keysize) (

int i,j,k,text Len - strIen(text),

int ent rypted(MAX LEN| = {0};

for ( i 0. i < text Len; i •= keysize) ( for ( j 0, j < keysize, j••) (

int sum = 0:

for (k — 0, k < keysize; k••) {

sum •= keyMatrix|j " keysize • k| ” charToNum(text|i • k]): encrypted(i + j| sum P 26;

for (i - 0, i < textLen; i+ +) (

text[ij numToChar(encrypted[i]);

int main() (

#### Scanned with OKEN Scanner

char plaintext[MAX LL N|,

int keysize;

printf(”L nter the plaintext: "); gets(plaintext):

printY("E nter the site of the key matr x: ");

scanf("%d", &keysize):

int i,j,keyMatrix[MAX LEN ” MAX LEN):

printf(”Enter the key matrix (row by row):\n"); far ( i 0; i < keysize; i +) {

for ( j 0; j < keySi7e; j+ +) (

scanf(”and”, &keyMatrix[i \* keysize + j]);

int textLen = strIen(plaintext);

int padding = keysize - (text Len % key5ize);

if (padding < keysize) {

for ( i 0; i < padding: i++) (

plaintext|textLen + i| - 1X';

plaintext[textLen + padding] - ’\0’;

encryptHilI(plaintext, keyMatrix, keysize):

printf{”Encrypted text: %s\n", plaintext); ret urn 0,

## OUTPUT:

Entef the plaintext: hello

Enter the size of the key matrix: 2

L nter t lie key mat rix {row by row):



2 3

Encrypted text: PAHDIT

1. Write a C program for one-time pad version of the Vigenére cipher. In this scheme,

the key is a stream of random numbers between D and 26. For example, if the key is 3 19

## Scanned with OKEN Scanner

5 ... , then the first leher of plalntext is encrypted with a shih of 3 lettex, the second

with a sh•iftof 19 lehers, the third with a sh1h of S leRers, and so on.

1. Encrypt the plaintext send more

money with the key stream 9 0 1 7 23

###### 15 Z1 14 11 1i 2 a g

h. Using the ciphertext produced In part (a), find a key so that the cipher text decrypts to the plsintext cash not

needed.

#### PROGRAM:

#incIude ctdio.h» #incIude <string.h>

void encrypt(const char •pIaintext, const

int \*ley, chsr “cipherte€) (

int plaintextLen = strIen(plaintext);

int i,

for (i  0; i < plaintextLen;i+\*) (

ciphertext[i] = (plaintext[i] - 'A' +

key|i]) 4t 26 'A';

ciphertext[pIaintextLenj '\0':

void decrypt(const char “ciphertext, const Int \*key, char ’plaintext) (

int ciphertextLen = strlen(ciphertext),i;

for (i  0; i < ciphertextLen; i++) (

pIaintext|i) = (ciphertext[i] - 'A' - key|ij

###### + 26) 4t 26 • 'A':

pIaintext|ciphertext£en)  ’\0';

int main() (

const char \*plaintext =

"SENDMOREMONEY";

int key|]  (9, 0, 1, 7, 23, 13, 21, 14, 11,

char ciphertext[strlen(plaintext) + 1]; encrypt(plaintext, ley, ciphertext);

printf("Ciphertext: %s\n", ciphertext),

char decryQedText[strlen(plaintext) +

1]:

## Scanned with OKEN Scanner

decrypt(ciphertext, key, decryptedText); printf(“Decrypted Text: Ks§n",

decryptedText):

return 0;

OUTPUT:

Ciphertext: B£OKJDMSRPMH

Decrypted Text: SENDMOREMOXEY

1. Write a C progmm that can perform a letter frequency attack on an additive cipher wiihout human intervention. Your software should produce possible plaintexls in rough order of likelihond. It «'ould bR good if your user interface allowed lhe user lo specify “give me the top 10 possible plaintexts.”

