Practical No. 1

Aim: - To Study Cryptography and its Techniques for Encryption and Decryption

Objective: Students will learn concept of Encryption, Decryption and various techniques to implement it.

Software Required: Turbo C / Dev C++.

Theory:

Definition: Cryptography is associated with the process of converting ordinary plain text into unintelligible text and vice-versa. It is a method of storing and transmitting data in a particular form so that only those for whom it is intended can read and process it. Cryptography not only protects data from theft or alteration, but can also be used for user authentication.

Cryptography is the study of securing communications from outside observers. Encryption algorithms take the original message, or plaintext, and converts it into ciphertext, which is not understandable. The key allows the user to decrypt the message, thus ensuring on they can read the message. The strength of the randomness of an encryption is also studied, which makes it harder for anyone to guess the key or input of the algorithm. Cryptography is how we can achieve more secure and robust connections to elevate our privacy. Advancements in cryptography makes it harder to break encryptions so that encrypted files, folders, or network connections are only accessible to authorized users.

Cryptography focuses on four different objectives:

1. Confidentiality: Confidentiality ensures that only the intended recipient can decrypt the message and read its contents.
2. Non-repudiation: Non-repudiation means the sender of the message cannot backtrack in the future and deny their reasons for sending or creating the message.
3. Integrity: Integrity focuses on the ability to be certain that the information contained within the message cannot be modified while in storage or transit.
4. Authenticity: Authenticity ensures the sender and recipient can verify each other's identities and the destination of the message.

Types of Cryptography

Cryptography can be broken down into three different types:

1. Symmetric Key Cryptography
2. Asymmetric Key Cryptography
3. Hash Functions

Symmetric Key Cryptography, or Secret Key cryptography, uses a single key to encrypt data. Both encryption and decryption in symmetric cryptography use the same key, making this the easiest form of cryptography. The cryptographic algorithm utilizes the key in a cipher to encrypt the data, and when the data must be accessed again, a person entrusted with the secret key can decrypt the data.

Examples: AES, DES, Caesar Cipher

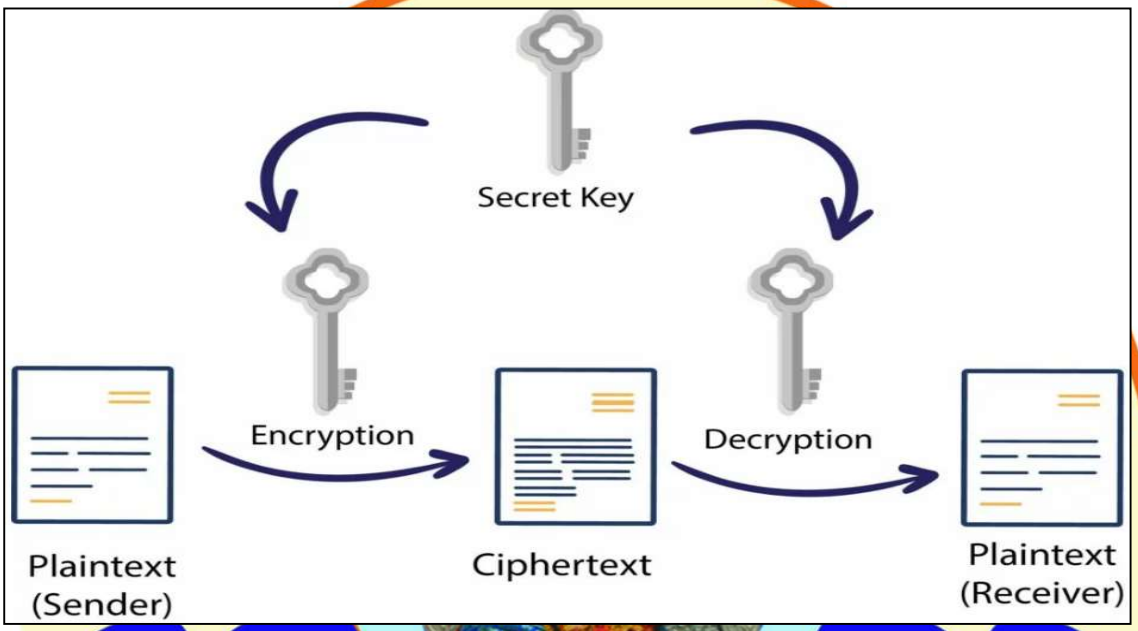


Fig 1.1 Symmetric Key Encryption and Decryption

Asymmetric Key Cryptography or Public Key Cryptography, uses two keys to encrypt data. One is used for encryption, while the other key can decrypt the message. Unlike symmetric cryptography, if one key is used to encrypt, that same key cannot decrypt the message, rather the other key shall be used.

One key is kept private, and is called the "private key", while the other is shared publicly and can be used by anyone, hence it is known as the "public key". The mathematical relation of the keys is such that the private key cannot be derived from the public key, but the public key can be derived from the private. The private key should not be distributed and should remain with the owner only. The public key can be given to any other entity.

Examples: ECC, Diffie-Hellman, DSS.

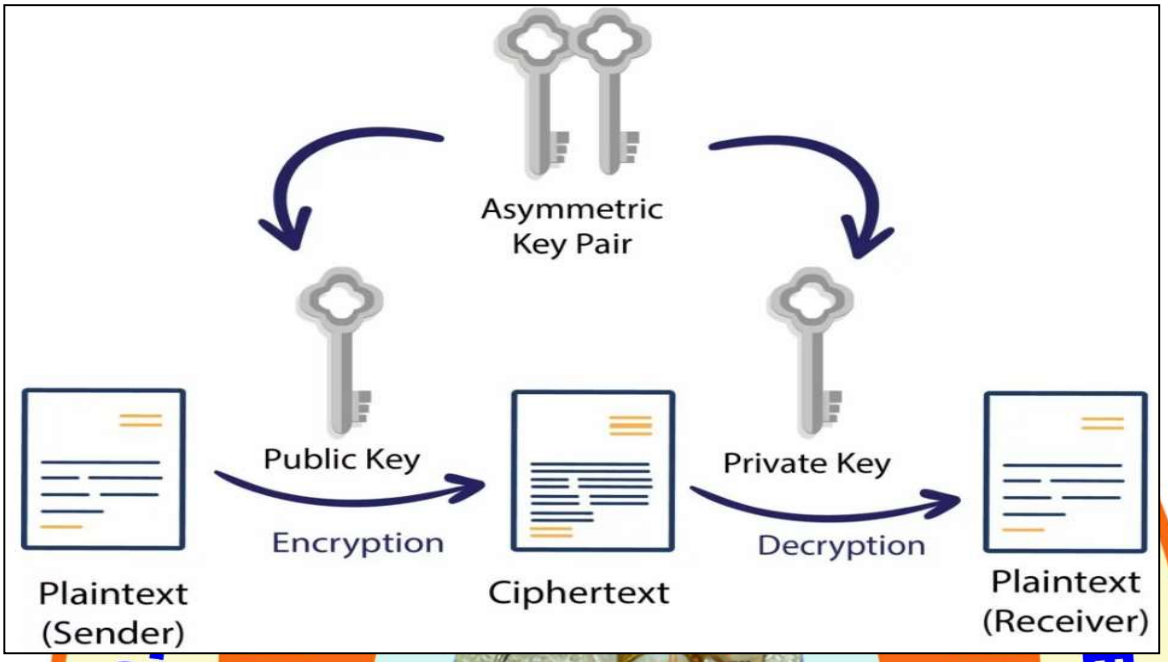


Fig 1.2 Asymmetric Key Encryption Decryption

Hash functions are irreversible, one-way functions which protect the data, at the cost of not being able to recover the original message. Hashing is a way to transform a given string into a fixed length string. A good hashing algorithm will produce unique outputs for each input given. The only way to crack a hash is by trying every input possible, until you get the exact same hash. A hash can be used for hashing data (such as passwords) and in certificates.

Some of the most famous hashing algorithms are: MD5, SHA-1, SHA-2 family which includes SHA-224, SHA-256, SHA-384, and SHA-512, SHA-3.

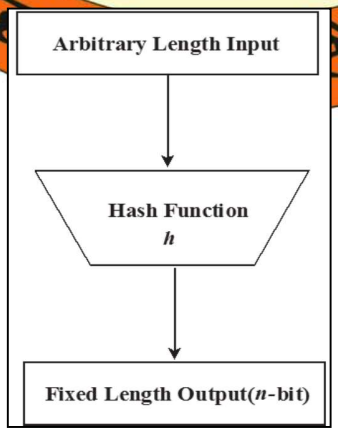


Fig 1.3 Hash Function

Cryptographic Technique flowchart:

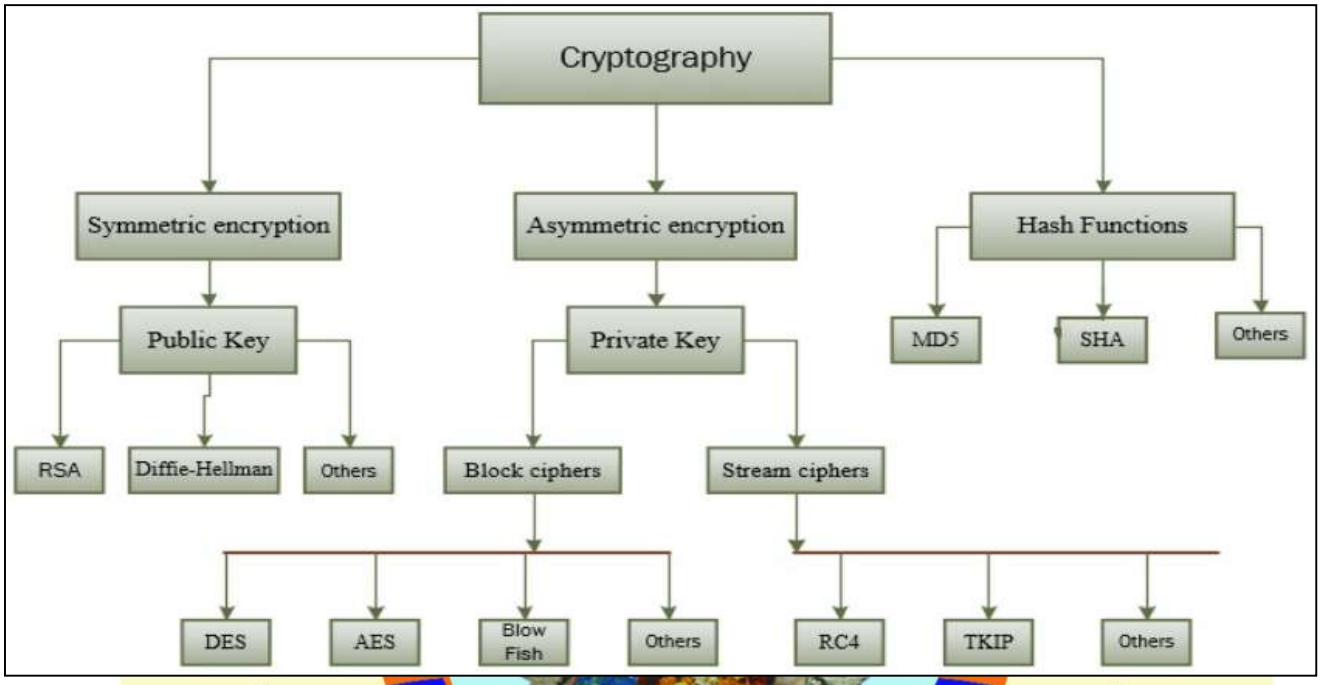


Fig 1.4 Cryptographic Techniques

Conclusion:

Thus, in this way we have studied basics of Encryption, Decryption and different cryptographic techniques.

