**Lab Report: 1 Date:2024/06/02**

**Encryption:**

**Experiment 1:** Implementation of encryption process

**Objective:**

To understand and implement the process of encryption and decryption using various algorithms, analyze the security features of encryption techniques, and evaluate their effectiveness in safeguarding information in Information Systems (I.S.).

**Theory:**

Encryption is the process of converting plain, readable data into a coded format (ciphertext) using an algorithm and a key. This ensures that only authorized users with the correct key can decrypt and access the original information, protecting it from unauthorized access or theft.

**Code Implementation:**

*#include<stdio.h>*

*int main()*

*{*

*char message[100], ch;*

*int i, key;*

*printf("Enter a message to encrypt: ");*

*gets(message);*

*printf("Enter key: ");*

*scanf("%d", &key);*

*for(i = 0; message[i] != '\0'; ++i){*

*ch = message[i];*

*if(ch >= 'a' && ch <= 'z'){*

*printf("Numeric ascii Accept");*

*ch = ch + key;*

*printf("Character Value: %C", ch);*

*if(ch > 'z'){*

*ch = ch - 'z' + 'a' - 1;*

*}*

*message[i] = ch;*

*}*

*else if(ch >= 100 && ch <= 126){*

*printf("Numeric ascii Accept");*

*ch = ch + key;*

*printf("Character Value: %C", ch);*

*if(ch > 'Z'){*

*ch = ch - 'Z' + 'A' - 1;*

*}*

*message[i] = ch;*

*}*

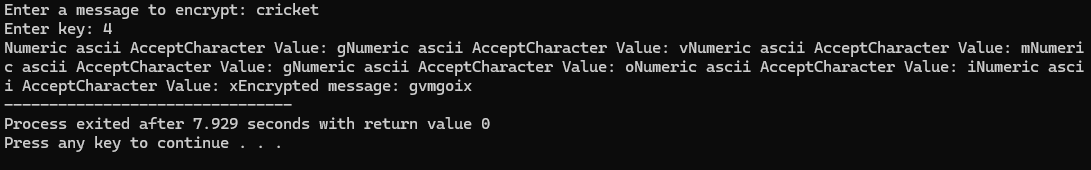
*}*

*printf("Encrypted message: %s", message);*

*return 0;*

*}*

**Output:**

****

**Conclusion:**

In this lab, we successfully implemented a basic encryption process using a simple substitution cipher. By applying a key to shift the characters of a message, we converted plaintext into ciphertext.

**Lab Report: 2 Date:2024/06/02**

**Decryption:**

**Experiment 2:** Implementation of decryption process.

**Objective:**

The objective of this lab report is to demonstrate the practical implementation of decryption techniques. This includes understanding the principles of cryptographic algorithms, utilizing private and public keys for secure data retrieval, and analyzing the effectiveness of decryption methods in maintaining data confidentiality and integrity.

**Theory:**

Decryption is the process of converting encoded or scrambled data (ciphertext) back into its original form (plaintext) using a specific key. This ensures that only authorized users can access the information.

**Code Implementation:**

*int main()*

*{*

*char message[100], ch;*

*int i, key;*

*printf("Enter a message to decrypt: ");*

*gets(message);*

*printf("Enter key: ");*

*scanf("%d", &key);*

*for(i = 0; message[i] != '\0'; ++i){*

*ch = message[i];*

*if(ch >= 'a' && ch <= 'z'){*

*ch = ch - key;*

*if(ch < 'a'){*

*ch = ch + 'z' - 'a' + 1;*

*}*

*message[i] = ch;}*

*else if(ch >= 'A' && ch <= 'Z'){*

*ch = ch - key;*

*if(ch < 'A'){*

*ch = ch + 'Z' - 'A' + 1;*

*}*

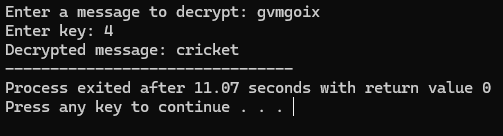
*message[i] = ch;}}*

*printf("Decrypted message: %s", message);*

*return 0;*

*}*

**Output:**

****

**Conclusion:**

In this lab, the decryption process was successfully implemented using a basic substitution cipher. The experiment demonstrated how ciphertext can be converted back into plaintext using a specific key, ensuring secure access to the original message.

**Lab Report: 3 Date:2024/06/25**

**RSA Algorithm:**

**Experiment 3:** Implementation of RSA algorithm.

**Objective:**

The objective of the RSA algorithm is to provide a secure method for encrypting and decrypting data, ensuring confidentiality and integrity. It uses a pair of keys (public and private) for secure communication, enabling authentication, digital signatures, and protection of sensitive information in various applications.