##### PROGRAM:

#include <stdio.h> #include <stdIib.h> #incIude <string,h> #incIude <ctype.h»

#define ALPHABET\_SIZE 26

// Funcdon to decrypt the ciphertext ustng the specihed shiR value

void decrypt(char "ciphertext, int shik) (

int lengh = strlen(éphertext);

int i;

for ( i  0; i < length; i++) ( if(isalpha(ciphertext|i))) (

if(isupper(ciphertext|i])) (

cipherte€|i] = 'A' + (ciphertext[i] - 'A' - shih + ALPHABET SIZE)B ALPHABET SIZE;

## Scanned with OKEN Scanner

) else (

ciphertext|i] = ’a' • (ciphertext[i] - ’a’ - shift + ALPHABET SIZE) B ALPHABET\_SIZE;

// Function to count the frequency of each lever in the plaintext

void countLeRerFequency(char ’text, int ’frequency) (

int lengh = strIen(text);

int i;

for (i = 0; i < length; i++) (

if(isalpha(text|ij)) {

if(isupper(text|i))) (

frequency|text[i] - 'A’]++;

) else (

frequency[text[i] - 'a']++;

// Funcñon to find the shih value with the maximum frequency match

Int findShihVaIue(int "frequency) (

int maxFrequency = 0;

int shik = 0;

for ( i  0; i «ALPHABET SIZE; i++) {

if(frequency[i] » maxFequency) (

maxFrequency  frequency|i);

## Scanned with OKEN Scanner

shift = (ALPHABET SIZE - i) 8 ALPHABET SIZE;



return shirt;

int main() {

char ciphe<ext[1000]:

printf("Enter the ciphertext: "); fgets(ciphertext, sizeof(ciphertext), stdin); int i;

int letterFrequency[ALPHABET SIZE] = (0);

countLeRerFrequency(ciphertext, IetterFrequency);

int shih = findShihValue(leRerFequency):

printf("Possible plaintext In order of likelihood:\n");

for (i 0; i < 10; i++) { decrypt(ciphertext, shift); printf("96d. %s\n", i + 1, ciphertext);

return 0; )

OUTPUT:

Entlzr the Ciphe.FteXt: lipps

Possible plaintexts in order of likelihood:

1. axeeh
2. pmttw
3. ebiil

## Scanned with OKEN Scanner

1. tqxxa
2. ifmmp
3. xubbe
4. mjqqt

###### byffi

1. qnuux
2. fc§m
3. Wrlte a C program that can perform a letter frequency attack on any monoalphabetic substituñon cipher without human intewenhon. Your soRware should produce possible plaintext in rough order of likelihood. It would be. good if your user interface allowed the user to specify “give me the top 10 possible plaintexts."

# PROGRAM:

#inclu#incIude <stdio.h> #incIude <string.h> #incIude <ctype.h>

#define ALPHABET SIE 26 #define NUM\_TOP\_PLAINTEXT5 10

const double engIishLexerFreq[ALPHABET SIZEj = (

0,0817, 0.0149, 0.0278, 0.042S, 0.1270, 0.0223, 0.O2O2,



0.0193,

0.0697, 0.0015, 0,0077, 0.603, D,0241, O.0675, D,O751,

###### 0.0010, 0.0599, 0.0633, 0.606, 0.0276, 0.0098,0.0236, 0.0015, o.oD7, a.0007

void calculateLetterFrequency(const char ”text, double "freq) { int totalLRtters = D,i;

!or li 0: te^tli]: i\*+I I

if(isaipha(text|i])) (

freq[toIower(text|i]) - ’a’j++;

totalLecers++;

for (i = 0; i < ALPHABET SEE; i++) (

feq[i] /= totalLetters;

double caIculateScore(const double ’freq) (

## Scanned with OKEN Scanner

double sco re 0.0,

int i;

for ( i 0; i c ALPHABET \iIZfi; i•+) ( score •= freq[i| engIishLetterFreq[i];

ret u n score,

void decryptSubs0tu0on(const char •ciphertext, char \*plaintext, int sh1ù) (

int i.

far (i 0, ciphertextt i); i++) {

if (isalpha(ciphertext ti)j) (

char base = isupper(ciphertext|i|) 2 'A' : 'a';

pIaintext|i) = (c›phertext[i) base shift + ALPHABET SIZE)8 ALPHABET SIZE + base;

} else {

pIaintext|i] = ciphertext|i|;

plaintext[strIen(ciphertext)] '\0';

int main() {

const char • ciphertext = "FALSXY XS LSX!'1: // Replace with your ciphertext

double ciphertext Freq|ALPHABET SIZE] = {0.0};

int shift;

calcuIateLetterFrequency(ciphertext, ciphertextFreq):

printf("Ciphertext: %s\n\n' , ciphertext);

printf("Top had possible plaintexts:\n1', NUM TOP PLAINTEXTS):

for (shift 0: shift < ALPHAB fiT SIZE; shift+•) {

char possible Plaintext|strlen(c iphertext) • 1];

dec rypt Substlt utio n(cip hertext, possibleP1aintext, Shih); double possiblePlaintext Freq|AL PHABET SIZE) = {0.D};

caIcuIateLetterFrepuency(possibIePlaintext, possiblePlaintextFrep):

double score - calculate5core{possible Plaintext Freq):

printf("Shift %d: %s (Sco re: 'a.4f)\n', shaft, possible Plaintext, score),

retu«› 0.