Viva-vice Questions

1. What is Cryptography?
2. What is Key?
3. What is Cipher Text?
4. What is Encryption and Decryption?
5. What are the examples of encryption and decryption?

ASSESSMENT SCHEME:-

Pre-Lab Test (2)	In Lab performance (5)	Post Lab Test (3)	Record (5)	Total (15)


P. R. POTE PATIL COLLEGE OF ENGINEERING & MANAGEMENT
AMRAVATI
विद्या परम दैवतम

Practical No. 2

Aim: - Write a Program in C to Implement CEASER Cipher.

Objective: Students will learn the concept of encryption using CEASER Cipher Technique.

Software Required: Turbo C / Dev C++.

Theory: -

In the substitution-cipher technique, the characters of a plain-text message are replaced by other characters, numbers or symbols. The Caesar cipher is a special case of substitution technique wherein each alphabet in a message is replaced by an alphabet three places down the line. For instance, using the Caesar cipher, the plain-text ATUL will become cipher-text DWXO.

Algorithm for Caesar Cipher:

Input:

1. A String of lower case letters, called Text.
2. An Integer between 0-25 denoting the required shift.

Procedure:

- Traverse the given text one character at a time .
- For each character, transform the given character as per the rule, depending on whether we're encrypting or decrypting the text.
- Return the new string generated.

The Caesar cipher is the simplest and oldest method of cryptography. The Caesar cipher method is based on a mono-alphabetic cipher and is also called a shift cipher or additive cipher. Julius Caesar used the shift cipher (additive cipher) technique to communicate with his officers. For this reason, the shift cipher technique is called the Caesar cipher. The Caesar cipher is a kind of replacement (substitution) cipher, where all letter of plain text is replaced by another letter.

Let's take an example to understand the Caesar cipher, suppose we are shifting with 1, then A will be replaced by B, B will be replaced by C, C will be replaced by D, D will be replaced by C, and this process continues until the entire plain text is finished.

Caesar ciphers is a weak method of cryptography. It can be easily hacked. It means the message encrypted by this method can be easily decrypted.

Algorithm

1. Declare two character arrays p and c of size 40 to store the plain text and cipher text.
2. Declare an integer variable key to store the encryption key.
3. Display a message to the user asking them to input the plain text that they want to encrypt using the CEASER cipher without any spaces.
4. Read the input plain text from the user using scanf() and store it in the array p.
5. Display a message to the user asking them to input the encryption key.
6. Read the input key from the user using scanf() and store it in the variable key.
7. Calculate the length of the plain text and store it in the integer variable n using the strlen() function.
8. Encrypt the plain text into cipher text using the following steps: a. For each character in the plain text (using a loop from i = 0 to n-1): i. Replace the ith character in the cipher text with the ith character in the plain text plus the encryption key. b. Display the encrypted \ cipher text.
9. Decrypt the cipher text into plain text using the following steps: a. For each character in the cipher text (using a loop from i = 0 to n-1): i. Replace the ith character in the plain text with the ith character in the cipher text minus the encryption key. b. Display the decrypted plain text.
10. End of the algorithm.

Program:-

```
#include<stdio.h>
#include<string.h>
void main()
{
 char p[40],c[40];
 int key,i,n;
printf("Practical No.2\nC Program to implement CEASER Cipher\n");
printf("\nEnter Plain Text that you want to encrypt using CEASER Cipher without space\n");
scanf("%s",p);
printf("\nEnter Key:");
scanf("%d",&key);
 n=strlen(p);
 for(i=0;i<n;i++)
   {
      c[i]=p[i]+key;
   }
  printf("\n Encrypted Text is:   %s",c);
 for(i=0;i<n;i++)
   {
      p[i]=c[i]-key;
```

```
    }
  printf("\nDecrypted Text is:   %s",p);
}
```

Output:

```
Practical No.2
C Program to implement CEASER Cipher
Enter Plain Text that you want to encrypt using CEASER Cipher without space
Cryptography
Enter Key:4
Encrypted Text is:   Gv}txskvetl}
Decrypted Text is:   Cryptography
```

Conclusion:

Thus, in this way we have studied working of simple and modified CEASER cipher and implemented it in C.

Viva-vice Questions

1. What is value of key in CEASER Cipher?
2. What is concept of Modified CEASER Cipher?
3. How Decryption of CEASER Cipher is done?
4. What will be the value of C if Key=5 in Encryption?
5. What will be the value of A if Key=7 in Decryption?

ASSESSMENT SCHEME:-

Pre-Lab Test (2)	In Lab performance (5)	Post Lab Test (3)	Record (5)	Total (15)

Practical No. 3

Aim: - Write a Program in C to implement Railfence Algorithm.

Objective: Students will understand how Rail fence algorithm works and how encryption is done in Rail Fence algorithm.

Software Required: Turbo C / Dev C++.

Theory: -

The rail fence cipher (sometimes called zigzag cipher) is a transposition cipher that jumbles up the order of the letters of a message using a basic algorithm.

The rail fence cipher works by writing your message on alternate lines across the page, and then reading off each line in turn.

For example, let's consider the plaintext **"This is a secret message".**

Plaintext	T	H	I	S	I	S	A	S	E	C	R	E	T	M	E	S	S	A	G	E

To encode this message we will first write over two lines (the "rails of the fence") as follows:

Rail Fence	T		I		I		A		E		R		T		E		S		G	
Encoding		H		S		S		S		C		E		M		S		A		E

Note that all white spaces have been removed from the plain text.

The ciphertext is then read off by writing the top row first, followed by the bottom row:

Ciphertext	T	I	I	A	E	R	T	E	S	G	H	S	S	S	C	E	M	S	A	E

Algorithm for rail-fence cipher are as follow:

1. Write down the plain text message as a sequence of diagonals.
2. Read the plain text written in step 1 , row wise.
3. Let's see example of rail-fence cipher. Suppose plain text is **CRYPTOGRAPHY** if we perform rail-fence cipher operation on this text it will be coded as **CYTGAHRPORPY**.
4. It involves writing plain text in a diagonal sequence and then reading it row by row to produce cipher text.
5. It is quite clear that this technique is simple for cryptanalyst to break into.

Algorithm:-

1. Start
2. Give the message “Cryptography Practical No : 3\nC Program to Implement Rail Fence Cipher\n”.
3. Declare two character arrays p and even, odd of size 20 to store the plain text and cipher text.
4. Declare an integer variable key to store the encryption key.
5. Read the input plain text form the user using scanf() and store it in the array p.
6. Calculate the length of the plain text and store it in the integer variable n using the strlen() function.
7. Encrypt the plain text into cipher test using the following steps: a. For each character in the plain text (using a loop from i = 0 to n-1): i. Replace the ith character in the cipher text with the ith character in the plain text plus the encryption key. b.Display the encrypted cipher text.
8. Decrypt the cipher text into plain test using the following steps: a. For each character in the cipher text (using a loop from i = 0 to n-1): i. Replace the ith character in the plain text with the ith character in the plain text plus the encryption key. b.Display the encrypted plain text.
9. End of the algorithm

Program:

```
#include<stdio.h>
#include<string.h>
int main()
{
   printf("Cryptography Practical No : 3\nC Program to Implement Rail Fence Cipher\n");
        char p[20],even[20],odd[20];
        int n,i,j=0,k=0;
        printf("Enter a plain text: ");
        gets(p);
        n=strlen(p);
        for(i=0;i<n;i++)
        {
                if(i%2==0)
                {
                        even[j]=p[i];
                        j++;
                }
                else
                {
                        odd[k]=p[i];
                        k++;
                }
        }
```

```
        printf("Encrypted text after Encryption is : ");
        printf(even);
        printf(odd);
}
```

Output:

Cryptography Practical No : 3

C Program to Implement Rail Fence Cipher

Enter a plain text: Welcome to Cryptography

Encrypted text after Encryption is : Wloet rporpyecm oCytgah

Conclusion:

Thus, in this way we have studied working of Rail Fence cipher Technique and implemented it in C.

Viva Questions:

1. Rail fence cipher is an example of which cipher?
2. What is the alternative name given to Rail fence cipher?
3. The number of columns in the table used for encryption in rail fence cipher depends upon the given key value. TRUE or FALSE?
4. What will be the plain text corresponding to cipher text “ELLECTIVE” if rail fence cipher is used with key value 1?
5. What will be the ciphered text if rail fence cipher is used for encrypting the plain text “CRYPTOGRAPHY” with the key value given to be 2?

ASSESSMENT SCHEME:-

Pre-Lab Test (2)	In-Lab performance (5)	Post-Lab Test (3)	Record (5)	Total (15)

Practical No. 4

Aim: - Write a Program in C to implement Mono Alphabetic Substitution Cipher.

Objective: Students will learn Substitution cipher technique and will learn working of monoalphabetic cipher.

Software Required: Turbo C / Dev C++.

Theory: -

A monoalphabetic substitution cipher is a type of encryption that replaces each letter of the plaintext with a corresponding letter from a fixed alphabet. The fixed alphabet used for substitution is called the key. The key is typically a permutation of the standard alphabet, where each letter is replaced by a unique letter from the key.

For example, if the key is "LFWOAYUISVKMNXPBDCRJTQEGHZ", then the plaintext letter 'A' would be replaced with 'L', 'B' with 'F', and so on.

The key must be kept secret to maintain the security of the cipher.

Monoalphabetic substitution ciphers are vulnerable to frequency analysis attacks, where the frequency of each letter in the ciphertext is compared to the expected frequency of that letter in the plaintext. This vulnerability can be mitigated by using a more complex substitution method, such as a polyalphabetic cipher.

Algorithm for a monoalphabetic substitution cipher:

1. Choose a secret key, which is a permutation of the 26 letters of the alphabet.
2. Map each letter of the plaintext to the corresponding letter in the key.
3. Generate the ciphertext by replacing each plaintext letter with its corresponding key letter.
4. Send the ciphertext to the recipient.
5. To decrypt the ciphertext, the recipient must use the same key to reverse the substitution and recover the plaintext.

Here is an example of how the algorithm works:

Secret key: ABCDEFGHIJKLMNOPQRSTUVWXYZ

Plaintext: HELLO WORLD

Mapping: H -> A, E -> B, L -> C, O -> D, W -> E, R -> F, and so on.

Ciphertext: ABCDD CEBDE

To decrypt the ciphertext, the recipient uses the same key to reverse the mapping and recover the plaintext: HELLO WORLD.

Program:-

```
#include<stdio.h>
#include<string.h>
void main()
{
  char pt[26]={'A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z'};
  char ct[26]={'Z','Y','X','W','V','U','T','S','R','Q','P','O','N','M','L','K','J','I','H','G','F','E','D','C','B','A'};
   char p[20]={'\0'},c[20]={'\0'},r[20]={'\0'};
        int i,j;
        printf("\nCryptography Practical No. 4");
        printf("\nC Program to implement MonoAlphabetic Substitution Cipher\n");
        printf("\n Enter the plain text in CAPITAL Letter without space\n");
        gets(p);
        for(i=0;i<strlen(p);i++)          //converting plain text into cipher text (encryption)
        {
                for(j=0;j<26;j++)
                {
                        if(pt[j]==p[i])
                        {
                                c[i]=ct[j];
                        }
                }
        }
        printf("\n Cipher text after Encryption is: %s",c);
        for(i=0;i<strlen(c);i++) //converting cipher text into plain text (decryption)
        {
                for(j=0;j<26;j++)
                {
                        if(ct[j]==c[i])
                        {
                        r[i]=pt[j];
                        }
                }
        }
        printf("\nPlain text after Decryption is: %s",r);
}
```

Output:

Cryptography Practical No. 4

C Program to implement MonoAlphabetic Substitution Cipher

Enter the plain text in CAPITAL Letter without space

CRYPTOGRAPHY

Cipher text after Encryption is: XIBKGLTIZKSB

Plain text after Decryption is: CRYPTOGRAPHY

Conclusion:

Thus, in this way we have studied working of monoalphabetic substitution cipher technique and implemented it in C.

Viva-vice Questions

1. What is the Logic behind mono alphabetic substitution cipher?
2. Why it is called Substitution cipher?
3. Can we assign any substitution to Alphabets?
4. What is Transposition Cipher Technique
5. Give any example of Transposition cipher Technique.

ASSESSMENT SCHEME:

-

Pre-Lab Test (2)	In Lab performance (5)	Post Lab Test (3)	Record (5)	Total (15)

Practical No. 5

Aim: - Write a Program in C to Implement Simple Columnar transposition technique.

Objective: Students will understand how simple columnar transposition technique work and how encryption is done in it.

Software Required: Turbo C / Dev C++.

Theory: -

A columnar transposition cipher is a type of encryption that rearranges the plaintext letters based on a fixed columnar structure.

The plaintext is written out in rows of fixed length, and then rearranged into columns based on a secret key.

The ciphertext is generated by reading the columns from left to right and top to bottom.

Here is a simple algorithm for a columnar transposition cipher:

1. Choose a secret key, which is a permutation of the numbers 1 to the length of the plaintext.
2. Write the plaintext out in rows of fixed length, filling any remaining spaces with a dummy character.
3. Rearrange the rows based on the alphabetical order of the key. If there are ties, use the order of the tied keys.
4. Read the columns from left to right and top to bottom to generate the ciphertext.

Here is an example of how the algorithm works:

Secret key: 3142

Plaintext: ATTACKATDAWN

Rows: ATTAC KATDA WN

Rearranged rows: ATTA CKAT DAWN

Columns: AAKW TTNA TCD

Ciphertext: AAKWTTNATCD

To decrypt the ciphertext, the recipient must know the secret key and reverse the process by writing the ciphertext into columns based on the order of the key, then rearranging the rows back into their original order to recover the plaintext.

Program:

```
#include<stdio.h>
#include<string.h>
void main()
{
char p[30]={'\0'},c[30]={'\0'},m[30][30]={'\0'};
int i,j,n,s,r,t=0,k=0,key[5];
printf("\nCryptography Practical No. 5\nTo Implement Simple Columnar transposition  Technique");
printf("\nEnter the Plain Text to encrypt:");
gets(p);
printf("Enter The Key Sequence(0-4):");
for(i=0;i<5;i++)
scanf("%d",&key[i]);
n=strlen(p);
r=n/5+1;
for(i=0;i<r;i++)
{
        for(j=0;j<5;j++)
        {
                m[i][j]=p[k];
                k++;
        }
}
for(i=0;i<5;i++)
{
        s=key[i];
        for(j=0;j<r;j++)
        {
                if(m[j][s]!='\0')
                { c[t]=m[j][s];
                        t++;
                }
        }
}
printf("Ciper text is :%s",c);
}
```

Output:

Cryptography Practical No. 5

To Implement Simple Columnar Transposition Technique

Enter the Plain Text to encrypt:Welcome to Cryptography

Enter The Key Sequence(0-4):3 2 4 1 0

Ciper text is :ctyrl rgyoopaeeCohWm tp

Conclusion:

Thus, in this way we have implemented Simple columnar transposition technique in c.

Viva Questions:

1. Columnar cipher falls under the category of?
2. What should be the sequence of column?
3. Encrypt plain text “Cryptography” using simple columnar technique.
4. What happens when we enter plain text in matrix and some memories are left?
5. How to overcome above problem?

ASSESSMENT SCHEME:-

Pre-Lab Test (2)	In Lab performance (5)	Post Lab Test (3)	Record (5)	Total (15)

Practical No. 6

Aim: - Write a Program in C to implement Deffi-Hellman Key Exchange Algorithm

Objective: Students will study symmetric key exchange concept with the help of Deffi-Hellman key exchange algorithm.

Software Required: Turbo C / Dev C++.

Theory: -

The Diffie-Hellman key exchange algorithm is a method for two parties to agree on a shared secret key over an insecure communication channel. The algorithm was developed by Whitfield Diffie and Martin Hellman in 1976.

Algorithm of the Diffie-Hellman key exchange algorithm:

1. Alice and Bob agree on a public modulus p and a base g, where p and g are both prime numbers.
2. Alice chooses a secret integer a, computes A = g^a mod p, and sends A to Bob.
3. Bob chooses a secret integer b, computes B = g^b mod p, and sends B to Alice.
4. Alice computes s = B^a mod p, which is the shared secret key.
5. Bob computes s = A^b mod p, which is the same shared secret key.

Both Alice and Bob now have the same shared secret key s, which they can use to encrypt and decrypt messages. An eavesdropper who intercepts the values A and B and the public parameters p and g cannot easily compute the shared secret key without solving the discrete logarithm problem.

The Diffie-Hellman key exchange algorithm is widely used in many cryptographic protocols, such as SSL/TLS for secure web browsing and PGP for secure email communication.

Program:-

```
//C Program to implement Deffi Hellman key exchange algorithm
#include <stdio.h>
int compute(int a, int m, int n) // Function to compute `a^m mod n`
{
   int r;
   int y = 1;
   while (m > 0)
   {
      r = m % 2;
      if (r == 1)
      {
         y = (y*a) % n;
      }
      a = a*a % n;
      m = m / 2;
   }
```

```
    return y;
}
int main()
{
    int p = 23;        // modulus
    int g = 5;        // base
    int a, b;    // `a` – Alice's secret key, `b` – Bob's secret key.
    int A, B;    // `A` – Alice's public key, `B` – Bob's public key
    a = 6;     // choose a secret integer for Alice's private key (only known to Alice)
    A = compute(g, a, p); // Calculate Alice's public key (Alice will send `A` to Bob)
    b = 15;     // choose a secret integer for Bob's private key (only known to Bob)
    B = compute(g, b, p);     // Calculate Bob's public key (Bob will send `B` to Alice)
    // Alice and Bob Exchange their public key `A` and `B` with each other
    // Find secret key
    int keyA = compute(B, a, p);
    int keyB = compute(A, b, p);
    printf("Cryptography Practical No.5");
    printf("\nC Program to generate KEY using Deffi-Hellman Key Exchange Algorithm\n");
    printf("\nAlice`s Public Key is %d",A);
    printf("\nBob`s Public Key is %d",B);
    printf("\nAlice's secret key is %d\nBob's secret key is %d", keyA, keyB);
    return 0;
}
```

Output:

Cryptography Practical No.5

C Program to generate KEY using Deffi-Hellman Key Exchange Algorithm

Alice`s Public Key is 8

Bob`s Public Key is 19

Alice's secret key is 2

Bob's secret key is 2

Conclusion:

Thus, In this way we have studied and implement Symmetric key algorithm i.e Deffi-Hellman Key Exchange Algorithm

Viva-vice Questions

1. How public key is calculated?
2. What is symmetric key algorithm?
3. What is asymmetric key algorithm?
4. How secret key is generated?
5. Is the generated secret key same for both sender and receiver?

ASSESSMENT SCHEME:-

Pre-Lab Test (2)	In Lab performance (5)	Post Lab Test (3)	Record (5)	Total (15)

Practical No. 7

Aim: - Write a Program in C to implement RSA Algorithm.

Objective: Students will understand how RSA works and how encryption and decryption is done in RSA.

Software Required: Turbo C / Dev C++.

Theory: -

The RSA algorithm is a widely used public-key encryption algorithm. It was invented by Ron Rivest, Adi Shamir, and Leonard Adleman in 1977 and is named after their surnames.

The RSA algorithm works by using two keys, a public key and a private key. The public key can be freely distributed and is used for encrypting messages, while the private key is kept secret and is used for decrypting the encrypted messages.

The security of RSA relies on the difficulty of factoring large composite numbers.

The algorithm works as follows:

Key generation:

1. Two large prime numbers, p and q, are chosen.
2. Their product, n = pq, is computed and kept secret.
3. The Euler totient function $\varphi(n) = (p-1)(q-1)$ is also computed.

Public key creation:

4. A number e is chosen such that $1 < e \leq \varphi(n)$ and e is coprime to $\varphi(n)$. The public key is the pair (n,e).

Private key creation:

5. The private key is the pair (n,d), where d is the multiplicative inverse of e modulo $\varphi(n)$. In other words, d is the unique integer such that $ed \equiv 1 \pmod{\varphi(n)}$.

Encryption:

6. To encrypt a message m, the sender computes $c \equiv m^e \pmod{n}$ and sends the ciphertext c to the recipient.

Decryption:

7. To decrypt the ciphertext c, the recipient computes $m \equiv c^d \pmod{n}$.

The security of RSA depends on the difficulty of factoring large composite numbers, as mentioned earlier. If an attacker can factor n, they can compute $\varphi(n)$ and hence compute the private key d from the public key e. Therefore, the security of RSA relies on the assumption that factoring large composite numbers is computationally difficult.

Algorithm

1. Define a function checkPrime(n) that takes a positive integer n as input and returns 1 if n is prime and 0 otherwise.
2. Define a function findfactor(n1, n2) that takes two positive integers n1 and n2 as input and returns their greatest common divisor.
3. Define a function powMod(a, b, n) that takes three positive integers a, b and n as input and returns a^b mod n.
4. In the main function, declare the following variables:
   - Two prime numbers p and q.
   - A temporary variable temp that is equal to (p-1)*(q-1).
   - Two integers e and d, which will be used as public and private keys.
   - An integer n that is equal to p*q.
   - Three integers data, cipher, and decrypt, which will be used to represent the plaintext, ciphertext, and decrypted text.
5. Prompt the user to enter two prime numbers p and q.
6. Check if both p and q are prime numbers using the checkPrime function.
7. Calculate n = pq and temp = (p-1)(q-1).
8. Find a value for e such that e is relatively prime to temp, i.e., gcd(temp, e) = 1, using the findfactor function.
9. Find a value for d such that d*e mod temp = 1.
10. Print out the public key e and private key d.
11. Prompt the user to enter the data to be encrypted.
12. Calculate the cipher text by calling the powMod function with data, e, and n as input.
13. Print out the cipher text.
14. Calculate the decrypted text by calling the powMod function with cipher, d, and n as input.
15. Print out the decrypted text.
16. End.

Program:-

//implementation of RSA Algorithm

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int checkPrime(int n)
{
        int i;
        int m = n / 2;
        for (i = 2; i <= m; i++)
        {
                if (n % i == 0)
                {
                        return 0; // Not Prime
                }
        }
        return 1; // Prime
}

int findfactor(int n1, int n2)
{
        int i, gcd;
        for(i = 1; i <= n1 && i <= n2; ++i)
        {
                if(n1 % i == 0 && n2 % i == 0)
                        gcd = i;
        }
        return gcd;
}
int powMod(int a, int b, int n)
{
        int x = 1, y = a;
        while (b > 0)
        {
                if (b % 2 == 1)
                        x = (x * y) % n;
                y = (y * y) % n; // Squaring the base
                b /= 2;
        }
        return x % n;
}

