ISP M2

Name: ISHANI PATIL

Roll No.: 5032

MBA Tech Data Science

Question 1:

(1)

→ (i) Diffie - Hellman key exchange, also called exponential key exchange is a method of digital encryption that uses numbers raised to specific powers to produce decryption keys on the basis of components that are never directly transmitted, making the task of a would-be code breaker mathematically overwhelming.

(ii) As the name suggests, this algorithm is used to exchange the secret key between the sender and the receiver.

(iii) This algorithm facilitates the exchange of secret key without actually transmitting it.

(iv) Example : Credit card transaction email

(2)

→ Given:

Prime value (n) = 17

Primitive root (a) = 5

Private key of Alice = 4

Private key of Bob = 6

Solution:

Step 1:

Public key of Alice:

$= 5^{\text{private key of Alice}} \bmod 17$

$= 5^4 \bmod 17 = 13$

Public key of Bob

$= 5^{\text{private key of Bob}} \mod 17$

$= 5^6 \mod 17$

$= 2$

Step 2:

Secret key obtained by Alice:

$= 2^{\text{private key of Alice}} \mod 17$

$= 2^4 \mod 17$

$= 16$

Secret key obtained by Bob:

$= 13^{\text{private key of Bob}} \mod 17$

$= 13^6 \mod 17$

$= 16$

Finally, both obtain same value of secret key. = 16

Thus, option (a) is correct.

(3)

→ Encryption:

The plain text (P) and key (K) are added mod 26.

$E_i = (P_i + K_i) \mod 26$

Decryption:

$D_i = (E_i - K_i + 26) \mod 26$

(4)

→

```
x = lambda a,b : a * b
print (x(5,6))
```

output :- 30

Question 2:

→ (i) To implement Diffie-Hellman, the two end users Alice and Bob, while communicating over a channel they know to be private, mutually agree on positive whole numbers p and q, such that p is a prime number and q is a generator of p.

(ii) The generator q is a number that, when raised to positive whole number powers less than p, never produces the same result for any two such whole numbers.

(iii) The value of p may be large but the value of q is usually small.

Alice	Bob
(a) Public keys available = P, G	(a) Public keys available = P, G
(b) Private key selected = a	(b) Private key selected = b
(c) Key generated $x = G^a \bmod P$	(c) Key generated $y = G^b \bmod P$
Exchange of generated keys takes place	
(d) Key received = y	(d) Key received = x
(e) Generated secret key $K_a = y^a \bmod P$	(e) Generated secret key $K_b = x^b \bmod P$

Algebraically, it can be shown that

$K_a = K_b$

Question 3:

→ (i) Vigenere cipher is a method of encrypting alphabetic text. It uses a simple form of poly alphabetic substitution.

(ii) A polyalphabetic cipher is any cipher based on substitution, using multiple substitution alphabets.

(iii) The encryption of the original text is done using the vigenere square or vigenere table.

(iv) The table consists of the alphabets written out 26 times in different rows, each alphabet shifted cyclically to the left compared to the previous alphabet, corresponding to the 26 possible caesar ciphers.

(v) At different points in the encryption process, the cipher uses a different alphabet from one of the rows. The alphabet used at each point depends on a repeating keyword.

Input : Plain text : GEEKFORGEEKS

Keyword : ISHANI

Output : cipher text : GCYCZFMLYLEIM

For generating key, the given keyword is repeated in a circular manner until it matches the length of the plain text.

The keyword "ISHANI" generates the key "ISHANIISHANI"

Encryption:

$E_i = (P_i + K_i) \bmod 26$

Decryption:

$D_i = (E_i - K_i + 26) \bmod 26$

Question 4:

→ string = " GEEKS FOR GEEKS"

keyword = " ISHANI"

```
def generateKey(string, key):
    key = list(key)
    if len(string) == len(key):
        return(key)

    else:
        for i in range(len(string) - len(key)):
            key.append(key[i % len(key)])
    return("".join(key))

def encryptCipherText(string, key):
    cipher_text = []
    for i in range(len(string)):
        x = ((ord(string[i]) + ord(key[i])) % 26) + ord('A')
        cipher_text.append(chr(x))
    return("".join(cipher_text))

key = generateKey(string, keyword)

print("Original Message", string)
print("Keyword:", keyword)
cipher_text = encryptCipherText(string, key)
print("Ciphertext:", cipher_text)
```

Output: Original Message GEEKS FOR GEEKS

Keyword: ISHANI

Ciphertext: YLEBSSGYGVEXK