### OUTPUT:

Ciphertext: FALSXY XS LSX!

Top 10 possible plaintexts:

Shift 0: FALSXY XS LSX! (Score: 0.D362)

Shih I: EZKRWX WR KRW! (Score: D.0359)

## Scanned with OKEN Scanner

Shift 2: DYJQVW VQ JQV!(Score. 0.0 110) Shih 3: CXIPUV UP IPU! (Score: 0.0290) Shih 4: BWHOTU TO HOT!(Score 0.0623) Shih 5: AVGNST SN GNS!(Score: 0.0559) St›ih 6: ZUFMRS RM FMR!(Score: 0.0353) Shift 7: YTELQR QL ELQ! (Score: D.0498) Sh‹h g: xsDKPQ PK DKP! (Score: 0.0211) Shik 9: WRCJOF OJ CJOI (Seare: 0.0353) Shift 1 0: VQBINO NI BIN! Score: 0.04 79)

Shift 1: UPAHMN MH AHM! (Score: 0.684)

Shift 12: TOZGLM LG ZGL! (Score: 0.0339) Shih 13: SNYFKL KF YFK I [Score: 0.0273) Shik 14: RMXEJK JE XEJ I (Score. 0.0437) Shift 15: QLWDIJ ID WDI!(Score: 0.0388) Shift 16: PKVCHI HC VCH! (Score: 0.0348) Shit j 7: OJUBGH GB UgG! {SCOfR: 0.027t)

Shift 18: NITAFG FA TAF! (Sco re: 0.0591)

Shift 19: MHSZEF EZ SZE! (Score: O,0561) Shih 20: LGRYDE DY RYD! (Score: 0.0449) Shit 21: KFQXCD CX QXCI [Sco re: 0.0148) Shift 22: JEPWBC BW PWB! |Score: 0.0282) Shift 23: IDOVAB AV OVAI (Score: 0.0502) Shih 24: HCNUZA ZU NUZ! (Sco re: 0.0353) Shih 25: GBMTYZ YT MTY! (Sco re: 0.0377)

1. Write a C program for DES algDrithm for decrypt1on, the 16 keys (K1, K2, c, K16) are used in reverse order. De‘sign a key-generat1on scheme with the appropriate shiR schedu ie for the decrypñon process.

##### PROGRAM:

#incIude <stdio.h>

#### Scanned with OKEN Scanner

#incIude <stdint.h>

iatcconstint(P(| ( 2, 6,3, 1, 4,8, 5,7);

staLc const int IP\_INV(] - ( 4, 1, 3, 5, 7, z, g, 6 ),

stahc const r int64 t KFY = 0x 133457799BBCDFF 1.

sta tic const uin(64 t CIPHERTE XT = 0x0L 23456789ABCDE F; uint64 t permute(uint64 t input, const int •tabIe, int size) (

uint64 t result - 0:

int i;

for (i ß; i < size; i••) {

result | ((input » (64 - tabIe[i|)) & 1) ‹< (size - ) - i);

ret um result;

uint64 t des decrypt(uint64 pt ciphertext, uint64 t key) { uint64 t permuted ciphertext = permute(ci pherte xt, IP, 64); uint64 t decrypted - pe‹muted ciphertext ^ key.

decrypted pern1ute(decrypted, IP INV, 6):

return decrypted,

int main() (

uint64 t decrypted • des decrypt(CIPHERTEXT. KEY):

printf("Ciphertext: 0x%016IIX\n“, CIPHERTEXT);

printf("Decrypted: 0x’t‘‹d16IIX\n", decrypted):

return 0;

OUTPUT:

Ciphertext: 0x0 t 23456789ABCOEF

#### Scanned with OKEN Scanner

Decrypted: 0x8B t02312f531 B66A

1. Write a C program for DES the first 24 bits of each subkey come from the same subset of 28 bits of the initial key and that the second 24 bits of each subkey come from a disjoint subset of 28 bits of the inifial key.

##### PROCRAM:

#incIude <stdio.h>

# inc lude <stdint. h>

stañc const int IP() ( 2, 6, 3, 1, 4, 8, 5, 7 );

static const int PCJ[] { Z, 4, 1, 6, 3, 9, 0, 8, 5, 7 );

stafic const int PC2[] - { 5, 2, 6, 3, 7, 4, 9, 8 );

state const uint64 t KEY - 0xLO00F FFFFFFFFFFF;

uint64 t permute(uint64 t input, const int • table, int size)

uint64 t result 0:

int i:

fo r ( i 0; i < site; i +) (

result I = Il'nput >> |64 - table[i])) & 1) << (size - 1 - i);

return result:

vo id generate subkeys(uint64 t key, uint64 t \*subkeys) (

key = permute(Ley, PC1, 56):

int i;

fo r ( i 0; i « IG. i+ › ) (

uint64 t shifted key (key « i) | (Ley » {Z8 - i));

subkeys(i] - pcrmute(shifted key, PC2, 48);

int main() (

uint64 t subkeys(T 6j;

generate subLeys(KEY, subkeys):

printf{"Generated Subkeys:\n"); for (i 0, i < 16, i+ ‹) (

printf("K%d: 0x‘)6012lIX\n", i 1, subkeys|i]);

return 0,

OUTPUT:

Generated Subkeys:

KI: 0x00FF 5EF5D92A K2: 0x00FF7DFB 7C7F K3: Ox00FF7CED6C7 F K4: 0x00FF 7CFD3CFF KS: 0xfX\FF768DFDF7

## Scanned with OKEN Scanner

K6:0xDOC07B7EF5FF K7: DxO23 F7FDF FDF7 K8. 0x01 FF 7B7FF SFF K9: Ox08FF 6FF F f OFC KI0: 0x20FF SFFFD9GE KI J: 0x82 FF 3DF F7E 7r K12: 0x07C07CFC7C SF K13: OxlB003CFC7C7F KI4: 0x6B3F7CFD7C5F KIS: 0xAgF FFC F57E 3B K16: DAFFF7CF97E7F

1. Write a C program for encrypfion in the cipher block chaining (CBC) mode using an algorithm stronger than DES. 3DES is a good candidste. Both of which follow from the definition of CBC.

###### \\’hlsll C\*l“ tllC I ’€\* \S''utild ¥'utl cllo€JSe:

* 1. For security?

I› For performance?

##### PROGRAM:

from Crypto.Cipher impon DES3

from Crypto.RandOI+› import get fandom bytes

def pall(text, block size):

padding size - block size Ien(text) % block size pa‹1ding - bytes(|pa‹JcIing size| ' padrllng sire) return text • padding

def encrypt 3des cbc(plaintext, key):

iv get random hytes(8) # Init1alizañon vector cipher - DES3.new(key, DES3.MODE CBC, iv) ciphertext = cipher.encrypt(pad(plaintext, 8)) ret urn iv • ciphertext

def decrypt 3des cbc(ciphertext, key):

iv c iphertext(:8|

ciphertext ciphertext|8:]

cipher — DES3.new(key, DES3.MODE CBC, iv) decrypted cipher.decrypt(ciphertext) padding size decrypted[-1]

return decrypted[: padding size]

def main():

key - get randDm bytes(24) # 3DES requires a 24-byte key

plaintext = "Hello, this is a test message."

## Scanned with OKEN Scanner

plaintext — plaintext.encode('utf-8'

enc rypted = encrypt 3des cbc(plaintext. key)

decrypted = decrypt 3des cbc(encrypted, key).decode('utf 8’)

print("Plaintext:", plaintext)

print("E ncrypted:", encrypted. hex())

print("Decrypte‹J‘.", ‹Jec rypted)



main()

OUTPUT:

plaintext: ’hello this is a text message1

Encrypted:

cd10cc23488298D42dd971953c 54f2e5c2of4cd34d4zigzds72348 16244 1 fdd718745aI4d4a25397

Decrypted:'hello this iS a text message

1. Write a C program for ECB mode, if there is an error in a block of the transmihed ciphertext, only the corresponding plaintext block is affected. However, in the CBC mode, this error propagates. For example, an error in the transmitted C1 obviously corrupts P1 and P2.

.‹ Are any blocks beyond P2 affected?

I› Suppose that there is a bit error in the source version of P1. Through how many ciphertext blocks is this error propagated? What is the eflect at the receiver 7

### PROGRAM:

#inc lude <stdio.h> Pinc lude <string.h>

void eneryp\BIocL{char plaintext, char "ciphertext) strcpy(ciphertext, plaintext),

void decryptBIock(char \*ciphertext, char \*plaintext)

strcpy(plaintext, ciphertext);

void simuIateTransmittedCiphertextError(char ’ciphertext, int blocklndex)

ciphertext[blockIndex) ^= 0xOt;

int main()

 Scanned with OKEN Scanner