CRYPTOGRAPHY AND NETWORK SECURITY (CSA5160)

1.Caesar cipher

#include <stdio.h>

#include <ctype.h>

void encrypt(char \*text, int k) {

for (int i = 0; text[i] != '\0'; ++i) {

if (isalpha(text[i])) {

char offset = isupper(text[i]) ? 'A' : 'a';

text[i] = (text[i] - offset + k) % 26 + offset;

}

}

}

void decrypt(char \*text, int k) {

for (int i = 0; text[i] != '\0'; ++i) {

if (isalpha(text[i])) {

char offset = isupper(text[i]) ? 'A' : 'a';

text[i] = (text[i] - offset - k + 26) % 26 + offset;

}

}

}

int main() {

char text[100];

int k;

int choice;

printf("Enter a string: ");

fgets(text, sizeof(text), stdin);

printf("Enter the shift value (1-25): ");

scanf("%d", &k);

if (k < 1 || k > 25) {

printf("Invalid shift value.\n");

return 1;

}

printf("Enter 1 to Encrypt or 2 to Decrypt: ");

scanf("%d", &choice);

if (choice == 1) {

encrypt(text, k);

printf("Encrypted text: %s\n", text);

} else if (choice == 2) {

decrypt(text, k);

printf("Decrypted text: %s\n", text);

} else {

printf("Invalid choice.\n");

return 0;

}

4.Vigenere cipher

#include <stdio.h>

#include <string.h>

void vigenereCipher(char\* plaintext, char\* key) {

char ciphertext[100];

int i, j, len = strlen(plaintext);

int keyLen = strlen(key);

for (i = 0, j = 0; i < len; i++) {

char c = plaintext[i];

if (c >= 'a' && c <= 'z') {

ciphertext[i] = (c - 'a' + (key[j] - 'a')) % 26 + 'a';

j = (j + 1) % keyLen;

} else if (c >= 'A' && c <= 'Z') {

ciphertext[i] = (c - 'A' + (key[j] - 'A')) % 26 + 'A';

j = (j + 1) % keyLen;

} else {

ciphertext[i] = c;

}

}

ciphertext[len] = '\0';

printf("Encrypted text: %s\n", ciphertext);

}

int main() {

char plaintext[100];

char key[100];

printf("Enter a string to encrypt: ");

gets(plaintext);

printf("Enter the key: ");

gets(key);

vigenereCipher(plaintext, key);

return 0;

}

11.Write a C program for possible keys does the Playfair cipher have? Ignore the fact that some keys might produce identical encryption results. Express your answer as an approximate power of 2.

#include <stdio.h>

#include <math.h>

// Function to calculate the log base 2 of factorial

double log2\_factorial(int n) {

double log2\_factorial = 0.0;

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

log2\_factorial += log2(i);

}

return log2\_factorial;

}

int main() {

int n = 25;

// Step 2: Calculate total possible keys

double total\_log2\_keys = log2\_factorial(n);

printf("Total possible keys: 2^%.2f\n", total\_log2\_keys);

// Step 4: Calculate symmetries

double symmetries\_log2 = 5 \* log2(2) + 2 \* log2\_factorial(5);

double unique\_log2\_keys = total\_log2\_keys - symmetries\_log2;

// Step 5 and 6: Calculate effectively unique keys and convert to power of 2

printf("Effectively unique keys: 2^%.2f\n", unique\_log2\_keys);

return 0;

}

12. 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. 9 4 5 7 a. Show your calculations and the result. b. Show the calculations for the corresponding decryption of the ciphertext to recover the original plaintext.

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int mod(int a, int m) {

int result = a % m;

if (result < 0) {

result += m;

}

return result;

}

void hillCipherEncrypt(int keyMatrix[2][2], char\* input, int len) {

int encrypted[len];

printf("Plaintext: %s\n", input); // Print the plaintext

for (int i = 0; i < len; i += 2) {

for (int j = 0; j < 2; j++) {

encrypted[i + j] = 0;

for (int k = 0; k < 2; k++) {

encrypted[i + j] += keyMatrix[j][k] \* (input[i + k] - 'A');

}

encrypted[i + j] = mod(encrypted[i + j], 26);

}

}

printf("Ciphertext: ");

for (int i = 0; i < len; i++) {

printf("%c", encrypted[i] + 'A');

}

printf("\n");

}

int main() {

int keyMatrix[2][2];

char input[100];

int len;

printf("Enter the 2x2 key matrix:\n");

for (int i = 0; i < 2; i++) {

for (int j = 0; j < 2; j++) {

scanf("%d", &keyMatrix[i][j]);

}

}

printf("Enter the plaintext (in uppercase): ");

scanf("%s", input);

len = strlen(input);

while (len % 2 != 0) {

input[len] = 'X';

len++;

}

hillCipherEncrypt(keyMatrix, input, len);

return 0;

}

15.Write a C program that can perform a letter frequency attack on an additive cipher without human intervention. Your software should produce possible plaintexts in rough order of likelihood. It would begood if your user interface allowed the user to specify “give me the top 10 possible plaintexts.”

#include <stdio.h>

#include <string.h>

#define ALPHABET\_SIZE 26

// Decrypts the ciphertext using Caesar cipher with a given shift

void decrypt(const char \*ciphertext, int shift, char \*plaintext) {

for (int i = 0; ciphertext[i]; i++) {

char c = ciphertext[i];

plaintext[i] = (c >= 'A' && c <= 'Z') ? 'A' + (c - 'A' - shift + ALPHABET\_SIZE) % ALPHABET\_SIZE : c;

}

plaintext[strlen(ciphertext)] = '\0';

}

int main() {

char ciphertext[] = "XLIWMWXEPMWLI"; // Example ciphertext

char plaintext[100];

// Decrypt and display each possible plaintext

for (int shift = 0; shift < ALPHABET\_SIZE; shift++) {

decrypt(ciphertext, shift, plaintext);

printf("Shift %2d: %s\n", shift, plaintext);

}

return 0;

}

24. Write a C program for RSA system, the public key of a given user is e = 31, n = 3599. What is the private key of this user? Hint: First use trial-and-error to determine p and q; then use the extended Euclidean algorithm to find the multiplicative inverse of 31 modulo f(n).

#include <stdio.h>

int gcd(int a, int b) {

if (b == 0)

return a;

return gcd(b, a % b);

}

int modInverse(int a, int m) {

int m0 = m, t, q;

int x0 = 0, x1 = 1;

if (m == 1)

return 0;

while (a > 1) {

q = a / m;

t = m;

m = a % m, a = t;

t = x0;

x0 = x1 - q \* x0;

x1 = t;

}

if (x1 < 0)

x1 += m0;

return x1;

}

int main() {

int e = 31;

int n = 3599;

int p, q;

for (p = 2; p < n; p++) {

if (n % p == 0) {

q = n / p;

break;

}

}

int phi\_n = (p - 1) \* (q - 1);

int d = modInverse(e, phi\_n);

printf("Private key (d) is: %d\n", d);

return 0;

}