int main()
{
        int p, q;
        int n, temp;
```

```
    int data, cipher, decrypt;
    while (1)
    {
    printf("\nCryptography Practical No.6\nC Program to implement RSA Algorithm\n");
        printf("Enter any two prime numbers: ");
        scanf("%d %d", &p, &q);
        if (!(checkPrime(p) && checkPrime(q)))
        printf("Both numbers are not prime. Please enter prime numbers only...\n");
        else if (!checkPrime(p))
        printf("The first prime number you entered is not prime, please try again...\n");
        else if (!checkPrime(q))
        printf("The second prime number you entered is not prime, please try again...\n");
        else
            break;
    }
    n = p * q;
    temp = (p - 1) * (q - 1);
    int e = 0;
    for (e = 2; e <= 100; e++)
    {
        if (findfactor(temp, e) == 1)
            break;
    }
    int d = 0;
    for (d = e + 1; d <= 100; d++)
    {
        if ( ((d * e) % temp) == 1)
            break;
    }
    printf("Public Key e is: %d\nPrivate Key d is: %d\n", e, d);
    printf("Enter data to be encrypted in number ");
    scanf("%d", &data);
    cipher = powMod(data, e, n);
    printf("The cipher text is: %d\n", cipher);
    decrypt = powMod(cipher, d, n);
    printf("The decrypted text is: %d\n", decrypt);
    return 0;
}
```

Output:

Cryptography Practical No.6

C Program to implement RSA Algorithm

Enter any two prime numbers: 7 17

Public Key e is: 5

Private Key d is: 77

Enter data to be encrypted in number 10

The cipher text is: 40

The decrypted text is: 10

Conclusion:

In this way we have studied how RSA algorithm works and how to implement it.

Viva-vice Questions

1. What is the full form of RSA?
2. How to choose Private key?
3. How to choose public key?
4. What is the formula for encrypting text?
5. What is the formula for decrypting text?

ASSESSMENT SCHEME:-

Pre-Lab Test (2)	In Lab performance (5)	Post Lab Test (3)	Record (5)	Total (15)

Practical No. 8

Aim: - Write a procedure to encrypt word/ PDF/ Image/ Any other document using password protection scheme (either default or online tool)_____.

Objective: Students will understand how to encrypt word and pdf document by giving it password

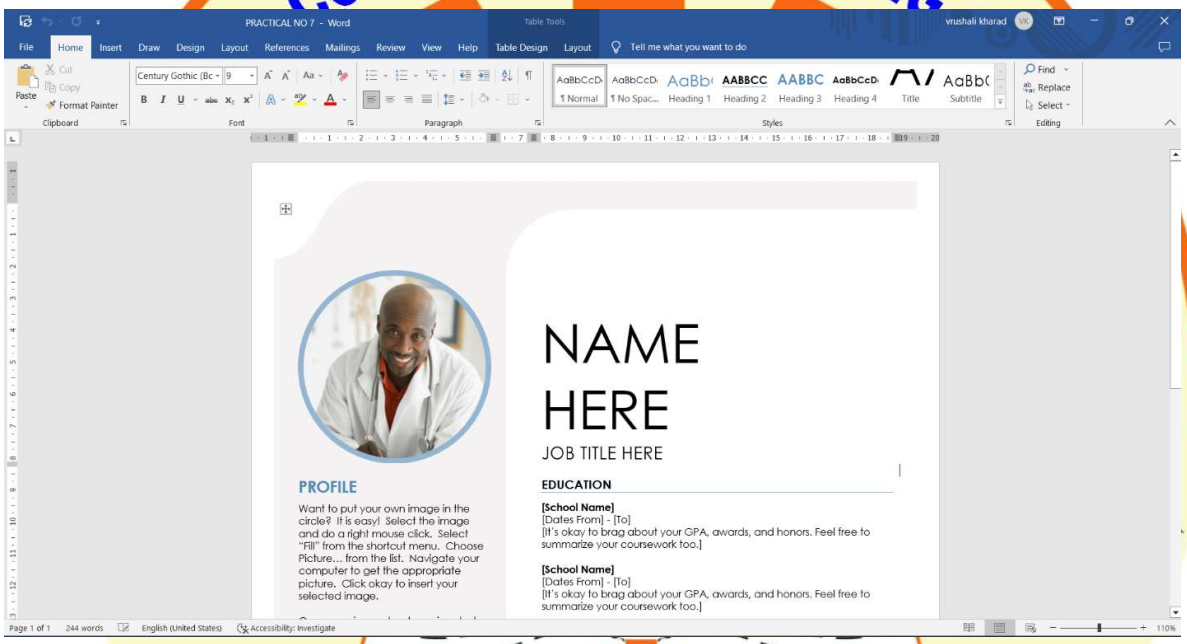
Software Required: Windows any 8/9/10.

Theory:

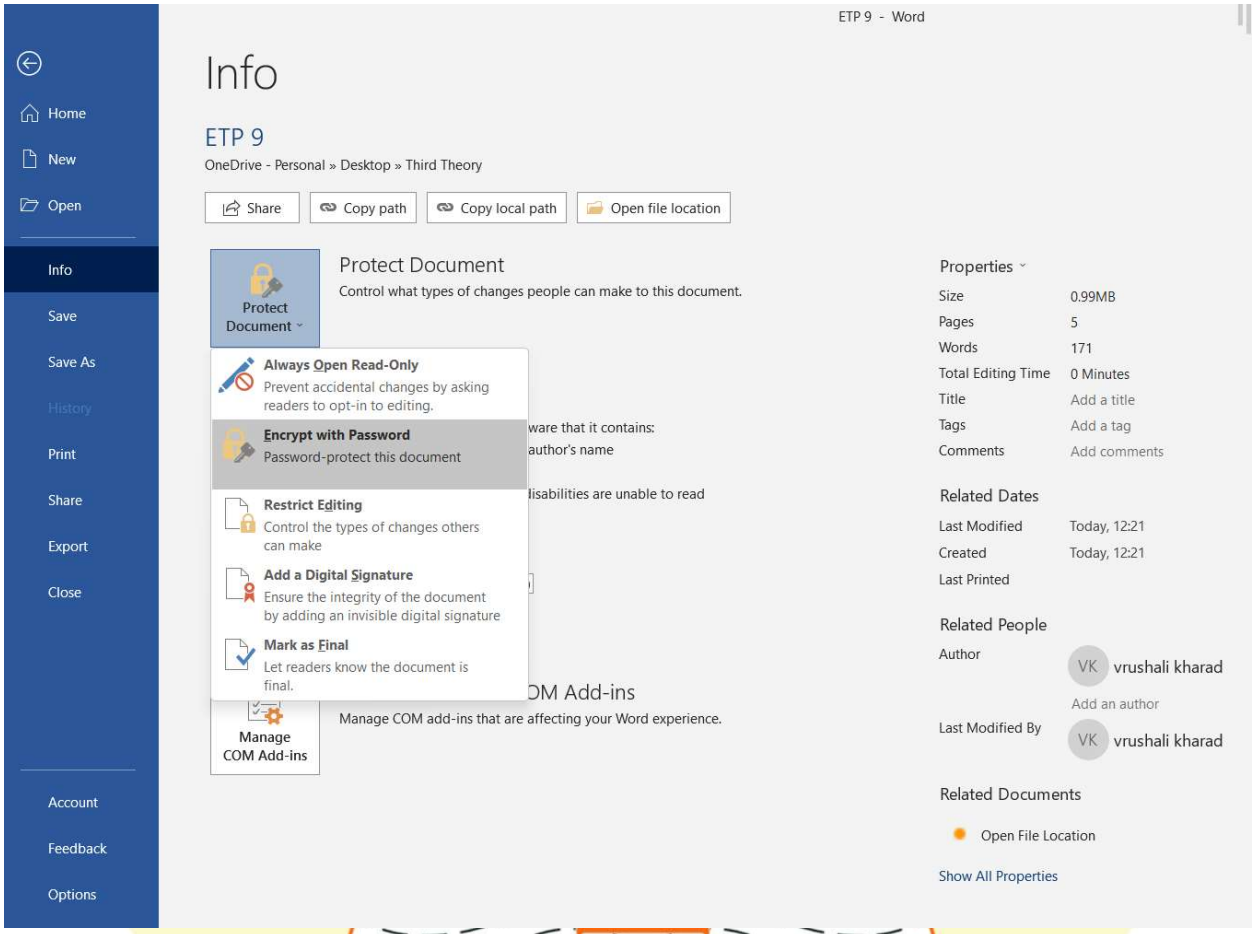
Screenshots:

1. Open/ write word document.

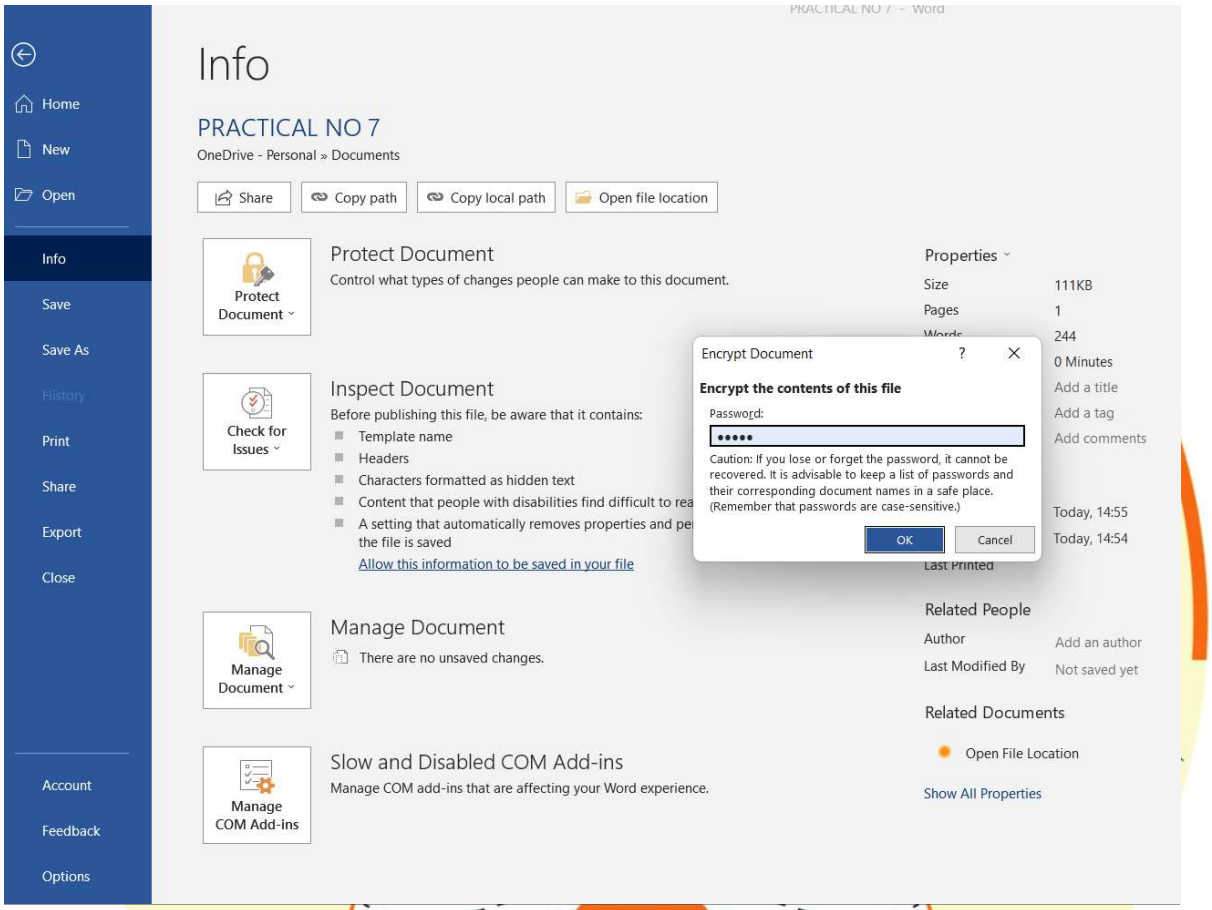
Here I have to create a resume, so I have created a sample resume document.



2. In second step go to FILE and click on INFO tab. Under it is protect document tab, there click on Protect Document and you will see dropdown list. Click on Encrypt with password tab.

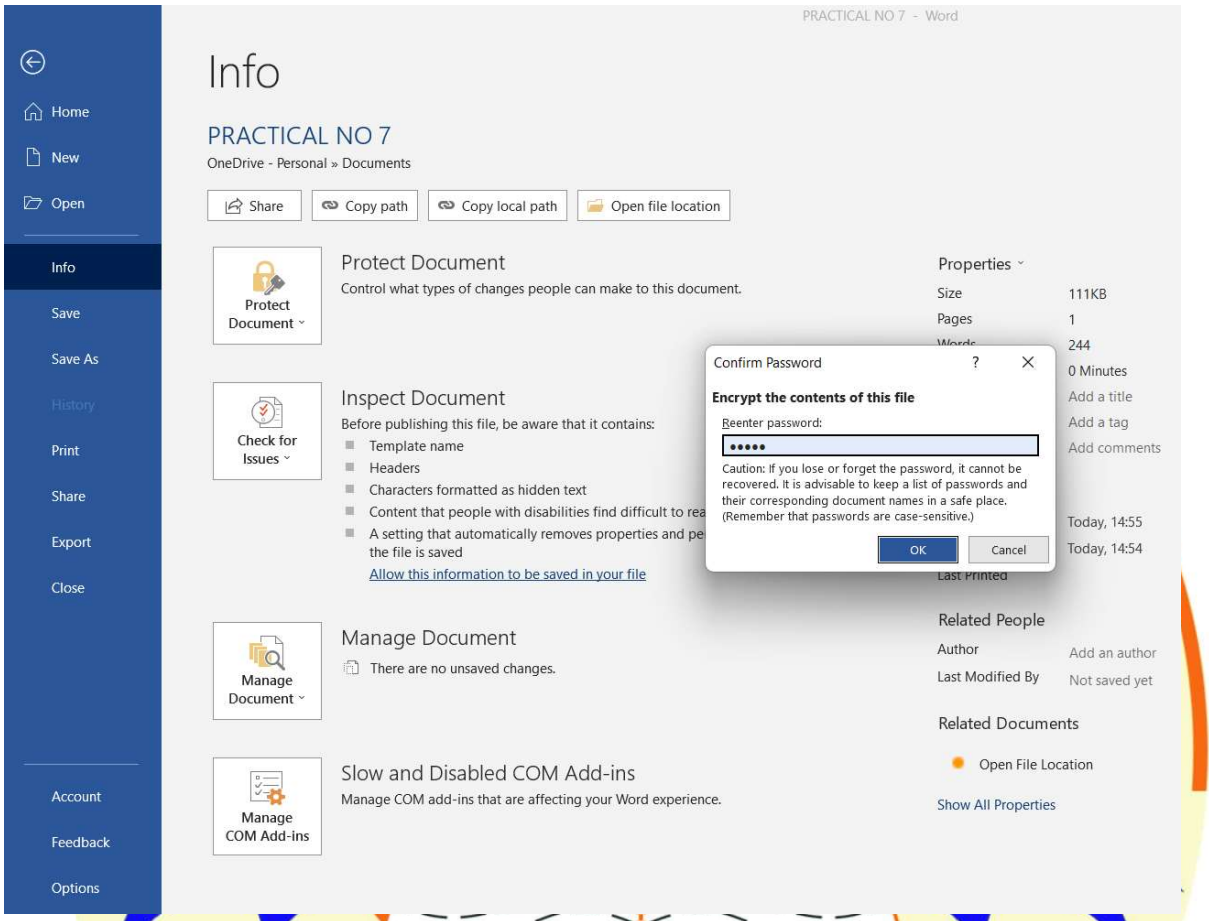


3. In step 3 you will get a popup screen showing contents “Encrypt the contents of this file”. Then enter password of your choice.

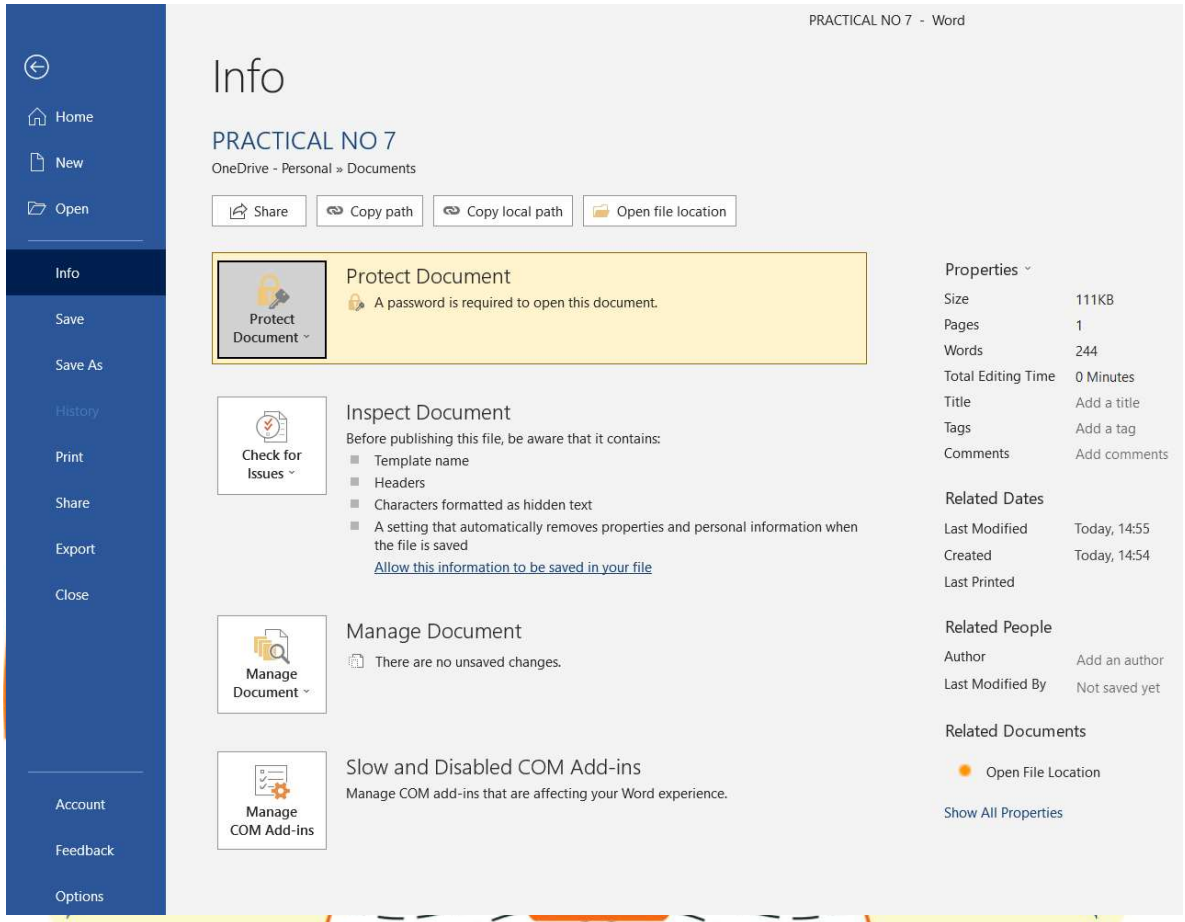


4. In step 4 you will need to reenter password for confirmation.

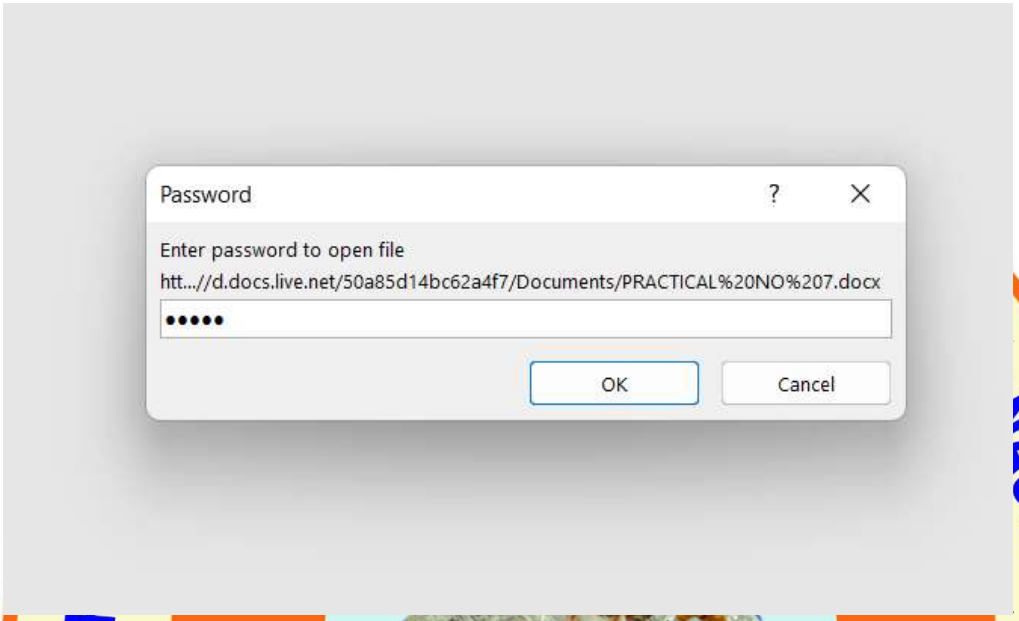
5.



