PRACTICAL No. 2

Aim :- Write a program that contains a string with a value 'Hello World'. The program should AND and XOR, OR each character in this string with 127 and display result.

Theory & Algorithm :-

We know the computers save the data in bits format. And thus 'characters or alphabets are also stored in bits format. This is done using ASCII (American standards Code for Information Interchange) values. ASCII table maps the characters, alphabets to the particular number. It starts from 0 to 127.

For alphabets it starts from 65 for capital letters and for small letters it starts from 97.

We are going to program in python.

Algorithm :-

```
i) store 'Hello World' in variable.
ii) Parse the string character by character
iii) get ascii value for current character
iv) perform AND (&), OR(|) and XOR(^)
    between ascii value of current character and 127.
v) store the results (add in) different
    variables.
```

vi) get the character from the resulted and store in variables.

vii) print the three operation variables.

Conclusion :- In this practical we performed AND, OR and XOR operation on characters of given string and printed the resulted strings.

PRACTICAL NO. 3

Aim :- Design and Implement symmetric encryption algorithm based on Feistel structure

Theory & Algorithm :- symmetric key cryptography is a type of encryption scheme in which the similar key is used both to encrypt and derypt messages.

feistel cipher is a design model that derives different symmetric block ciphes, such as DES. It uses the same key for encryption and deryption process thus a symmetric encryption algorithm.

Algorithm :- Encryption

i) Convert plain text into binary using ASCII code.

ii) Divide the data into blocks, processed one at a time.

iii) The encryption process takes two inputs, one block of data and a key.

iv) When the block is ready for the encryption process, divide it into two halves of equal length. The left half is denoted Lo and right half is denoted by Ro.

v) Data is passed through n rounds of execution, where n is specified by the design of Algorithm (here we take n=1)

vi) Each round uses the same encryption

function and different subkey generated from master key.

vii) To generate the left half of the next round, L_{i+1} the current right half R_i is assigned to it.

viii) To generate right half of the next round, R_{i+1} the current right half R_i undergoes the following steps:-

a) R_i and subkey k_i is passed through round function.

b) The result of a is XORed with left half of current round L_i

c) result from b is assighed to right half of next round.

ie $L_{i+1} = R_i$

$R_{i+1} = F(R_i, k_i) \oplus L_i$

ix) The left and right half of data obtained after n rounds is swapped again before concluding the Feistel cipher.

Decryption - it follow same process just with difference that reverse order of keys used.

Conclusion :- In this practical, we designed and implemented symmetric encryption algorithm based on Feistel cipher where number of rounds performed was 1 ie $n=1$.

PRACTICAL No. 4

Aim:- Implement DES and RSA algorithm.

Algorithm :- Data Encryption Standard (DES) is an algorithm for data encryption which uses symmetric key of 56 bit size and block data of 64 bit size. It perform 16 rounds.

steps for encryption :-

i) Text converted to 64 bits blocks (binary using ASCII)
ii) Initial permutation
iii) spliting of data in 32 bits two blocks. Left block is Lo and Right block is Ro.
iv) sub key is generated containing 48 bit (from 56 bits)
v) Right block is passed to round function. which performs following steps.
   a) 32 bit Ro is expanded into 48 bits
   b) 48 bit Ro is XORed with ki (48 bits)
   c) Result of step b is passed to S boxes to make it 32 bits again.
   d) result of c is passed to permutation P box.
   e) result from d is XORed with Li and passed to Ri+1.

ie L_{i+1} R_i

$R_{i+1} : F(R_i, k_i) \oplus L_i$

vi) 16 rounds of this is performed.
vii) after this swapping of left and right block is done
viii) Reverse permutation is done and encryption ends.

Decryption - same as encryption only keys used are reverse in order.

RSA algorithm :- Encryption -

step 1 - select two prime numbers namely p and q, and then calulate N ie $N = p \times q$

step 2 - choose a number 'e' less than 'n' such that n is relatively prime to $(p-1) \times (q-1)$ i-e e and $(p-1) \times (q-1)$ have no common factor except 1. e is $1 < e < \phi(n)$, e is prime $\phi(n)$ ie. $\gcd(e, \phi(n)) = 1$

step 3 - public key <e, n>. Encryption is done using formula.
$c = m^e \bmod n$ where m → message
c → cipher text

step 4 - to find private key, $De \bmod (\phi(n)) = 1$
private key is <d, n>

step 5 - Decryption is done using formula
$m = c^d \bmod n$ where c → cipher text
m → decrypted message

<u>Conclusion</u> :- In this practical we implemented DES and RSA algorithms.

PRACTICAL No. 5

Aim :- Demonstrate how Diffe-Hellman key exchange works with Man-In-Middle attack.

Theory and Algorithm :- Diffe-Hellman key exchange is a method of securely exchanging cryptographic keys between two parties over unstrusted commonication channel. Here is demonstration of Man-In Middle attack in Diffe-Hellman.

suppose we have 3 parties - Alice, Bob and Eve. Alice and Bob want to share secret key but Eve intends to intercept and modify communication.

step 1 - Alice, Bob agree on common modulus (P) and base (g) they make these public.
Alice, Bob each choose private key (a and b resp.) without revealing them.

step 2 - Alice, Bob exchange public keys,
Alice's public key $A = g^a \% P$
Bob's public key $B = g^b \% P$

step 3 - Eve intercepts Alice and Bob's public key (A, B)

step 4 - Eve generate her own key pair (e) and compute two fake public keys.
Fake Alice's public key $A' = g^e \% P$
Fake Bob's public key $B' = g^e \% P$

step 5 - Eve Relays fake public keys i.e. send A' to Bob and B' to Alice

Step 6 - Alice compute what she thinks shared key with Eve's fake key.

shared secret with Eve : $A'^{b} \% P$

Similarly happens with : $B'^{a} \% P$

Step 7 - Eve now has both S-Eve from Alice and S-Eve from Bob, which are not same. Eve can decrypt and intercept the message between Alice and Bob and then re-encrypt them to maintain her position as man-in-middle.

In this scenario, Diffe-Hellman key exchange is vulnerable to MITM attack because Eve can generate fake key to intercept and manipulate the communication between Alice and Bob. To prevent it in real-world, additional security measures like digital signature or public key infrastructure are used to ensure authenticity of exchanged keys.

<u>Conclusion</u> :- In this practical we studied and demonstrated how Diffe-Hellman key exchange works with Man-In-The-Middle attek.

PRACTICAL No. 7

Aim :- Calculate the message digest of text using SHA-1 algorithm in Java.

Algorithm:- SHA-1 is a hash function that takes input message and produces fixed-size (160 bit) hash value. Here is how it works -

1. Padding message - SHA-1 operate on 512 bits (64 byte) so if message not multiple of 64 it is padded padding start with '1' followed by '0'
2. Message Block Processing - padded message is divided in 512 bits blocks processed one at a time
3. Initial Hash values - SHA-1 uses five 32 bit (A, B, C, D and E) as initial hash value. These are usually derived from binary representation of square roots of small prime number
4. Main Hashing loop - for each 512 bit, the block is divided into 16 32bit words. A series of logical function and bitwise operation are applied.
5. Word Expansion - 32 bit words is expanded to 80 words.
6. Rounds - SHA-1 operates 80 rounds in total. Each round apply different AND, OR, XOR, addition modulo 2^{32} and bit rotation operations.
7. Updating Hash values - After processing 80 rounds, the resulting hash value are updated
8. Final Hash value - Finally concatenation of these 5 32bit word is returned as final 160 bit hash value.

Algorithm for program –

1. Import required classes for working with SHA-1 algorithm. like MessageDigest.
2. create 'MessageDigest' instence with SHA-1 algorithm.
3. Compute the hash by updating digest with the bytes of input text.
4. Convert the resulting hash from a byte array to hexadecimal string using the 'byteesToHex' method.
5. Finally print the input text and SHA-1 hash.

Conclusion :- In this practical we calculated message digest for given input text using SHA-1 algorithm in Java program.