NAME: Abey George SUBJECT: M-2 ISP PAGE NO.:
TOPIC: DATE: TERM: ROLL NO.: 5034 STD: DIV: ASSIGNMENT NO.:

Q1] What is Diffie-Hellman key exchange, ~~also called~~ ~~ex~~ state some examples.

→ 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 could be code breaker mathematically overwhelming.

As the name suggests, the algorithm is used to exchange the secret key between the sender and the receiver. The algorithm facilitates the exchange of secret key without actually transmitting it.

Examples:
Credit card transaction, email.

Q2] In a diffie-Hellman key Exchange, Alice and Bob have chosen prime $q = 17$ and primitive root = 5. If Alice's secret key is 4 and Bob's secret key is 6, what is the secret key they exchanged?

→ Option 1] 16.

Q3] Write Encryption and decryption formula for Vigenere Cipher.

→ Encryption: The plaintext (P) and key (K) are added modulo 26.

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

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

ROLL NO.:____ STD:____ DIV:____

TOPIC:____ DATE:____ TERM:____ ASSIGNMENT NO.:____

Q4] Multiplication using Lambda.

→ x = lambda a,b : a * b

→ print (x(5,6))

Q5] Write an steps for Diffie-Hellman key exchange. State.

→ 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. 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. The values of p may be large but the value of q is usually small.

Alice	Bob
→ Public key available = P, G	→ Public keys available P, G
→ Private key selected = a	→ Private key selected = a
→ key Generated = $x = G^a \bmod P$	→ key Generated = $y = G^b \bmod P$
→ Exchange generated keys take place	
→ key recieved = y	→ key recieved = x
→ Generated Secret key = $k_a = y^a \bmod P$	→ Generated Secret key = $k_b = x^b \bmod P$

Algebraically, it can be shown that

$k_a = k_b$

SUBJECT: _____ PAGE NO.: _____

NAME: _____ ROLL NO.: _____ STD: _____ DIV: _____

TOPIC: _____ DATE: _____ TERM: _____ ASSIGNMENT NO.: _____

Q6) What is Vigenere Cipher State its formula. Explain with example.

→ Vigenere Cipher is a method of encrypting alphabetic text. It uses a simple form of polyalphabetic substitution. A polyalphabetic cipher is any cipher based on substitution, using multiple substitution alphabets. The encryption of the original text is done using the Vigenere Square or Vigenere table.

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 Caeser Ciphers. 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 : Plaintext : GEEKSFORGEEKS
keyword : AYUSH
Output : Ciphertext : GCYCZFMLYLEIM

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

Encryption: The plaintext (P) and key (K) are added to modulo 26. $E_i = (P_i + K_i) \bmod 26$

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

eg. string = "GEEKSFORGEEKS" keyword = "SHARAN"
ciphertext : YLEBSSGYGVEXK.

Q7) Write Encryption code for vigenere cipher.

→ String = "GEEKSFORGEEKS"
keyword = "SHARAN"

```
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 encrypt_cipherText(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 = encrypt_cipherText(string, key)
print("Ciphertext :", cipher_text)
```

Original Message : GEEKSFORGEEKS
keyword : SHARAN
CipherText : YLEBSSGYGVEXK