**Theory:**

The RSA algorithm is a widely used asymmetric encryption technique based on the mathematical properties of large prime numbers. It involves generating two keys: a public key for encryption and a private key for decryption. Security relies on the difficulty of factoring the product of two large prime numbers.

**Code Implementation:**

*#include<stdio.h>*

*#include<math.h>*

*int n;*

*int main(){*

*int x;*

*first:*

*printf("Enter first number ");*

*scanf ("%d", &x);*

*if (primecheck(x) == 1){*

*goto first;*

*};*

*int y;*

*second:*

*printf(" \n Enter second number");*

*scanf ("%d", &y);*

*if (primecheck(y) == 1){*

*goto second;*

*};*

*calculaten(x,y);*

*}*

*int primecheck(int x){*

*int i, count = 0;*

*for (i=2; i<=x/2; i++){ // 2 to 5*

*if(x%i==0){*

*count = 1;*

*break;*

*}}*

*if (count == 1 ){*

*printf ("number is not prime \n");*

*return 1;*

*}*

*else{*

*printf ("number is prime \n");*

*} }*

*int calculaten (int x, int y){*

*int tn;*

*n = x\*y;*

*tn = (x-1) \* (y-1);*

*printf("Value of n = %d and tn = %d", n , tn);*

*calculatee(tn);*

*}*

*int calculatee(int tn){*

*int e;*

*esec:*

*printf("Enter any e value \n");*

*scanf("%d", &e);*

*if (e > 1 && e < tn){*

*if (primecheck(e)==1){*

*// condition match*

*goto esec;*

*}// if not match*

*else if (tn%e==0){*

*goto esec;*

*}*

*}else{*

*goto esec;*

*}*

*calculated(e,tn);}*

*int calculated(int e, int tn){*

*//iota(d,string,10); //*

*//e.d (mod t(n)) = 1*

*printf("value of e %d and tn %d", e, tn);*

*int i;*

*for (i=2; i<tn; i++){*

*if ((e\*i)%tn == 1){*

*break;*

*} }*

*int m;*

*printf("Value of d is %d",i);*

*printf("Enter value of message");*

*scanf("%d",&m);*

*ciphertext(m,e);*

*}*

*int ciphertext(int m, int e){*

*//c = (m^e)mod n*

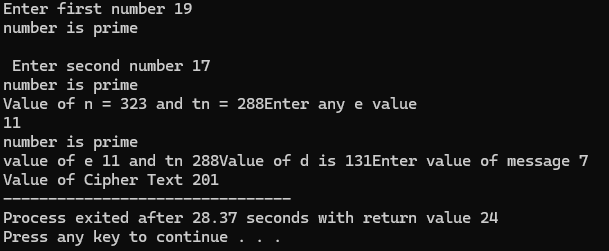
*int po = pow(m,e);*

*int C = po % n;*

*printf ("Value of Cipher Text %d", C);*

*}*

**Output:**

****

**Conclusion:**

In this lab, the RSA algorithm was successfully implemented, demonstrating the process of generating keys, encrypting a message, and computing the ciphertext. The experiment highlighted the mathematical foundations of RSA, including prime number validation, key generation, and modular arithmetic.

**Lab Report: 4 Date:2024/07/10**

**Brute Force:**

**Experiment 4:** Implementation of brute force process.

**Objective:**

The objective of the brute force process in Information Security is to systematically attempt all possible combinations or keys to crack encrypted data or gain unauthorized access.

**Theory:**

Brute force is a method used to break encryption or crack passwords by trying every possible combination until the correct one is found. It relies on computational power to test all potential solutions, making it time-consuming and inefficient for strong encryption systems.

**Code Implementation:**

*import string*

*import itertools*

*target\_password = input("Enter the password you want to crack:")*

*def brute\_force\_password(target):*

*chars = string.ascii\_lowercase*

*for length in range(1, len(target) + 1):*

*for attempt in itertools.product(chars, repeat=length):*

*guess = ''.join(attempt)*

*print(f"Trying: {guess}")*

*if guess == target:*

*return guess*

*return None*

*cracked\_password = brute\_force\_password(target\_password)*

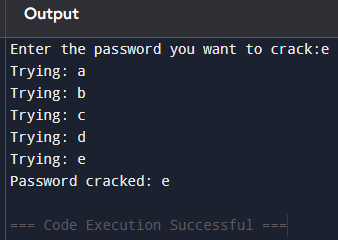
*if cracked\_password:*

*print(f"Password cracked: {cracked\_password}")*

*else:*

*print("Password could not be cracked.")*

**Output:**

****

**Conclusion:**

In this lab, the brute force process was successfully implemented to demonstrate how all possible combinations can be systematically tested to crack a password. While the method is straightforward and guarantees finding the correct password, it is computationally intensive and inefficient for strong passwords with greater complexity.