6. In step 5 after clicking ok on reenter password window. “Protect document” tab will become yellow and the line written on it will change to ”A Password is required to open this document”.



7. In Step 6 Go to the file where it is stored and double click on it to open it.
   You will get the window asking to enter the password to open this document.



Now just enter the password to open the document.

Conclusion:

Thus, in this way we have studied how to encrypt document using default available password protection scheme.

Viva Questions:

1. What are the default techniques to encrypt any document?
2. List some online tools that provide encryption scheme.
3. How have you done encryption of your document?
4. Is there any opensource software/ tool available that can encrypt image in any format?
5. How to decrypt encrypted document?

ASSESSMENT SCHEME:-

Pre-Lab Test (2)	In Lab performance (5)	Post Lab Test (3)	Record (5)	Total (15)

Practical No. 9

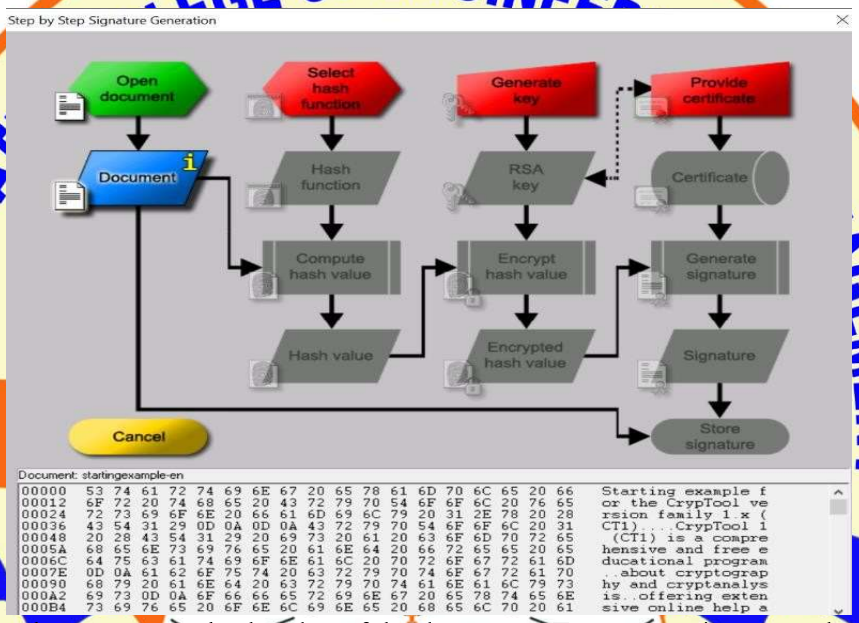
Aim: - Write a procedure to create digital signature using any online tool available.

Objective: Students will understand and learn how to use different online tool to create digital signature and how to use it to sign the document.

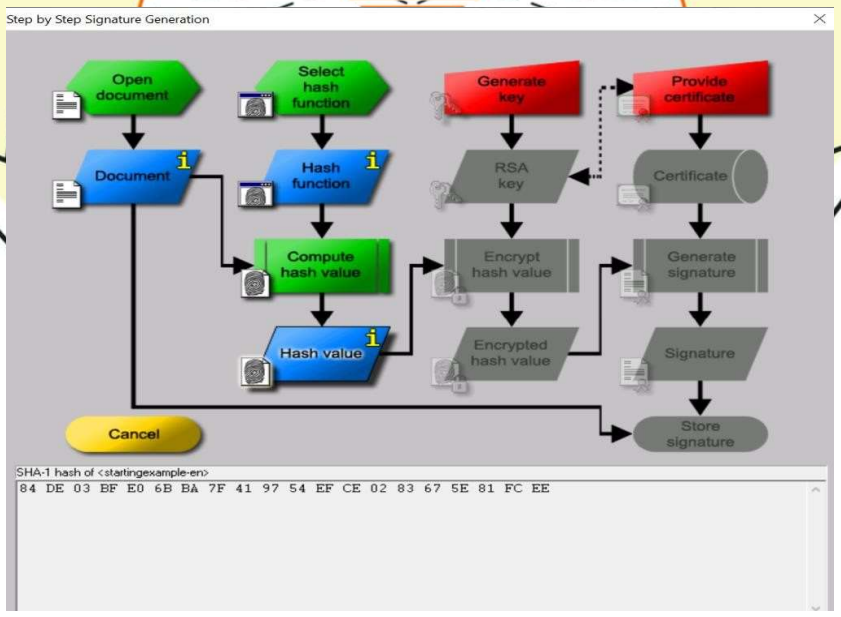
Software Required: Any browser, Any operating system

Procedure :

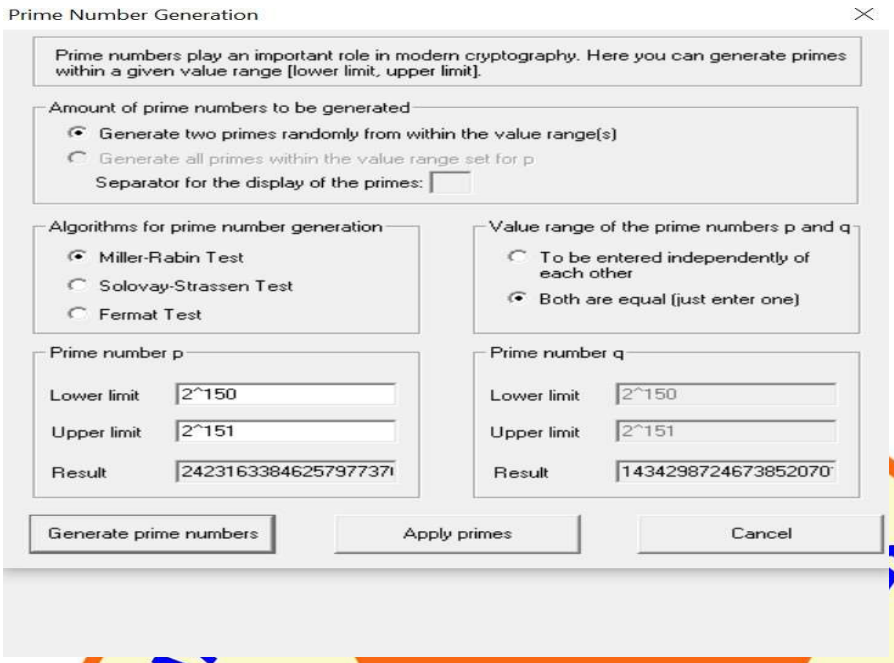
- **Digital Signature Creation :**



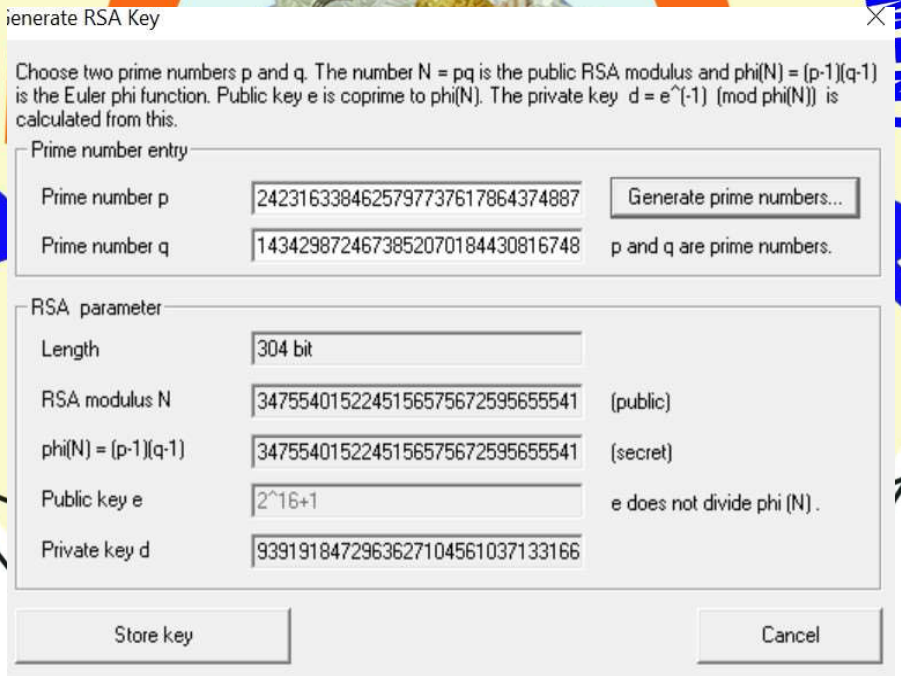
- First we need to generate a hash value of the document. To generate it, we need to select a hashing algorithm. We'll use the SHA-1 algorithm.



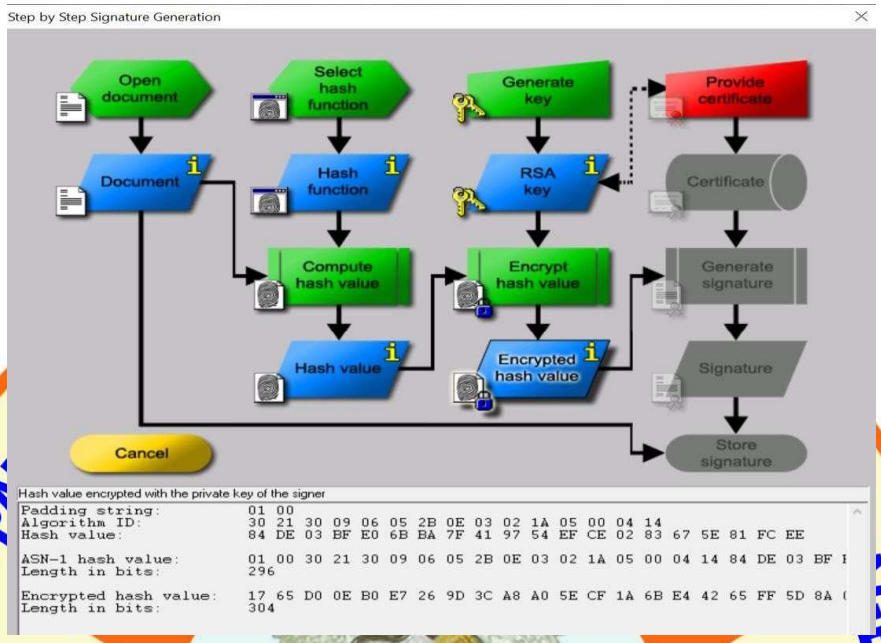
- Next, generate a key pair. We'll generate RSA keys. For RSA key generation, two large prime numbers and a mathematical function are required.



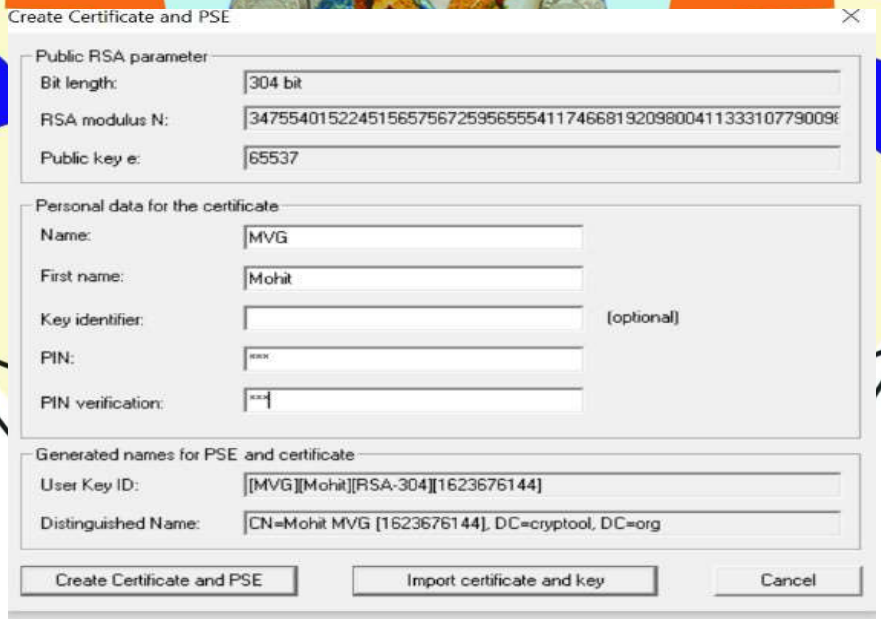
➢ After successfully generating keys, encrypt the hash value generated earlier.



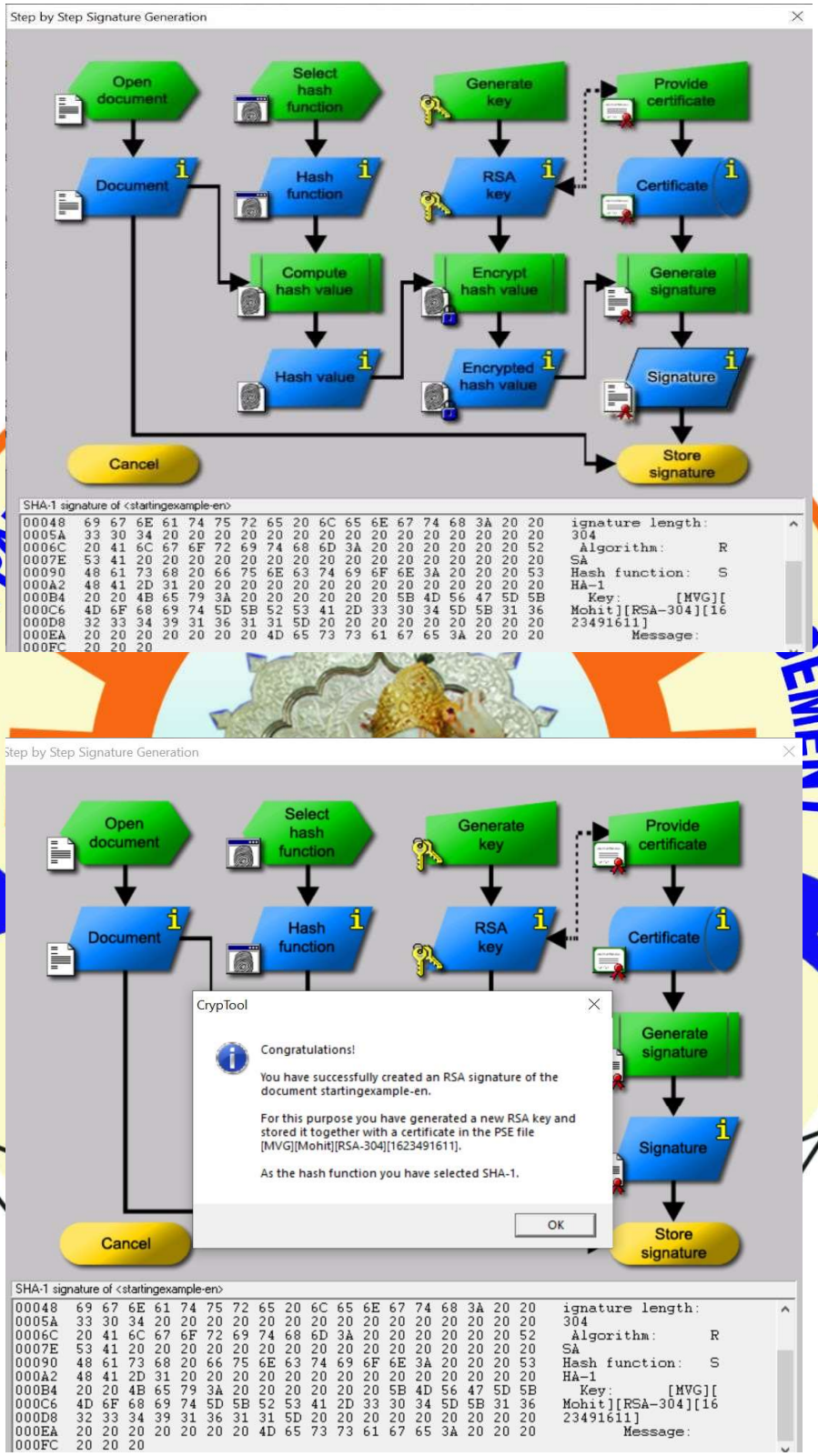
- We need to create a certificate associated with the RSA key. Provide the following details and click on “create certificate.” It’ll be used for communication between the sender and recipient.



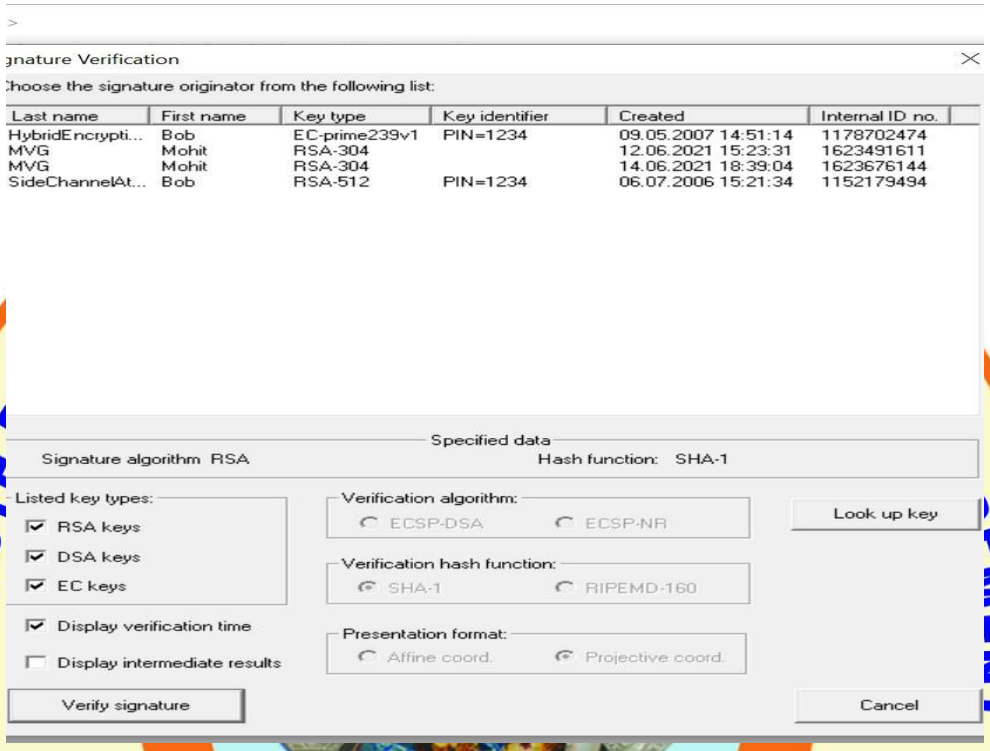
- Click on generate signature to create a digital signature.



- Click on “store signature.”



- Verifying the Digital signature.
  - Click on “Digital Signatures” > Verify signature.
  - Select the Digital Signature created above



- This message confirms a correct signature.

Conclusion : Thus we have successfully create digital signature using online tool.

Viva Questions:

1. What is digital signature?
2. What are the steps to create digital signature?
3. What do you mean by sign the document?
4. A digital signature is a mathematical technique which validates?

   A. authenticity B. integrity C. Non-repudiation D. All of the above

5. How many algorithms digital signature consists of?

   A. 2 B. 3 C. 4 D. 5

ASSESSMENT SCHEME:-

Pre-Lab Test (2)	In Lab performance (5)	Post Lab Test (3)	Record (5)	Total (15)

Practical No. 10

Aim: - Write a Program in C to Implement Hill Cipher technique.

Objective: Students will understand how hill cipher technique work and how encryption is done in it.

Software Required: Turbo C / Dev C++.

Theory: -

Hill cipher is a type of polygraphic substitution cipher invented by Lester S. Hill in 1929. It is a block cipher that operates on groups of letters, rather than individual letters. The basic idea behind the Hill cipher is to use matrix multiplication to transform plaintext into ciphertext.

Here's how it works:

Key generation: A square matrix of size n x n is selected as the key, where n is an integer. The key matrix must be invertible (i.e., have a non-zero determinant) in order for the encryption and decryption processes to work correctly.

Message preparation: The plaintext message is divided into blocks of n letters, where n is the size of the key matrix. If the last block of the plaintext is shorter than n letters, it is padded with additional letters to make it the same size as the other blocks.

Encryption: Each block of plaintext is multiplied by the key matrix modulo 26 (i.e., each element of the resulting matrix is the remainder of the corresponding element in the plaintext matrix multiplied by the corresponding element in the key matrix divided by 26). The resulting matrix is then converted back into letters using a simple substitution table (e.g., A = 0, B = 1, etc.) to produce the ciphertext.

Decryption: The ciphertext is divided into blocks of n letters and each block is multiplied by the inverse of the key matrix modulo 26 (i.e., each element of the resulting matrix is the remainder of the corresponding element in the ciphertext matrix multiplied by the corresponding element in the inverse key matrix divided by 26). The resulting matrix is then converted back into letters using the same substitution table used for encryption to produce the plaintext.

The Hill cipher is relatively strong against known-plaintext attacks (i.e., attacks in which an attacker knows some of the plaintext and corresponding ciphertext), but it is vulnerable to brute-force attacks (i.e., attacks in which an attacker tries every possible key until the correct one is found) if the key size is small. Therefore, the security of the Hill cipher depends on the size and complexity of the key matrix.

Algorithm:

1. Initialize a[3][3] and b[3][3] matrices with integer values.
2. Read plain text msg as input.
3. Convert each character of msg to its respective numerical value using ASCII code by subtracting 65 from it and store in an array c[20].
4. For encryption, perform matrix multiplication of a[3][3] and c[3][1] and store in d[3][1] matrix. a. Initialize t=0 b. for i=0 to 2, do i. t=0 ii. for j=0 to 2, do 1. t=t+(a[i][j]*c[j]) iii. d[i]=t mod 26
5. Display the encrypted cipher text by converting each number in d[3][1] array back to its corresponding character value using ASCII code by adding 65 to it.
6. For decryption, perform matrix multiplication of b[3][3] and d[3][1] and store in c[3][1] matrix. a. Initialize t=0 b. for i=0 to 2, do i. t=0 ii. for j=0 to 2, do 1. t=t+(b[i][j]*d[j]) iii. c[i]=t mod 26

Program:

```
#include<stdio.h>
#include<math.h>

float encrypt[3][1], decrypt[3][1], a[3][3], b[3][3], mes[3][1], c[3][3];

void encryption(); //encrypts the message
void decryption(); //decrypts the message
void getKeyMessage(); //gets key and message from user
void inverse(); //finds inverse of key matrix

void main()
{
getKeyMessage();
encryption();
decryption();
}

void encryption()
{
        int i, j, k;
        for(i = 0; i < 3; i++)
        for(j = 0; j < 1; j++)
```

```
        for(k = 0; k < 3; k++)
encrypt[i][j] = encrypt[i][j] + a[i][k] * mes[k][j];
printf("\nEncrypted string is: ");
for(i = 0; i < 3; i++)
printf("%c", (char)(fmod(encrypt[i][0], 26) + 97));

}

void decryption()
{
        int i, j, k;
        inverse();
for(i = 0; i < 3; i++)
for(j = 0; j < 1; j++)
for(k = 0; k < 3; k++)
decrypt[i][j] = decrypt[i][j] + b[i][k] * encrypt[k][j];
printf("\nDecrypted string is: ");
for(i = 0; i < 3; i++)
printf("%c", (char)(fmod(decrypt[i][0], 26) + 97));
printf("\n");
}

void getKeyMessage()
{
int i, j;
char msg[3];

printf("Enter 3x3 matrix for key (It should be inversible):\n");
for(i = 0; i < 3; i++)
for(j = 0; j < 3; j++)
{
        scanf("%f", &a[i][j]);
        c[i][j] = a[i][j];
}
printf("\nEnter a 3 letter string: ");
scanf("%s", msg);
for(i = 0; i < 3; i++)
mes[i][0] = msg[i] - 97;
}

void inverse()
{
int i, j, k;
float p, q;
for(i = 0; i < 3; i++)
for(j = 0; j < 3; j++)
```

```
{
if(i == j)
b[i][j]=1;
else
b[i][j]=0;
}
        for(k = 0; k < 3; k++)
        {
                for(i = 0; i < 3; i++)
                {
                        p = c[i][k];
                        q = c[k][k];
                        for(j = 0; j < 3; j++)
                        {
                                if(i != k)
                                {
                                        c[i][j] = c[i][j]*q - p*c[k][j];
                                        b[i][j] = b[i][j]*q - p*b[k][j];
                                }
                        }
                }
        }
for(i = 0; i < 3; i++)
for(j = 0; j < 3; j++)
b[i][j] = b[i][j] / c[i][i];
printf("\n\nInverse Matrix is:\n");
for(i = 0; i < 3; i++)
{
        for(j = 0; j < 3; j++)
        printf("%d ", b[i][j]);
printf("\n");
}
}
```

Output:

Enter 3x3 matrix for key (It should be inversible):

0 3 2

6 0 4

0 2 5

Enter a 3 letter string: ACT

Encrypted string is: UWL

Inverse Matrix is:

1 -180514652 -180514652

1 -180514652 -180514652

1 -180514652 -180514652

Decrypted String is: ACT

Conclusion:

Thus, in this way we have encrypted plain text using Hill Cipher Technique.

Viva Questions:

1. Encryption in hill cipher is done using which technique?
2. Hill cipher is an example of which cipher?
3. What will be the size of a key matrix if the plain text is "STUDY"?
4. Hill cipher requires prerequisite knowledge of?
5. Encryption in hill cipher is done using matrix multiplication. TRUE or FALSE?

ASSESSMENT SCHEME:-

Pre-Lab Test (2)	In Lab performance (5)	Post Lab Test (3)	Record (5)	Total (15)