is

May 6, 2025

[1]:

```
#IS-1
def and_xor_with_127(input_str):
    print("Original String:", input_str)
    print("\nResults after AND and XOR with 127:\n")

    for char in input_str:
        and_result = ord(char) & 127
        xor_result = ord(char) ^ 127
        print(f"Character: '{char}' | AND 127: {and_result} | XOR 127: {xor_result}")

# String with special characters properly escaped
input_string = "\\Hello\nWorld"
and_xor_with_127(input_string)
```

```
Original String: \Hello
World

Results after AND and XOR with 127:

Character: '\' | AND 127: 92 | XOR 127: 35
Character: 'H' | AND 127: 72 | XOR 127: 55
Character: 'e' | AND 127: 101 | XOR 127: 26
Character: 'l' | AND 127: 108 | XOR 127: 19
Character: 'l' | AND 127: 108 | XOR 127: 19
Character: 'o' | AND 127: 111 | XOR 127: 16
Character: '
' | AND 127: 10 | XOR 127: 117
Character: 'W' | AND 127: 87 | XOR 127: 40
Character: 'o' | AND 127: 111 | XOR 127: 16
Character: 'r' | AND 127: 114 | XOR 127: 13
Character: 'l' | AND 127: 108 | XOR 127: 19
Character: 'd' | AND 127: 100 | XOR 127: 27
```

[3]:

```
#IS-2
def encrypt_message(message, key):
    num_columns = len(key)
    num_rows = len(message) // num_columns + (len(message) % num_columns != 0)
```

```python
    padding = (num_columns * num_rows) - len(message)
    message += '_' * padding  # Padding with underscores for empty slots

    # Create the matrix row-wise
    matrix = [list(message[i:i+num_columns]) for i in range(0, len(message), num_columns)]

    # Generate the order of columns based on sorted key
    sorted_key = sorted(list(key))
    col_order = [key.index(k) for k in sorted_key]

    # Read column-wise in sorted key order
    ciphertext = ''
    for idx in col_order:
        for row in matrix:
            ciphertext += row[idx]
    return ciphertext

def decrypt_message(ciphertext, key):
    num_columns = len(key)
    num_rows = len(ciphertext) // num_columns

    sorted_key = sorted(list(key))
    col_order = [key.index(k) for k in sorted_key]

    # Create an empty matrix
    matrix = [['' for _ in range(num_columns)] for _ in range(num_rows)]

    # Fill the matrix column-wise
    k = 0
    for idx in col_order:
        for row in range(num_rows):
            matrix[row][idx] = ciphertext[k]
            k += 1

    # Read row-wise to get the original message
    plaintext = ''.join([''.join(row) for row in matrix])
    return plaintext.rstrip('_')  # Remove padding

# Example usage
message = "HELLOTRANSPOSITION"
key = "4312"

cipher = encrypt_message(message, key)
print("Encrypted:", cipher)

plain = decrypt_message(cipher, key)
```

```
print("Decrypted:", plain)
```

```
Encrypted: LRPT_LAOI_ETSINHONSO
Decrypted: HELLOTRANSPOSITION
```

[5]:

```
#IS-3
#sudo apt update
#sudo apt install python3-pip
#pip3 install pycryptodome
from Crypto.Cipher import DES
from Crypto.Random import get_random_bytes
from Crypto.Util.Padding import pad, unpad

def des_encrypt(message, key):
    cipher = DES.new(key, DES.MODE_ECB)
    padded_text = pad(message.encode(), DES.block_size)
    encrypted_text = cipher.encrypt(padded_text)
    return encrypted_text

def des_decrypt(encrypted_text, key):
    cipher = DES.new(key, DES.MODE_ECB)
    decrypted_padded_text = cipher.decrypt(encrypted_text)
    decrypted_text = unpad(decrypted_padded_text, DES.block_size)
    return decrypted_text.decode()

# DES key must be exactly 8 bytes
key = b'8bytekey'  # You can also use get_random_bytes(8)

# Example usage
message = "HelloDES"
print("Original:", message)

ciphertext = des_encrypt(message, key)
print("Encrypted:", ciphertext.hex())

decrypted = des_decrypt(ciphertext, key)
print("Decrypted:", decrypted)
```

```
Original: HelloDES
Encrypted: f40d2b29ec13b758974620abef5ef24a
Decrypted: HelloDES
```

[4]:

```
#IS-4
import math

p = int(input("Enter a prime number p : "))
q = int(input("Enter a prime number q : "))
```

```python
n = p*q
phi = (p-1)*(q-1)

e=2
while e<phi:
    if math.gcd(e,phi) == 1:
        break
    e += 1

k=2
d=((k*phi)+1)//e

print("\nPublic key:(",e,",",n,")")
print("\nPrivate key:(",d,",",n,")\n")

msg = int(input("Enter a number message(less than n):"))

C = pow(msg,e,n)
print("Encrypyed message: ",C)

M = pow(C,d,n)
print("Decrypted message : ",M)
```

```
Public key:( 5 , 91 )

Private key:( 29 , 91 )

Encrypyed message:  63
Decrypted message :  7
```

#IS-6 <!DOCTYPE html>

Diffie-Hellman keyExchange

Diffie-Hellman Key Exchange

[ ]: